Міністерство освіти і науки України
Чернівецький національний університет
імені Юрія Федьковича

Л.М. Мельничук, В.М. Лучко, Г.М. Перун

Інтерпретована динамічна візуальна мова програмування (Scratch)

Навчальний посібник

Чернівці
Чернівецький національний університет
Імені Юрія Федьковича
2021

УДК 004.439Scratch(075.8)
М 483

Рекомендовано Вченою радою Чернівецького національного університету імені Юрія Федьковича (Протокол №7 від 30 червня 2021 р.)

Рецензенти:

Семчук А.Р. – канд. фіз.-мат. наук, доцент, доцент кафедри методики викладання природничо-математичних дисциплін ІППОЧО;

Лавро С.В. – вчитель інформатики вищої категорії Хмельницької середньої загальноосвітньої школи І-ІІІ ступенів №20, вчитель-методист.

Мельничук Л.М.

М 483 Інтерпретована динамічна візуальна мова програмування (Scratch): навчальний посібник/ Л.М. Мельничук, В.М. Лучко, Г.М. Перун. – Чернівці: Чернівецький нац. ун-т ім. Ю. Федьковича, 2021. 128 с.

У посібнику описано програмне середовище Scratch 1.4 та розглянуто основні методи і прийоми створення проектів у ньому.

Для студентів спеціальностей Середня освіта (математика) і Середня освіта (інформатика), учнів закладів середньої освіти, вчителів інформатики та тих, хто вивчає Scratch самостійно.

УДК 004.439Scratch(075.8)

Передмова

Дисципліна «Інтерпретована динамічна візуальна мова програмування (Scratch)» читається для студентів спеціальностей Середня освіта (математика) та Середня освіта (інформатика). Тут вивчається програмне середовище Scratch, яке учні початкових та середніх класів закладів середньої освіти використовують на уроках інформатики при вивченні розділу «Алгоритми та програми». Виклад здійснюється в основному для версії Scratch 1.4, оскільки практично всі шкільні підручники 3-7-х класів орієнтовані на цю версію [8-12].

Навчальний посібник написаний на основі лекцій та практичних занять, проведених в ЧНУ, і є спробою зібрати і упорядкувати відомості про програмне середовище Scratch та основні методи і прийоми створення проектів у ньому.

Посібник складається із семи тем та індивідуальних завдань восьми практичних робіт. Виклад лекційного матеріалу проілюстрований більш ніж 45-ма розв'язаними прикладами, на основі яких можна вчитися програмувати у Scratch і виконувати практичні роботи. Крім цього, у посібнику є додатки про призначення всіх блоків-команд версії 1.4 та про нумерацію кольорів.

При вивченні дисципліни рекомендується спочатку опрацьовувати теоретичний матеріал, апробуючи наведені там розв'язані приклади. Після цього виконувати відповідне завдання свого варіанту.

Зміст

Тема 1. Поняття про мову програмування Scratch

1. Історія створення та призначення Scratch.
2. Завантаження та встановлення середовища Scratch.
3. Основні поняття та елементи інтерфейсу.
4. Збереження, відкривання та редагування проекту.
5. Вибір нового спрайта. Розміщення та переміщення спрайта.

1. Історія створення та призначення Scratch

Для навчання програмування дітей треба використовувати спеціальні прості мови програмування. «Дорослі» мови не підходять, бо

- учні можуть робити помилки при написанні команд;
- результат написання програми видно лише після написання всієї програми;
- важко організувати взаємодію користувача і комп'ютера у процесі виконання програми.

Тому виникла необхідність створення дуже простої мови програмування для дітей, яка би обходила ці складнощі. Вона має бути:

- інтерактивною (щоб була взаємодія з комп'ютером в процесі виконання програми);
- динамічною (здатною до розвитку і видозмін);
- візуальною (щоб було видно виконання кожного етапу програми).

Вперше цей підхід було здійснено у мові програмування **Logo** (1967р.), де виконавцем є черепашка, що може виконувати прості команди. З цих команд, як з цеглинок у грі Лего, можна будувати складніші програми. Пізніше ця мова удосконалювалась і видозмінювалась.

Вже в середині 90-х років професор **Мітчел Рєзнік** (США) і його група розробили продукт **StarLogo**, в якому діяло багато черепашок. Синтез ідей **StarLogo** і можливостей мови **Squeak** привели до створення мови **Scratch**, яка стала

навчальним середовищем для програмування для дітей, починаючи з 7 років або і ще менше.

Перша версія **Scratch** представлена **8 січня 2007** року дослідницькою групою «Дитячий садок на все життя» під керівництвом професора **Мітчела Рєзніка** при Массачусетському технологічному інституті (США) при фінансовій підтримці Національного наукового центру, компаній Microsoft, Intel Foundation Nokia та Каліфорнійського університету. **9 травня 2013 року** було представлено Скретч 2.0, а **2 січня 2019 року** з'явився Скретч 3.0.

У школі на даний час програмування на основі Scratch вивчають діти у 3 – 7 класах (починаючи з 2013 р.). У навчальних підручниках використовується переважно версія 1.4.

Тлумачення назви

У перекладі з англійської іменник ***scratch*** має багато тлумачень: «карлючки», «скрип», «дряпання», «насічка», «мітка», «стартова межа» і т.і. Дієслово ***to scratch***, крім очікуваного «дряпати» і «шкрябати», додатково має варіанти перекладу: «рити кігтями» і «надряпати лист або малюнок». Будучи частиною виразів, це слово часто перекладають як «випадково, поспішно, на швидку руку, використання для чернеток або нарисів». Вираз «**start from scratch**» перекладають «почати з нуля». Дряпання кішок — це також *scratch*. Мабуть тому символом програми служить веселий рудий кіт  .

Сфера використання

Основна мета цього програмного середовища — навчити дітей основних понять програмування шляхом створення програм-проектів, що містять програмовані об'єкти. Scratch дозволяє створювати інтерактивні програми шляхом комбінування блоків-команд, залучаючи різноманітні графічні об'єкти, зображення, звуки та музику. Учні можуть створювати у середовищі Scratch власні анімовані та інтерактивні проекти:

- ігри,
- мультфільми,
- фільми,

- тести,
- презентації,
- анімаційні та інтерактивні історії тощо.

Це середовище можуть використовувати як молодші школярі, так і студенти, бо легко почати роботу і є можливість створити складні проекти.

Scratch дозволяє створювати потужні програми, бо має хороший математичний апарат, можливості для роботи з цілими числами та числами з десятковою комою, з рядковими змінними, масивами, має оператори циклу та розгалуження. Наприклад, у Scratch було реалізовано алгоритми розв'язання квадратних рівнянь, систем лінійних рівнянь, знаходження площі трикутника, розклад числа на прості множники тощо.

Важливо, що проекти у Scratch можна легко показати іншим користувачам у всьому світі, бо організована ***міжнародна інтернет-спільнота***, членами якої може бути кожен, хто програмує у цьому середовищі.

2. Завантаження та встановлення середовища Scratch

Використання Scratch безкоштовне, не потребує додаткового ліцензування, можна завантажувати на необмежену кількість комп'ютерів та передавати третім особам. Завантаження здійснюється на комп'ютери та ноутбуки з операційними системами Windows, Macintosh чи Linux. Є кілька версій Scratch . Використовуватимемо **версію 1.4.**

Є два способи роботи в Scratch:

1) з використанням Інтернету в браузері;
2) без підключення до Інтернету шляхом встановлення офлайнового редактора.

При першому способі роботи спочатку треба зареєструватися на сайті scratch.mit.edu. Для цього треба:

- вказати ім'я (логін) – краще видумане;
- ввести два рази пароль;
- після цього виконувати інструкції, вказавши інформацію про себе.

Далі для початку роботи клацнути на папку S, тоді після переходу на сторінку «Мої роботи» натиснути «+ Новий проект». Відкриється головне вікно Scratch.

При другому способі роботи, щоб встановити програму Scratch на своєму комп'ютері, потрібно:

1. Перейти за посиланням https://scratch.mit.edu/scratch_1.4/
2. Обрати програму відповідно до своєї операційної системи:
   - для Mac OS X вибрати посилання MacScratch1.4.dmg;
   - для ОС Windows вибрати посилання ScratchInstaller 1.4. exe;
   - для Debian / Ubuntu вибрати *Встановіть Скретч через Software Center*.
3. З'явиться вікно завантаження. **Зберегти файл у зручному місці.**
4. Згорнути вікно браузера та знайти завантажений вами файл.
5. **Запустити майстра встановлення програми** Scratch (двічі клацнути на файл, майстер запускається автоматично).
6. Встановити програму.
7. У вікні програми Scratch, яке відкриється після встановлення, обрати українську мову.

Після встановлення програми Scratch на комп'ютері, на робочому столі появиться ***ярлик з рудим котом*** 

. Клацнувши на нього двічі, відкриємо головне вікно Scratch.

Інший спосіб відкрити Scratch – за допомогою головного меню: Пуск – Програми - Scratch - Scratch .

3. Основні поняття та елементи інтерфейсу

Введемо основні поняття.

❖ **Спрайт** — це об'єкт Scratch, що пов'язаний із зображенням, набором змінних і скриптів (програм) , які визначають його поведінку. Фактично спрайт – це виконавець програми.

Зазвичай використовують спеціального виконавця

вказівок — Рудого кота  . Він може рухатися, говорити, змінювати зовнішній вигляд, взаємодіяти з іншими виконавцями на сцені. Можна долучати інших виконавців.

❖ **Скрипт** (script — сценарій, метод) — послідовність вказівок, що визначає, які дії і в якому порядку має виконати певний об'єкт (спрайт).

Скрипти створюють методом сполучення окремих блоків. Один спрайт може мати декілька скриптів, які запускають незалежно дією користувача (натисненням клавіші або кнопки миші), таймером або отриманням повідомлення від іншого спрайту.

❖ **Блок** — це мінімальний фрагмент програми у Scratch: змінна, оператор, функція або структура, що керує. Блоки згруповано у ***8 категорій*** (див. далі ілюстрації): Рух, Вигляд, Звук, Олівець, Керувати, Датчики, Оператори, Змінні. Всі команди одної категорії мають свій колір: синій, фіолетовий, рожевий, зелений, жовтий, голубий, салатовий і оранжевий відповідно. При виборі одної з категорій, команди вибраної групи відображаються у нижній частині вікна.

❖ **Образи** (вигляд спрайта, костюми) — сукупність зображень одного й того ж об'єкта (спрайта), кожне з яких дещо відрізняється від попередніх.

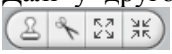
❖ **Звуки** — приєднані звукові ефекти й музика.

❖ **Сцена** — область, в якій діє об'єкт (спрайт) при виконанні програми.

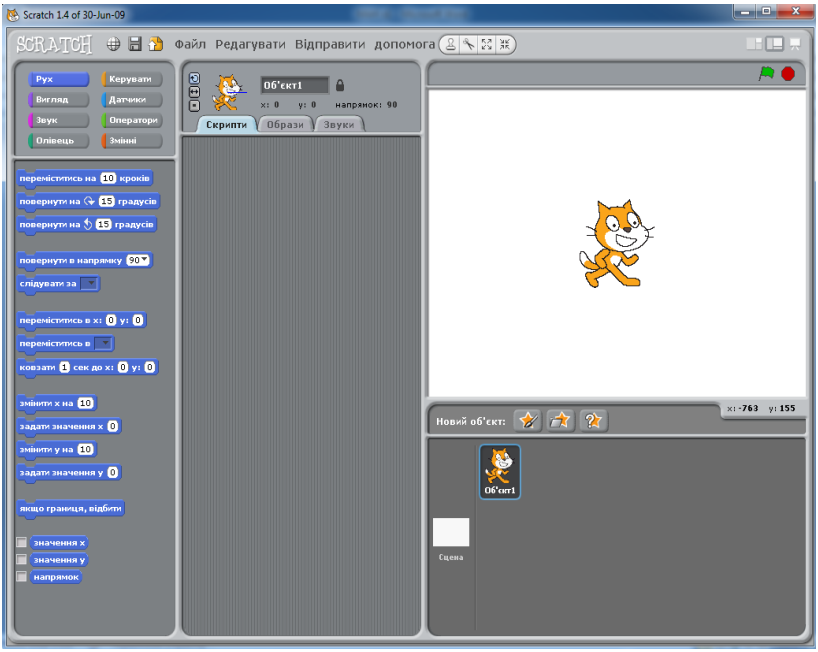
Далі розглянемо структуру головного вікна Scratch у версії 1.4.

У верхньому рядку (***рядку-заголовку***) записана версія 1.4 або назва проекту. Справа у цьому ж рядку є *кнопки управління вікном*  : згортання, відкривання та закриття вікна.

Другий рядок – це ***рядок меню***. Зліва у ньому є три кнопки з малюнками  , де *глобус* означає вибір мови; *дискета* – зберегти проект; *аркуш із стрілкою* – поділитися проектом з іншими (завантажити на сервер Scratch). Далі у

цьому ж рядку опції меню *Файл, Редагувати, Відправити, Допомога*. Далі у другому рядку є ще 4 кнопки управління об'єктом  : дублювати, вилучити, збільшити, зменшити. Три останні кнопки у цьому рядку – кнопки управління розміром сцени  : зменшений розмір сцени, повний розмір сцени, режим перегляду.

Решта головного вікна середовища поділена на 3 частини, згруповані у три стовпчики.



Лівий стовпчик містить **поле команд**. У верхній частині записані категорії блоків: **Рух, Вигляд, Звук, Олівець, Керувати, Датчики, Оператори, Змінні**, кожна з яких має свій колір. При клацанні мишкою на категорію, у нижній частині поля команд відкривається список всіх блоків-команд цієї категорії (палітра блоків). Всі блоки тут такого ж кольору, що і категорія.

Середнє поле головного вікна Scratch – це **поле скриптів**, де створюються програми. Зверху зображений спрайт, для якого складається скрипт, і вказані його властивості. Зліва від спрайта є три *кнопки стилів обертання*:  у всі боки;  вліво-вправо;  без змін.

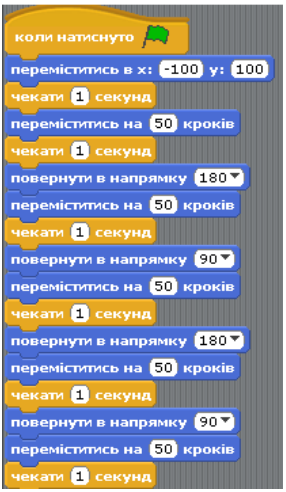
Нижче є три закладки: 

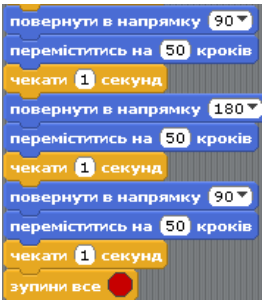
Скрипти, Образи, Звуки, які стосуються вибраного спрайта.

Правий стовпчик містить зверху кнопки *старту проекту*  (зелений прапорець) та *зупинки проекту*  (червоний круг). Нижче є **сцена**, де відображається виконання проекту, а під нею - **поле спрайтів,** де є зображення і назви спрайтів. Голубою рамкою обведений спрайт, для якого написані скрипти у полі скриптів. Тут також є зображення сцени (фон сцени). По замовчуванню фон білий.

При складанні програми блоки-команди з лівої частини вікна перетягують мишкою у середню частину і складають у певному порядку. Блоки з'єднуються між собою.

Приклад 1. Рух кота вниз по сходах.



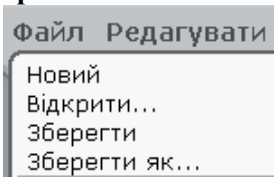


4. Збереження, відкривання та редагування проекту

Збереження проекту

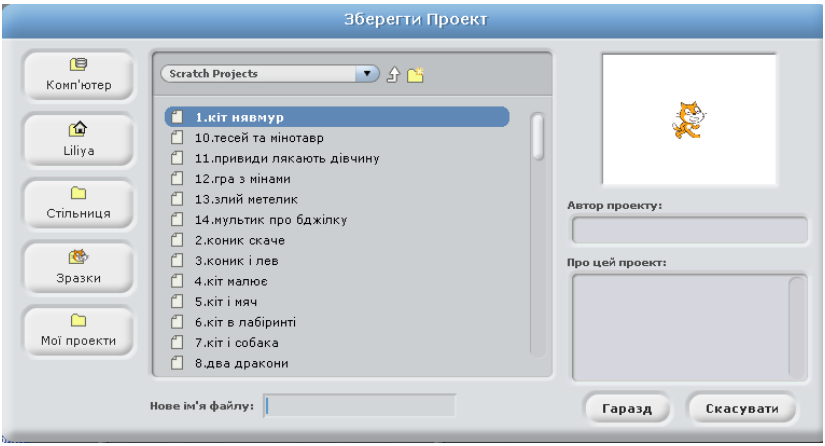
Зберегти проект можна двома способами:

1. В рядку меню вибрати значок **Зберегти цей проект**.
2. Вибрати **Файл - Зберегти.**



Якщо збереження відбувається вперше, то відкривається вікно **Зберегти проект**, де треба вибрати папку і ввести ім'я. Краще свої проекти зберігати у папці **Мої проекти**.

Зробити поточною папку **Мої проекти** можна, вибравши кнопку **Мої проекти** на панелі в лівій частині вікна **Зберегти проект**.



Вибір кнопки **Стільниця** робить поточною папку **Робочий стіл**. А вибрати потрібний носій даних і папку на ньому можна, відкривши список доступних носіїв даних вибором кнопки **Комп'ютер**.

У цьому самому вікні можна за потреби створити нову папку, вибравши кнопку - *папка з зірочкою*.

У вікні збереження проекту можна в поле **Автор проекту** ввести своє прізвище, а в поле **Про цей проект —** короткий опис проекту.

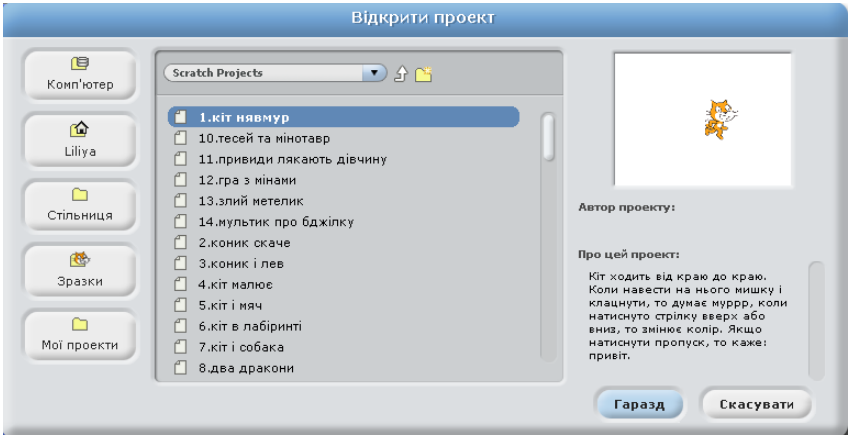
Після введення даних у поля цього вікна потрібно вибрати кнопку **Гаразд.**

Якщо вищевказаними діями проект зберігається не вперше, то жодних вікон не відкривається і він автоматично зберігається в тій самій папці і під тим самим іменем, що були вказані під час першого збереження проекту.

Якщо потрібно зберегти проект в іншій папці і(або) під іншим іменем, то потрібно виконати **Файл – Зберегти як**. Після цього у вікні **Зберегти проект** потрібно вибрати іншу папку і(або) вказати інше ім'я проекту.

Відкривання проекту

Збережений на носії проект можна відкрити в середовищі Scratch. Для цього потрібно виконати **Файл – Відкрити**. У результаті виконання цієї команди відкривається вікно **Відкрити проект** :



У цьому вікні потрібно:

1. Відкрити вміст потрібної папки.
2. Вибрати файл з проектом.
3. Вибрати кнопку **Гаразд.**

Якщо при цьому в **Області скриптів** є інший алгоритм,то відкриється вікно збереження цього проекту.

Якщо у вікні **Відкрити проект** вибрати кнопку **Зразки**, то відкривається список папок, у яких збережено зразки різних цікавих проектів. Ці проекти можна відкрити, змінити і запустити на виконання.

Пропонуємо переглянути приклади готових проектів з бібліотеки, вибравши: **Файл** – **Відкрити** – **Зразки** (ігри – FishChomp; анімація - Aquarium; історії – BeeStory).

Редагування проектів

Алгоритм, розміщений в **Області скриптів**, можна редагувати: видаляти блоки з командами, додавати нові блоки з командами, копіювати і переміщувати блоки, змінювати значення в командах тощо.

Видалити один блок з командою можна одним із трьох способів:

1) перетягнувши його мишкою з **Поля скриптів** до **Поля команд**;
2) натиснути праву кнопку миші і вибрати в контекстному меню команду **Вилучити**;
3) виконавши алгоритм:
   a) вибрати кнопку  **Вилучити** в рядку меню вікна;
   b) установити вказівник на блок, який потрібно видалити;
   c) клацнути ліву кнопку миші.

Останній спосіб зручний для видалення блоку всередині програми.

Видалити групу блоків можна одним із способів:

1) перетягнути цю групу мишкою з **Поля скриптів** до **Поля команд**;
2) вибрати в контекстному меню першого блоку команду **Вилучити**.

Для ***вставки блока*** в алгоритм потрібно перетягнути його з **Поля команд** до **Поля скриптів** і розмістити у потрібному місці алгоритму.

Блок або групу блоків можна ***копіювати*** так:

1. Відкрити контекстне меню блока або першого блока групи.
2. Виконати команду **Дублювати.**
3. Перетягнути утворену копію блока або групи блоків у потрібне місце алгоритму.
4. Клацнути ліву кнопку миші.

5. Вибір нового спрайта. Розміщення та переміщення спрайта

За замовчуванням середовище Scratch пропонує складати алгоритм для виконавця Рудого кота. Але можна використовувати й інших виконавців (спрайтів). Їх зображення розміщені в полі спрайтів під сценою.

Помістити нового виконавця в поле спрайтів можна одним з трьох способів:

- намалювати його в графічному редакторі, вбудованому в середовище Scratch;
- вставити з вибраного файлу;
- вставити з файлу, вибраного випадковим чином.

Для того, щоб самому ***створити новий спрайт***, треба:

- вибрати під сценою кнопку  **Намалювати новий об'єкт;**
- намалювати спрайт;
- натиснути кнопку **Гаразд.**

Після цього намальований спрайт з'явиться в полі спрайтів.

Для того, щоб ***вставити новий спрайт з файлу***, потрібно вибрати під сценою кнопку  **Вибрати новий об'єкт з файлу**. Після цього відкривається вікно **Новий образ**, у якому можна вибрати нового виконавця з колекції Scratch або з будь-якого іншого файлу з графічним зображенням.

Якщо вибрати під сценою кнопку  **Вставити випадковий об'єкт** , то в поле спрайтів вставляється випадковий об'єкт з колекції Scratch.

Щоб ***вилучити спрайт*** з поля спрайтів, треба вибрати команду **Вилучити** в контекстному меню зображення цього

виконавця. Командою **Експортувати цей об'єкт** можна зберегти зображення виконавця на носії.

Для кожного виконавця алгоритм створюється на окремій вкладці **Скрипти**. Під час вибору виконавця в полі спрайтів відкривається вкладка **Скрипти** саме для цього виконавця.

Щоб одночасно запустити на виконання алгоритми для кількох виконавців, потрібно алгоритми кожного з них розпочинати командою 

з категорії **Керувати**.

Розглянемо **<u>розміщення спрайта на сцені</u>**.

Виконавець може рухатися по сцені, а його положення визначається ***координатами x та y*** і кутом повороту, які відображаються у полі скриптів. Абсциса *x* змінюється від -240 до 240, а ордината *y* змінюється від -180 до 180. Спрайт можна переносити мишкою у будь-яку точку. Координати вказівника миші відображаються під сценою.

Кут повороту визначається від додатної півосі ординат: за годинниковою стрілкою кути рахуються від 0 до 180, а проти год. стрілки – від 0 до -180. Тобто

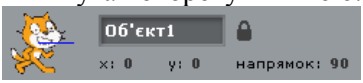
кут 0 – це напрям вверх,

кут 90 – вправо,

кут 180 – вниз,

кут -90 – вліво.

Координати та кут повороту спрайта записано біля його зображення в полі скриптів. Там же на зображенні спрайта є синя стрілка для зміни кута повороту мишкою.



Переміщення виконавця у проекті програмують, використовуючи блоки категорії **Рух.**

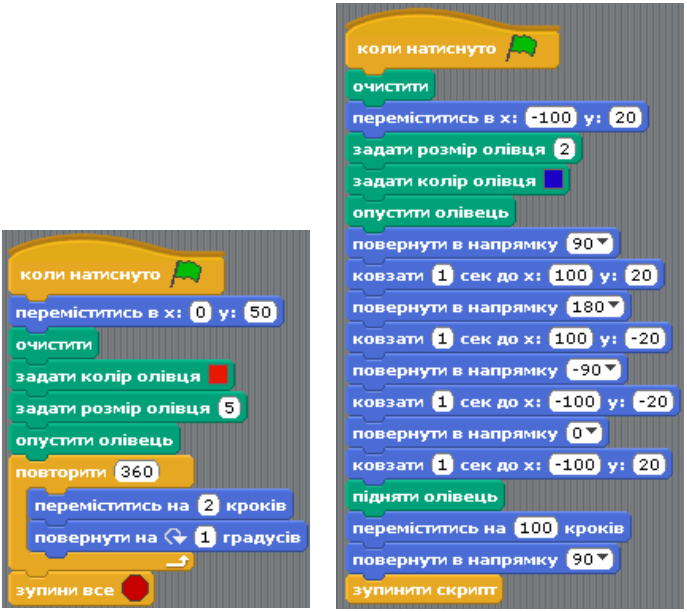
Приклад 2. *(Кіт танцює).* Написати проект для нескінченного руху Кота вліво і вправо на 10 кроків із очікуванням у крайніх точках 1 сек. Зробити рух Кота скінченним (5 разів).

Приклад 3. *(Кіт з другом).* Створити новий проект з двома спрайтами. Крім Кота добавити з бібліотеки спрайтів іншу тваринку, яку відповідно назвати. Кіт рухається по колу, починаючи з точки (0;90). Для цього треба в циклі чергувати команди 

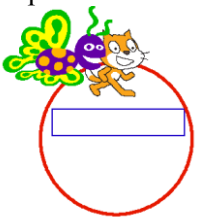
та 

. Інший спрайт рухається по прямокутнику з вершинами (100;200), (-100;20), (-100; -20), (100;-20). Рухи тварин запускаються зеленим прапорцем і малюють за собою лінії.

Перший скрипт для кота, другий для метелика.

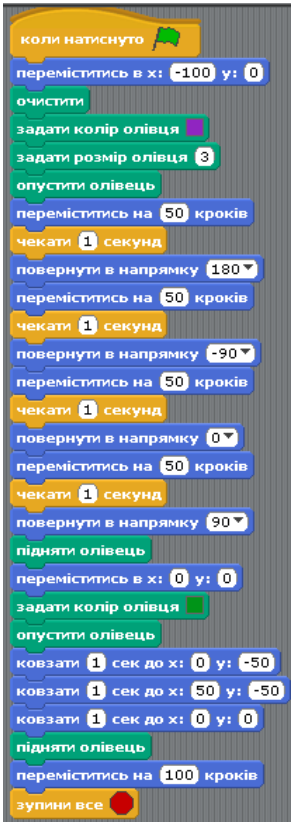


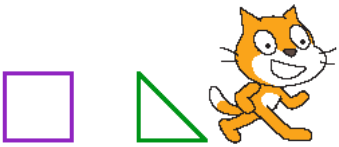
Результат виконання програми:



Приклад 3. *(Кіт малює фігури).* Створити проект, у якому Кіт одну за одною малює різними кольорами квадрат і прямокутний трикутник.

Скрипт для кота і результат малювання:





Практична робота №1.

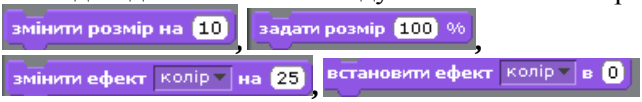
Перше знайомство з Scratch. Спрайти та їх рух

1. Прямолінійний рух спрайта.
2. Повороти спрайта.
3. Зміна вигляду спрайта (кольору, розміру, інших ефектів).
4. Вибір іншого спрайта з бібліотеки.
5. Спрайт малює. Малювання різних фігур.

Завдання

1. По можливості дома завантажити Scratch на своєму комп'ютері за посиланням https://scratch.mit.edu/scratch_1.4/.
2. Ознайомитися з командами руху та повороту для Рудого Кота. Для цього в категорії блоків **Рух** вибирати блоки, перетягати їх мишкою в область скриптів та запускати подвійним клацанням. Змінювати значення чисел у блоках. Слідкувати за координатами Кота та кутами повороту.
3. Розглянути блоки категорії **Керувати**:

4. **Завдання 1** ***(Кіт танцює).*** Написати проект для нескінченного руху Кота вліво і вправо на 20 кроків із очікуванням у крайніх точках 0,5 сек. Проект запускається зеленим прапорцем. Зберегти під назвою *«(Прізвище) Кіт танцює»* у папці «Мої проекти». Експериментувати з числами, поворотом спрайта за допомогою синьої риски. Зробити рух Кота скінченним (15 разів).
5. Відкрити проект *«(Прізвище) Кіт танцює»* та добавити команди для зміни вигляду Кота з категорії **Вигляд:**



Випробовувати різні ефекти з різними числами.

6. **Завдання 2** ***(Кіт з другом).*** Створити новий проект з двома спрайтами. Крім Кота добавити з бібліотеки спрайтів іншу тваринку, яку відповідно назвати. Кіт рухається по колу, починаючи з точки (0,100). Для цього треба в циклі чергувати

команди 

та 

. Інший спрайт рухається по прямокутнику з вершинами (200;50), (-200;50), (-200;-50), (200;-50). Рухи тварин запускаються зеленим прапорцем. Назвати проект *«(Прізвище) Кіт з другом»*. Можна зробити рух по описаних траєкторіях нескінченним.

7. Вивчити команди з категорії **Олівець: опустити олівець, підняти олівець, очистити, задати колір олівця та інші.** У проекті *«(Прізвище) Кіт з другом»* заставити звірят малювати лінію за собою, причому задати різні колір і товщину ліній.
8. **Завдання 3** ***(Кіт малює фігури).*** Створити проект *«(Прізвище) Кіт малює фігури»*, у якому Кіт одну за одною малює різними кольорами і різною товщиною трикутник, квадрат і ромб.
9. **Завдання 4** ***(Олівець-малювець).*** Знайти у бібліотеці спрайтів олівець (Drawing Pencil). Для нього є готовий скрипт. При його виконанні треба натиснути ліву клавішу мишки. Олівець перейде на вказівник. Якщо рухати мишкою, не відпускаючи клавішу, то олівець малює лінію за вказівником. Модифікувати цей скрипт, передбачивши можливість зміни кольору та товщини лінії. Зберегти під назвою *«(Прізвище) Олівець-малювець»*.

Тема 2 . Графічний редактор Scratch

1. Основні об’єкти вікна графічного редактора.
2. Палітра кольорів.
3. Інструменти малювання.
4. Створення та збереження нового спрайта.
5. Редагування. Створення нових образів виконавця
6. Створення і збереження фону сцени.

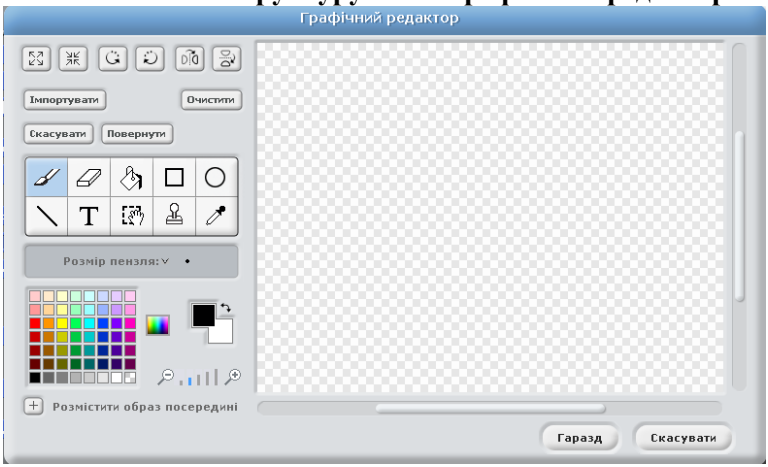
1. Основні об’єкти вікна графічного редактора

Для того, щоб створювати власні спрайти, змінювати існуючі, малювати фон сцени, у Scratch є власний вбудований графічний редактор. **Графічний редактор** – це спеціальна програма для створення малюнків, ілюстрацій та інших графічних об’єктів.

У середовищі Scratch він запускається одним із таких способів:

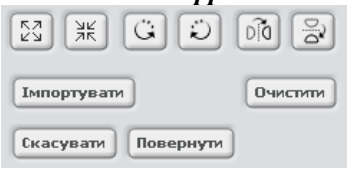
1) натисканням кнопки  **Створення нового об’єкта** під сценою;
2) вибором опції **Малювати** на закладці **Образи** для кожного спрайта;
3) вибором опції на закладці **Фони** для створення нового фону сцени.

Розглянемо **структуру вікна графічного редактора.**

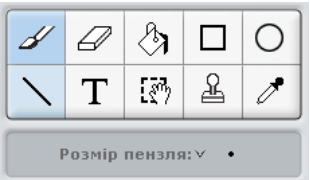


Вікно редактора розділене на дві частини. В *лівій частині* вікна розташовані:

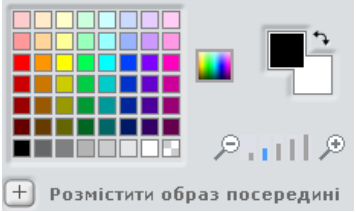
- ***кнопки для виконання дій над фрагментами малюнка***



- ***інструменти малювання***



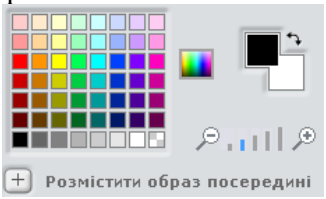
- ***палітра кольорів***



У *правій частині* вікна є ***поле для малювання***. Воно прозоре. Знизу і справа є смуги прокручування. Нижче дві кнопки: **Гаразд** (коли можна використати готовий малюнок) та **Скасувати** (малюнок стирається).

2. Палітра кольорів

Палітра кольорів розташована у нижній лівій частині вікна графічного редактора.

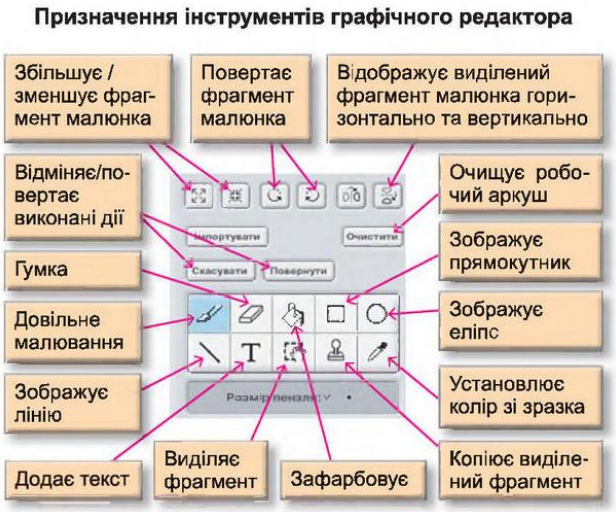


Вона складається з:

- набору готових 56 кольорів;
- поля з кольорами, що плавно переходять в інші;
- двох квадратиків верхнього (основний колір малювання) та нижнього (допоміжний колір);
- шкали зменшення-збільшення малюнка;
- інструмента для розміщення образу посередині.

Вибір нового кольору здійснюється так. При наведенні вказівника на палітру кольорів вказівник перетворюється на піпетку. Наводимо піпетку на потрібний колір і клацаємо ліву клавішу миші. Вибраний колір появиться у верхньому квадратику. При клацанні на праву клавішу миші вибраний колір відображається у нижньому квадратику. Є можливість поміняти кольори місцями у верхньому та нижньому квадратиках.

3. Інструменти малювання



Далі опишемо кожний інструмент.

3.1. Лінія. Інструмент **лінія**  дозволяє малювати на полі прямі лінії.

Як і в інших графічних редакторах (наприклад, Paint), перш ніж малювати, треба вибрати колір та розмір лінії.

Щоб встановити колір лінії, потрібно вибрати його на **палітрі кольорів.** Вибраний колір відобразиться у віконечку справа від палітри.

Щоб встановити розмір лінії, треба:

- вибрати інструмент **лінія** серед інструментів малювання;
- вибрати напис **розмір пензля** у нижньому віконці поля інструментів малювання;
- вибрати розмір серед запропонованих.

Для малювання прямої лінії слід вибрати вказівником миші точку початку лінії і натиснути ліву клавішу миші. Не відпускаючи клавіші, пересунути вказівник миші у кінець лінії та відпустити ліву клавішу миші. Лінія готова.

Якщо при використанні інструмента **лінія** тримати натиснутою кнопку **Shift**, то будуть малюватися тільки горизонтальні або вертикальні лінії.

3.2. Пензель. Інструмент **пензель**  використовується для малювання довільних ліній мишкою. Спосіб малювання, колір та розмір інструмента **пензель** встановлюється аналогічно, як для лінії.

3.3. Гумка. Інструмент **гумка**  використовується для витирання непотрібних елементів малюнка. Для інструмента **гумка** встановлюється лише розмір.

Для витирання вибрати інструмент **гумка**. На полі появиться кружечок вибраного розміру. Він керується мишкою. Підвести кружок до непотрібної частини малюнка і , тримаючи натиснутою ліву клавішу миші, витерти, рухаючи мишею.

Щоб акуратніше малювати чи витирати, можна збільшити малюнок, вибравши + чи – на шкалі справа від палітри кольорів.

3.4. Заливка. Інструмент **заливка**  використовується для заповнення кольором обмеженої області. Якщо область необмежена, то зафарбується все поле для малювання.

Вибір кольору заливки здійснюється клацанням лівої клавіші мишки на потрібний колір на палітрі. Цей колір відображається у верхньому квадратику справа від палітри.

Зафарбувати можна як одним кольором, так і так званою «градієнтною заливкою», колір якої є плавним перетіканням одного кольору в інший. При виборі **заливки** під кнопками інструментів малювання можна вибрати кнопки з однотонним фарбуванням чи трьома способами градієнтної заливки.

Градієнтна заливка визначається кольорами верхнього та нижнього квадратиків справа від палітри. Колір верхнього квадратика вибирається на палітрі клацанням лівої клавіші миші, а нижнього – правої клавіші миші.

3.5. Піпетка. Інструмент **піпетка**  використовується для того, щоб намалювати або зафарбувати щось кольором, який вже використовувався в процесі малювання.

Щоб вибрати колір, треба:

- вибрати інструмент **піпетка;**
- навести вказівник на потрібний колір вже створеного малюнка;
- клацнути лівою клавішею миші.

Після цього новий колір встановиться у верхньому квадратику в палітрі, і можна малювати.

3.6. Текст. Інструмент **текст**  використовується для створення текстових підписів.

При виборі інструмента **текст** у центрі поля для малювання з'являється маркер у вигляді чорного квадрата і текстовий курсор. Після цього можна вводити потрібний текст. Далі під панеллю інструментів можна змінити розмір і шрифт тексту: 

Щоб перемістити введений текст, треба:

- перевести вказівник на чорний квадрат, він перетвориться на долоню;
- утримуючи натиснутою ліву клавішу миші, пересунути текст у потрібне місце;

- відпустити ліву клавішу миші.

Інструменти **текст** і **піпетка** можна використовувати як для створення нових виконавців чи їх образів, так і для малювання нового фону сцени.

3.7. Вибрати інструмент (переміщення, зміна, виділення області). Цей інструмент  використовується для виділення малюнка чи його фрагмента з метою його переміщення в іншу область поля для малювання, для його зміни або вилучення. Послідовність дій:

- вибрати інструмент  ;
- помістити вказівник у лівому верхньому куті від фрагмента малюнка;
- тримати натиснутою ліву клавішу миші і рухати мишею в правий нижній кут так, щоб чорна рамка вмістила потрібний фрагмент;
- відпустити ліву клавішу миші.

У результаті цих дій фрагмент малюнка виділено, тобто поміщено у чорну рамку. Далі з виділеним фрагментом можна робити такі дії:

- ***видалити*** кнопкою **Delete** на клавіатурі;
- ***перемістити***: навести вказівник на рамку (він перетвориться на руку); утримуючи натиснутою ліву клавішу миші, перемістити виділений фрагмент у потрібне місце;
- ***збільшити (зменшити)*** – натиснути кнопку **збільшити**  **(зменшити**  **)** у верхній лівій частині вікна графічного редактора потрібну кількість разів;
- ***дзеркально відобразити*** – вибрати кнопки **повернути горизонтально**  чи **повернути вертикально**  у верхній лівій частині вікна графічного редактора;
- ***повернути*** – вибрати кнопки **повернути за годинниковою стрілкою**  чи **повернути проти**

годинникової стрілки  у верхній лівій частині вікна графічного редактора потрібну кількість разів.

3.8. Штамп. Цей інструмент  дозволяє копіювати фрагменти створеного малюнка. Треба:

- вибрати інструмент **штамп**  ;
- виділити фрагмент, який треба скопіювати (як при використанні інструмента ) ; скопійований фрагмент буде рухатися за вказівником миші;
- помістити вказівник у потрібне місце і відпустити ліву клавішу миші.

Щоб припинити штампування, треба вибрати інший інструмент.

3.9. Прямокутник. Еліпс. Спочатку розглянемо інструмент **прямокутник**  , який дозволяє малювати прямокутники потрібного розміру. При виборі цього інструменту під панеллю інструментів з'являється значки зафарбованого та не зафарбованого прямокутників:  . Колір прямокутника чи його межі вибирається на палітрі, як звичайно, і відображається у верхньому квадраті праворуч палітри.

Щоб намалювати прямокутник, потрібно:

- вибрати інструмент **прямокутник**  ;
- вибрати кнопку з зафарбованим чи ні прямокутником;
- вказівник поставити там, де має бути лівий верхній кут прямокутника;
- натиснути ліву клавішу миші і, утримуючи її, переміщати вказівник вниз і вправо, поки не одержимо прямокутник потрібного розміру;
- відпустити ліву клавішу миші.

Інструмент **еліпс**  використовується аналогічно.

Якщо при використанні інструментів **прямокутник** чи **еліпс** тримати натиснутою кнопку **Shift**, то буде намальований квадрат чи коло відповідно.

Серед кнопок виконання дій над фрагментами малюнка є ще наступні:

– очистити полотно від всіх зображень;

– імпортувати малюнок з бібліотеки спрайтів чи іншого файла для редагування;

– скасувати останню дію у графічному редакторі;

– відновлює останню скасовану попередньою кнопкою дію.

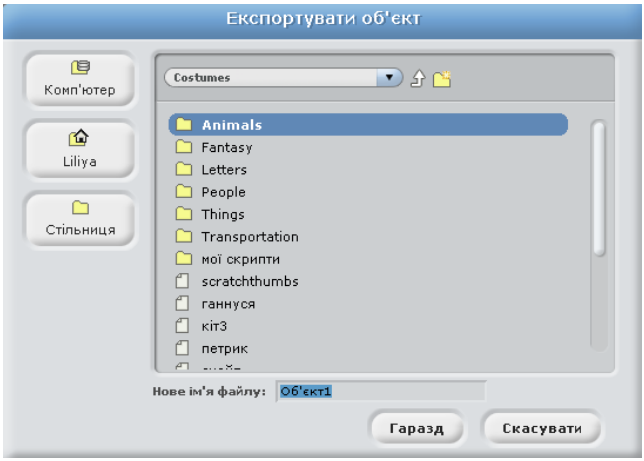
4. Створення та збереження нового спрайта

Щоб ***створити*** нового виконавця, треба запустити Scratch, вибрати кнопку  **створення нового об'єкта** під сценою. Відкриється графічний редактор Scratch. Описаними засобами створити нового виконавця.

Щоб ***зберегти*** намальований спрайт, треба натиснути кнопку **гаразд** в правій нижній частині вікна графічного редактора. Зображення спрайта появиться у списку виконавців під сценою у головному вікні Scratch.

Цього виконавця можна використовувати тільки в даному проекті. Щоб мати можливість використати намальований спрайт в інших проектах, його треба ***запам'ятати у бібліотеці*** спрайтів. Роблять це так:

1) вибрати виконавця в списку під сценою;
2) клацанням правої клавіші миші відкрити контекстне меню;
3) вибрати опцію **експортувати цей об'єкт**, відкриється вікно **експортувати об'єкт:**

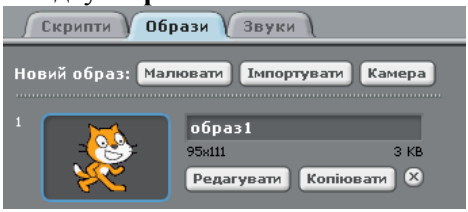


4) вибрати папку, де зберегти виконавця (часто це папка «Мої спрайти»);
5) у поле **нове ім'я файла** ввести ім'я і натиснути кнопку **Гаразд**.

5. Редагування. Створення нових образів виконавця

Часто створений малюнок спрайта згодом треба виправити, ***редагувати***. Для цього потрібно:

- вибрати спрайт в списку під сценою;
- відкрити закладку **Образи**:



- натиснути кнопку **Редагувати,** відкриється графічний редактор з зображенням спрайту;
- внести потрібні зміни;
- натиснути **Гаразд**.

Зауважимо, що таким способом можна редагувати не лише власних виконавців, але і будь-яких з бібліотеки.

Часто треба ***створити різні образи*** одного виконавця, які б тільки трошки відрізнялися один від одного. Це потрібно для створення анімації. Для цього:

- вибрати спрайт в списку під сценою;
- відкрити закладку **Образи**;
- зробити потрібну кількість копій спрайта, натискаючи кнопку **Копіювати**; утворяться однакові образи під різними іменами (імена можна змінювати);
- у кожному образі зробити відповідні зміни.

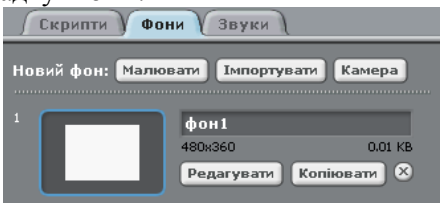
Образи мають свої номери, які співпадають з порядковим номером образу у списку.

6. Створення і збереження фону сцени

У графічному редакторі Scratch можна малювати не лише нових виконавців, але і нові декорації (фони) для сцени. Для цього використовують всі вище описані прийоми та інструменти малювання.

Щоб ***намалювати фон***, треба:

- вибрати ескіз сцени в правому нижньому куті головного вікна Scratch;
- вибрати вкладку **Фони**:



- натиснути кнопку **Малювати;**
- відкриється вікно графічного редактора, де створити малюнок фону;
- натиснути **Гаразд**, що збереже малюнок для використання у даному проекті.

Для використання створеного фону в інших проектах вибрати в контекстному меню опцію **експортувати цей об'єкт**, як це було для збереження спрайту. Фон можна редагувати так само, як і образ спрайта.

Приклад. Створити спрайт «Дівчинка». Голова – еліпс; очі, руки, ноги однакові (дублювати). Контур чорний. Зразок:

Практична робота №2

Створення об'єктів у графічному редакторі Scratch

1. Створення нових спрайтів за допомогою графічного редактора.
2. Створення кількох образів спрайта.
3. Створення декорацій.
4. Проект «Танцюрист».
5. Музичний супровід.
6. Листівка.

Завдання

1. Запустити Scratch. Вилучити Рудого кота, вибравши відповідну опцію **Вилучити** у контекстному меню спрайта.

2. Відкрити графічний редактор вибором кнопки створення нового об'єкта  . Намалювати нового виконавця – хлопчика Петрика, використовуючи інструменти: **олівець, пензель, еліпс, штамп, гумка, заливка, виділення фрагмента малюнка, дзеркальне відображення**. Очі, руки, ноги мають бути симетричні. Розфарбувати.

3. Зберегти малюнок спрайта, назвавши його Петриком. Зберегти цього виконавця в бібліотеці у папці «Мої спрайти» у файлі з іменем *«(Прізвище) Петрик».*

4. Збережи проект з іменем *«(Прізвище) Танцюрист».*

5. Створити ще три образи Петрика, де б він був зображений з боків і ззаду. Для цього:

a) відкрити проект *«(Прізвище) Танцюрист»*;
b) для виконавця Петрика відкрити вкладку **Образи**;
c) створити три копії виконавця Петрика;
d) вибрати першу копію виконавця, натиснути кнопку **Редагувати** і змінити вигляд, щоб утворився вигляд збоку

 ;

e) вибрати другу копію виконавця, зробити вигляд ззаду

 ;

 .

f) з третьої копії зробити вигляд з другого боку

6. Зберегти проект *«(Прізвище) Танцюрист».*

7. Намалювати фон сцени у проекті *«(Прізвище) Танцюрист»*. Для цього:
a) вибрати ескіз сцени у правому нижньому куті головного вікна Scratch;
b) перейти на вкладку **Фони**;
c) вибрати кнопку **Малювати**, відкриється графічний редактор;
d) створити малюнок, де би був зображений будинок, дерево, хмари, сонце, розмалювати;

e) зберегти фон у цьому проекті, зберегти малюнок у бібліотеці в файлі з іменем *«(Прізвище) Двір Петрика».*

8. Створити скрипт для Петрика, у якому він би танцював, змінюючи образи. Використати блок 

.

9. Додати музичний супровід, використавши блок 

з категорії **Звуки**. Спочатку перейти на закладку **Звуки**, де натиснути **Імпортувати.** Мелодію вибрати з бібліотеки. Ця мелодія відобразиться у списку в блоці 

Скрипт із звуком прикріпити до фону.

10. Намалювати листівку з анімацією *«(Прізвище)Листівка»* на задану тему. Є фон, рухомі об'єкти, музичний супровід, текст рухається (також є спрайтом).

Теми листівок:

1) З Новим роком!
2) З Різдвом Христовим!
3) З святом 8 березня!
4) З Великоднем!
5) З Днем Перемоги!
6) З Днем матері!
7) З Днем Конституції України!
8) З Днем незалежності України!
9) З Днем знань!
10) З Днем учителя!
11) З Днем захисника України!
12) З Днем народження!
13) З Днем ангела!
14) З одруженням!
15) З народженням донечки!
16) З народженням синочка!
17) З хрестинами!
18) З ювілеєм!
19) Із закінченням вузу!
20) З Днем сміху!
21) З Днем закоханих!
22) З Днем числа π!

Тема 3 . Анімація. Керування спрайтом. Взаємодія виконавців

1. Методи створення анімації.
2. Команди **говорити, думати, грати**.
3. Організація управління спрайтом.
4. Взаємодія між спрайтами.

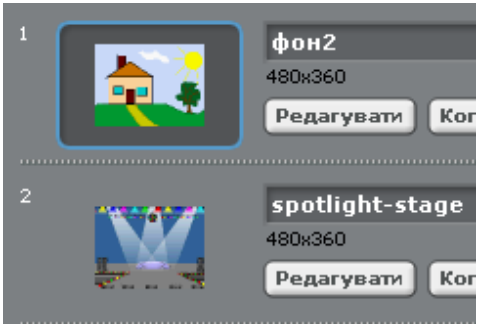
1. Методи створення анімації

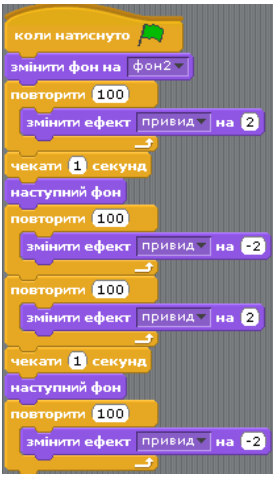
I. Анімація для спрайта шляхом зміни фону

Можна створити кілька фонів сцени. Далі для сцени скласти скрипт, у якому змінювати образи.

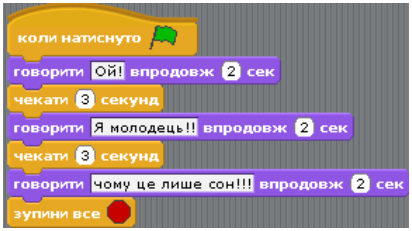
Приклад. *Сон.* Собака спить біля будинку, а йому сниться, що він на сцені.

Фони і скрипт для них:





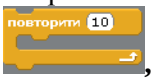
Скрипт для собаки:

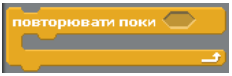


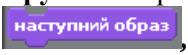
II. Анімація для спрайта шляхом використання багатьох образів

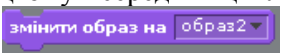
Для створення рухомого об'єкта в Scratch потрібно мати кілька образів цього об'єкта, які трошки відрізняються один від одного. Ці образи показані на вкладці **Образи** для кожного спрайта і пронумеровані: образ1, образ2 і т.д.

Анімація полягає у періодичній зміні цих образів за допомогою використання блоків 

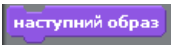
, 

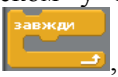
, 

та інших команд з категорії **Керувати**. При цьому всередині циклу слід використати блоки 

, 

з категорії **Вигляд.** Так поступали у лабораторній роботі 2.

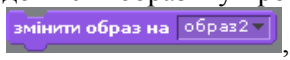
Блок 

змінює поточний образ спрайта на наступний за списком у вкладці **Образи**. Якщо цей блок помістити у цикл 

, то виникнуть всі образи згідно із зазначеним списком, а далі почнуть повторюватися спочатку. Якщо потрібно, щоб образ довше побув на сцені, то використовують блок 

.

Якщо треба змінити послідовність образів, то слід перейти на вкладку **Образи**, далі мишкою попересувати зображення образів у бажаному порядку.

Якщо треба змінити послідовність образів у процесі дій, то можна використати блок 

, де у віконечку вибрати потрібний образ із списку наявних образів.

III. Анімація для спрайта з одним образом

Анімацію можна створити і для спрайта з одним образом. Для цього можна в циклі здійснювати такі дії:

- переміщувати виконавця командами руху;
- змінювати розмір спрайта;

- підключити різні ефекти зміни спрайта;
- поєднувати перелічені прийоми.

Команди переміщення виконавця ми вже вивчали.

Для ***зміни розміру*** у проекті є такі блоки з категорії **Вигляд**:

a) блоком 

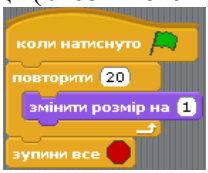
можна збільшити (більше 100%) чи зменшити (менше 100%) спрайт;

b) блок 

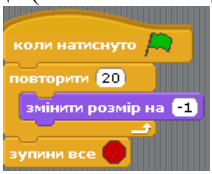
дозволяє організувати плавне зменшення спрайта (коли число у віконечку від'ємне) чи його збільшення (коли число додатне), якщо використовувати його в циклі.

Приклад 1. *Зміна розміру виконавця.*

Збільшення виконавця (ілюзія його наближення):



Зменшення виконавця (ілюзія його віддалення):



Різні ефекти можна встановити вибором таких блоків з категорії **Вигляд**:

a) 

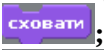
;

b) 

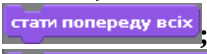
;

c) 

;

d) 

;

e) 

;

f) 

.

У блоках **a)** і **b)** у першому віконечку можна вибрати такі ефекти:

- **колір** (вибрати з палітри);
- **вздуття**;
- **обертання** (скрут);
- **пікселями** (розділення на квадрати за кольорами);
- **мозаїка** (спрайти множаться, зображення розміщуються як елементи матриці);
- **яскравість**;
- **привид** (зображення стає прозорішим).

У другому віконечку цих двох блоків пишуть число, яке вказує ступінь вираження ефекту. Описані ефекти можна усунути блоком 

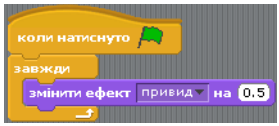
.

Блоки **c)** і **d)** приводять до появи і зникнення спрайта. Їх використовують, якщо потрібна миттєва поява або зникнення виконавця. Для поступової появи або зникнення використовують ефект **привид.**

Блоки **e)** і **f)** організовують першочерговість зображень спрайтів у випадку, якщо вони накладаються.

Приклад 2. *Зміна ефектів.*

Поступове зникнення спрайта:



Здуття спрайта зі зміною кольору:

2. Команди говорити, думати, грати

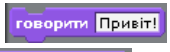
При створенні проектів з анімацією часто потрібно, щоб спрайти говорили, висловлювали свої думки, видавали звуки тощо. У Scratch для цього є наступні можливості.

1. Команда 

з категорії **Вигляд.** У першому віконечку можна ввести будь-яку фразу, яку має говорити спрайт. Ця фраза виникне поряд із спрайтом після запуску проекта. Вона не озвучується. У другому віконечку треба ввести, скільки секунд фраза має висвічуватися. Після закінчення цього часу фраза зникне.

2. Команда 

з категорії **Вигляд**. Тут також потрібно вводити репліку для спрайта, лише ця фраза буде висвічуватися до кінця виконання програми. Можна перервати її виконання, якщо після неї записати блоки 

; 

з порожнім віконечком.

3. Команди 

та 

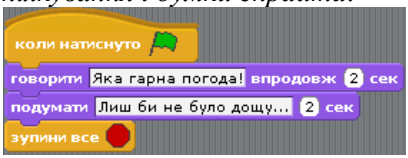
з категорії **Вигляд** діють так само, як попередні, лише записані фрази виникають біля спрайта у вигляді думок.

4. Команди 

, 

з категорії **Звук.** Дозволяють вибрати наявний звуковий уривок у списку на вкладці **Звуки** і озвучити його. Відрізняються ці дві команди тим, що у першій команді звучання музики відбувається одночасно з наступними командами в скрипті. У другій же спочатку грає музика до завершення уривка, а потім виконується наступний блок скрипта. Можна одночасно грати іншу мелодію або складати скрипти для різних музичних інструментів по нотах. Різні музичні уривки можна вибирати з наявної бібліотеки (вибрати **Імпортувати** на вкладці **Звуки**) або записувати самим (якщо є мікрофон).

Приклад 3. *Спілкування і думки спрайта.*



3. Організація управління спрайтом

Метод І. Керування спрайтом включає насамперед керування початком його дій. Тому спочатку розглянемо ***блоки відстежування моменту настання події.*** Вони містяться у категорії **Керувати** і мають форму прямокутника з опуклою верхньою стороною. Це такі блоки:

1) 

;

2) 

;

3) 

;

4) 

.

При виборі блоку 1) приєднаний до нього скрипт починає працювати, якщо натиснути кнопку з зеленим прапорцем над сценою.

При виборі блоку 2) дії відбуваються після того, як ми натиснемо відповідну кнопку на клавіатурі.

Блок 3) розпочинає дії, якщо клацнути лівою клавішею миші по зображенню того спрайта на сцені, імя якого вказано у блоці. Блок реалізує взаємодію з спрайтом в реальному часі.

Блок 4) починає роботу при отриманні повідомлення, про що буде далі.

Метод ІІ. Керувати рухом спрайта в реальному часі (здійснювати інтерактивний зв'язок) можна за допомогою мишки або клавіш на клавіатурі.

Організувати ***керування спрайта мишкою*** можна так:

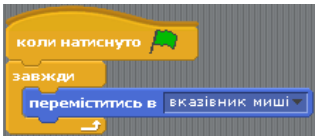
1) вибрати потрібний блок початку дії;
2) вибрати цикл 

;
3) помістити в цикл блок 

з категорії **Рух**;
4) вибрати у віконечку **вказівник миші.**

У результаті спрайт буде рухатися по сцені за вказівником миші.

Приклад 4. *Керування мишкою.*



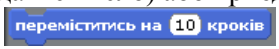
Організувати ***керування спрайтом клавішами із стрілками*** для вибору різних напрямків руху можна так.

Розглянемо скрипт:

1) вибрати блок початку дій 

(вибрати кнопку зі стрілкою вправо);
2) приєднати блок 

(змінити x на 1 або інше додатне число) або приєднати блоки 

та 

(переміститись на 1 або інше додатне число кроків).

Цей скрипт рухає виконавця *вправо* на вказану кількість кроків, якщо натиснути кнопку зі стрілкою вправо.

Рух вліво реалізує такий алгоритм:

1) вибрати блок початку дій 

(вибрати кнопку зі стрілкою вліво);
2) приєднати блок 

(змінити x на -1 або інше від'ємне число) або приєднати блоки 

(повернути вліво в напрямку -90) та 

(переміститись на 1 або інше додатне число кроків).

Рух вверх:

1) вибрати блок початку дій 

(вибрати кнопку зі стрілкою вверх);
2) приєднати блок 

(змінити y на 1 або інше додатне число) або приєднати блоки 

(повернути вгору в напрямку 0)

та 

(переміститись на 1 крок або інше додатне число кроків).

Рух вниз:

1) вибрати блок початку дій 

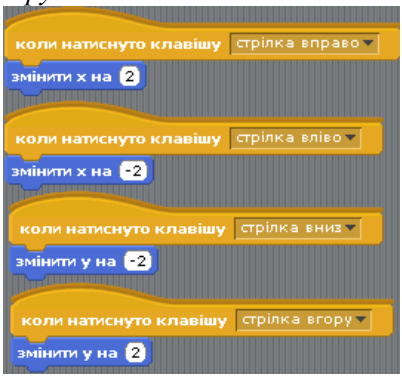
(вибрати кнопку зі стрілкою вниз);
2) приєднати блок 

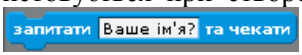
(змінити у на -1 або інше від'ємне число) або приєднати блоки 

(повернути вниз в напрямку 180) та 

(переміститись на 1 крок або інше додатне число).

Приклад 5. *Керування клавішами*



Метод III. Дії спрайта можуть залежати від нашої ***відповіді на поставлене запитання.*** Цей прийом використовується при створенні тестів. Потрібно використати блок 

з категорії **Датчики**. У віконечку сформулювати питання. Тоді при виконанні виникне віконечко, де потрібно надрукувати відповідь, і після цього натиснути **Enter** на клавіатурі. Цю відповідь можна використати далі, вибравши блок 

з категорії **Датчики.**

Якщо у віконечку перед блоком 

клацнути мишкою, то появиться галочка, а на сцені у верхньому лівому кутку відобразиться наша відповідь.

Приклад 6. *Сміх і сльози.* Котик сміється чи плаче за нашим бажанням.

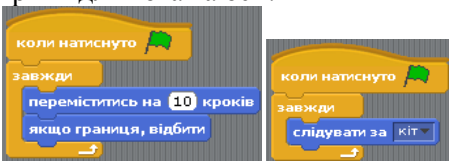
4. *Взаємодія між спрайтами*

При написанні проекту у Scratch потрібно, щоб виконавці взаємодіяли між собою: спілкувалися, реагували один на одного, узгоджували свої дії. Далі розглянемо деякі види взаємодії.

❖ ***Слідкування за рухом іншого спрайта*** можна організувати, поклавши в циклі блок 

з категорії **Рух**. У віконечку цього блоку слід вибрати ім'я того спрайта зі списку наявних у проекті, за яким має слідкувати даний спрайт. Якщо для даного спрайта натиснута кнопка руху на всі сторони, то він буде обертатися так, щоб той, за ким він слідкує, знаходився завжди в полі зору. Якщо ж натиснута кнопка руху лише вліво-вправо, то спрайт не крутиться, тільки повертається вліво чи вправо.

Приклад 7. *Слідкування.* Кіт бігає вліво-вправо, а оса слідкує за ним. Скрипт для кота та оси:



❖ ***Реагування на дотик до іншого об'єкта або до якогось кольору*** здійснюється за допомогою таких блоків з категорії **Датчики**:

Ці блоки мають форму шестикутника і не приєднуються до вивчених раніше блоків. Датчики використовуються, щоб вставляти їх у віконечка інших блоків, причому віконечка мають бути шестикутної форми. Такі віконечка мають деякі команди циклу і розгалуження з категорії **Керувати** та логічні оператори з категорії **Оператори.**

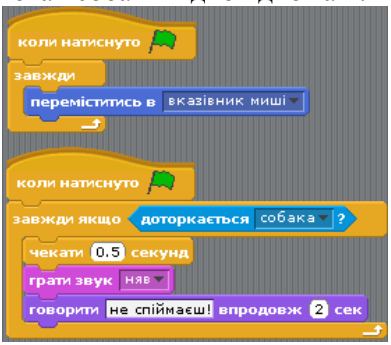
У датчику 

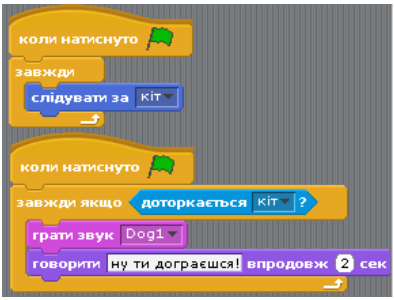
треба у віконці вибрати або вказівник миші, або границю, або ім'я спрайта, до якого може доторкатися даний спрайт. Датчик 

перевіряє,чи даний спрайт доторкається до чого-небудь, що має заданий колір. Датчик 

має у віконечках зразки двох кольорів і перевіряє, чи об'єкт першого кольору доторкається до чогось, що має другий колір.

Приклад 8. *Реагування на дотик* (*Кіт і собака*). Кіт керується мишкою, собака слідкує за котом. Якщо кіт торкається до собаки, то останній гавкає і каже: «ну ти дограєшся!», а кіт нявкає і каже: «не догониш!». Скрипти для кота і собаки відповідно такі:

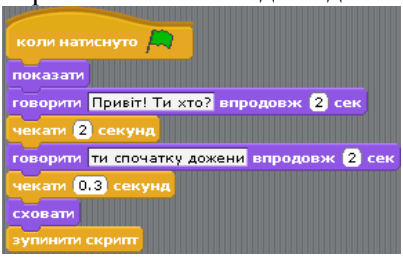


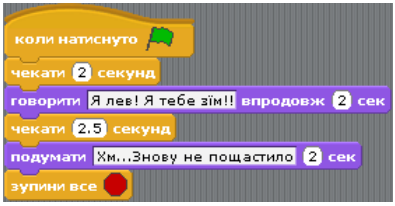


❖ ***Узгодження дій спрайтів за часом.*** Злагоджені дії виконавців можна запрограмувати, точно підраховуючи тривалість кожної дії. Наприклад, при діалозі перший задає питання і чекає потрібну кількість часу, необхідну для того, щоб другий встиг відповісти. Для цього використовується блок

з категорії **Керувати**. Як правило, час очікування вибирають експериментально.

Приклад 9. *Узгодження за часом.*. Діалог кота і лева. Скрипт кота і лева відповідно:





❖ ***Узгодження дій спрайтів по одержанню повідомлення.*** Цей метод полягає в тому, що один спрайт виконує певний набір команд. Після цього передає повідомлення іншому, а той після отримання повідомлення виконує свої дії.

Передача повідомлення відбувається за допомогою блоків 

або 

з категорії **Керувати**. У віконечках цих блоків треба записати повідомлення для передачі іншим.

При виборі 

передається повідомлення і продовжується виконання наступних блоків. При виборі 

спрайт передає повідомлення і очікує виконання скрипту для виконавця, який зреагував на повідомлення. Після цього перший продовжує виконання команд.

Для іншого спрайту, який має зреагувати на виконання дій першим спрайтом, скрипт із наступними діями треба розпочати блоком 

з категорії **Керувати**. У віконечку цього блоку треба записати те саме повідомлення, яке передав перший.

Приклад 10. *Узгодження за повідомленням.* Кіт шукає і кличе жабку. Коли вона відгукується, то кіт радіє, що її знайшов. Скрипт кота:

Скрипт жабки:

Практична робота № 3

Організація взаємодії між спрайтами

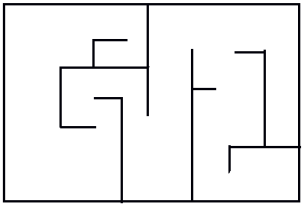
1. Використання елементів анімації.
2. Організація діалогів.
3. Управління спрайтами.
4. Організація взаємодії між спрайтами.

Завдання

1. Створити проект ***«(Прізвище) Розмова»*** з анімацією і діалогом двох виконавців, які змінюють образи і мають свої думки. Взаємодія спрайтів за допомогою часу. Наприклад, розмова учня з вчителем на іспиті, розмова хлопця з дівчиною, торгівля на базарі тощо.
2. Створити проект ***«(Прізвище) Лабіринт»***, у якому виконавець має управлятися стрілочками вверх, вниз, вліво, вправо. На фоні зображено лабіринт(приклад див. на мал.).

Виконавець починає рухатися з верхнього лівого кута сцени. У нижньому правому куті є приз. Якщо виконавець доторкнеться до стінок лабіринту, то зникне, і треба сповістити про поразку. Якщо ж він добереться до призу, то виграє. На честь переможця хай грає музика.

3. Створити проект ***«(Прізвище) Пастух»***. Тварина пасеться на зеленій траві, а пастух за нею спостерігає, пісеньку співає. Якщо тварина заходить у жовту пшеницю, то пастух кричить на неї і виганяє. Коли він доторкнеться до тварини, вона назад перейде на траву. Тварина управляється мишкою. Узгодження дій за повідомленням.



Практична робота № 4

Створення мультфільму на основі казки зі звуковими ефектами та діалогами

1. Використання графічного редактора для створення образів та фонів.
2. Використання елементів анімації.
3. Організація діалогів та звукових ефектів.
4. Управління спрайтами та організація взаємодії між ними.
5. Навички оцінювання проекту.

Завдання

Виконати проект ***«(Прізвище)(Назва казки)»***, у якому створити мультфільм на основі вибраної казки. Вимоги:

- фон із назвою казки;
- сценарій відповідає казці;
- кілька виконавців з різними образами;

- різні фони;
- відповідні діалоги та дії виконавців, звукові ефекти;
- слід передбачати положення виконавців, їх появу і зникнення, розраховувати час, керувати спрайтами за допомогою повідомлень.

Рекомендований список казок:

1. Попелюшка.
2. Теремок.
3. Три ведмеді.
4. Ріпка.
5. Лев і мишеня.
6. Лисичка та журавель.
7. Білосніжка.
8. Курочка Ряба.
9. Цап та баран.
10. Снігуронька.
11. Троє поросят.
12. Колобок.
13. Червона шапочка.
14. Рукавичка.
15. Коза-дереза.
16. Котигорошко.
17. Пан Коцький.
18. Їжак та заєць.
19. Котик і півник.
20. Колосок.
21. Принцеса на горошині.
22. Кресало.

Тема 4 . Організація циклів та розгалужень у Scratch

1. Команди циклу.
2. Команди розгалуження.
3. Вкладені цикли.
4. Вкладені розгалуження.
5. Вкладені цикли та розгалуження.

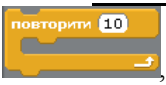
1. Команди циклу

Фрагмент алгоритму, у якому одна або кілька команд можуть виконуватися більше ніж один раз, називається **циклом.** Алгоритм, який містить цикл, називається **алгоритмом з циклом** або **алгоритмом з повторенням**.

У середовищі Scratch алгоритми з циклами можна реалізувати за допомогою кількох команд. Всі вони знаходяться у категорії **Керувати.**

1.1. Цикл без лічильника реалізується командою циклу 

. Команди, що є всередині циклу (в тілі циклу), повторюються нескінченну кількість разів. Зупинити цей процес можна, натиснувши червоний кружечок  над сценою.

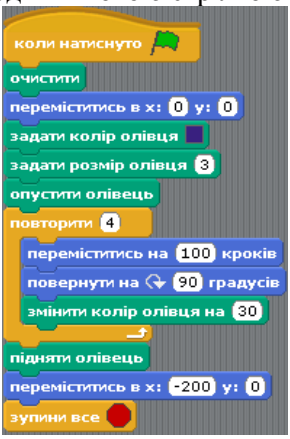
1.2. Цикл з лічильником реалізується командою 

, де у віконечку слід написати кількість повторень команд, що містяться в тілі циклу.

Далі зображена блок-схема алгоритму з циклом з лічильником. У задачі треба наповнити діжку 5 відрами води з колодязя.

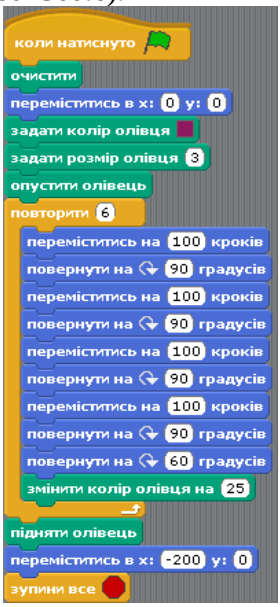
Приклад 1. *Квадрат.* Намалювати квадрат із стороною 100 можна, повторивши 4 рази команди:

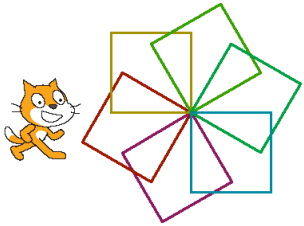
- переміститись на 100 кроків;
- повернути за годинниковою стрілкою на 90 градусів.



Приклад 2. *Орнамент.* Намалювати орнамент з шести квадратів із стороною 80 можна, повторивши 6 разів команди:

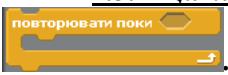
- переміститись на 80 кроків;
- повернути за годинниковою стрілкою на 90 градусів.
- переміститись на 80 кроків;
- повернути за годинниковою стрілкою на 90 градусів.
- переміститись на 80 кроків;
- повернути за годинниковою стрілкою на 90 градусів.
- переміститись на 80 кроків;
- повернути за годинниковою стрілкою на 90 градусів.
- повернути за годинниковою стрілкою на 60 градусів (60=360:6).





Приклад 3. *Коло.* Намалювати коло, повторивши 120 разів команди:

- переміститись на 4 кроки;
- повернути за годинниковою стрілкою на 3 градуси.

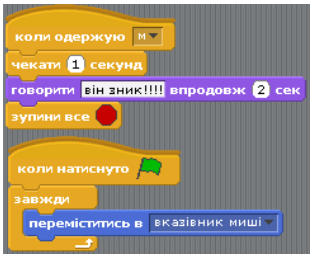
1.3. Цикл з післяумовою реалізується командою 

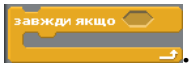
. У віконечку шестикутної форми слід записати умову припинення повторення. Тобто команди у тілі циклу будуть повторюватися, поки умова не стане істинною. Відповідні команди шестикутної форми розміщені у категорії **Датчики**. Тіло такого циклу виконається принаймні один раз. Елемент блок-схеми:

Приклад 4. *Кіт ловить м'яч.* М'яч літає по сцені, поки не доторкнеться до кота, а тоді зникає. Кіт керується мишкою. Коли м'яч зникне, кіт каже: «Він зник!»

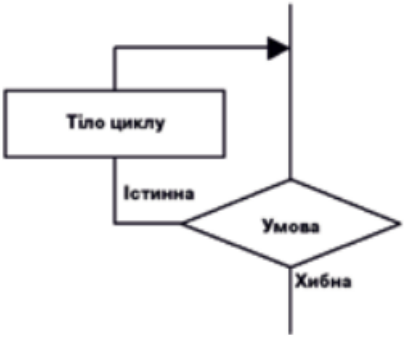
Скрипт для м'яча використовує цикл з післяумовою:

Скрипт для кота:



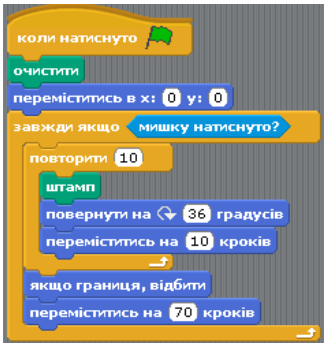
1.4. Цикл з передумовою 

. У віконечку шестикутної форми слід записати умову продовження повторень. Тобто команди у тілі циклу будуть повторюватися, поки умова залишатиметься істинною. Виконання команд припиниться, якщо умова стане хибною. Елемент блок-схеми:



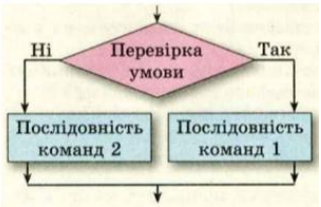
Приклад 5. *Коло малює.* Фіолетове коло утворює квітки з 10 кіл. Квітки з'єднуються в узор. Це відбувається тоді, коли натиснута ліва клавіша миші.

Скрипт і приклад узору:



2. Команди розгалуження

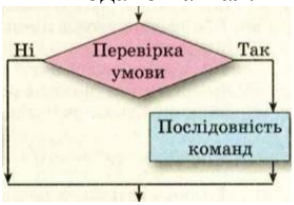
В алгоритмі може бути фрагмент, який містить команду перевірки умови, і залежно від результату виконання цієї умови (**так** чи **ні**) буде виконуватись або одна послідовність команд, або інша. Такий фрагмент в алгоритмі називають **повним розгалуженням**. Фрагмент алгоритму з повним розгалуженням подано на мал.



Виконання повного розгалуження відбувається так: виконавець виконує **команду перевірки умови**; якщо результат виконання цієї команди **так**, то виконується **послідовність команд 1**, після чого переходить до виконання першої команди наступного фрагмента; якщо результат виконання команди перевірки умови **ні**, то виконується **послідовність команд 2,** після чого також переходить до виконання першої команди наступного фрагмента.

В алгоритмах використовується і **неповне розгалуження**. Воно відрізняється від повного тим, що при отриманні результату **ні** при перевірці умови послідовності команд 2 немає, а виконавець зразу переходить до виконання

першої команди наступного фрагмента. Фрагмент алгоритму з неповним розгалуженням подано на мал.

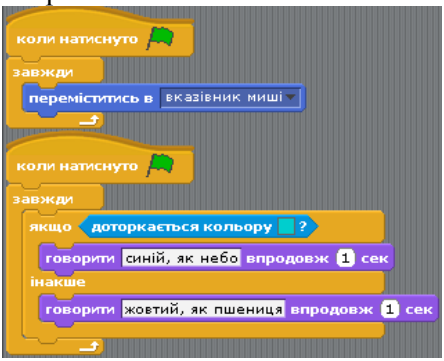


Алгоритм, який містить розгалуження, називається **алгоритмом з розгалуженням.**

2.1. Повне розгалуження в Scratch реалізується блоком

з категорії **Керувати**. У шестикутному віконечку треба поставити команду перевірки умови з категорії **Датчики,** а далі записувати послідовність команд 1 у випадку, коли умова виконується, та послідовність команд 2, якщо умова не виконується.

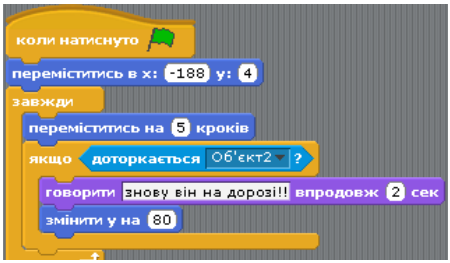
Приклад 6. *Прапор.* Кіт керується мишею і розказує про український прапор.



2.2. Неповне розгалуження реалізується блоком

з категорії **Керувати**. У шестикутному віконечку також кладуть команду перевірки умови, а далі записують команди, які слід виконати у випадку **так.**

Приклад 7. *Кіт обходить перешкоду.* Кіт рухається по сцені. Якщо доторкнеться до ноутбука, то обійде його.

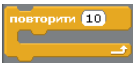


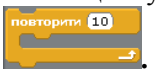
3. Вкладені цикли

Вкладений цикл – це такий фрагмент алгоритму з циклом, у тілі циклу якого є інші цикли. Цикл, який міститься в тілі іншого циклу, називається **внутрішнім**. А цикл, у тілі якого розміщений інший цикл, називається **зовнішнім**.

Під час виконання вкладених циклів спочатку починається виконання зовнішнього циклу. У ході його виконання, коли настає черга виконання внутрішнього циклу, то цей внутрішній цикл виконується повністю, після чого продовжується виконання зовнішнього циклу. І так відбувається під час кожного виконання тіла зовнішнього циклу.

В Scratch можна реалізувати такі вкладені цикли.

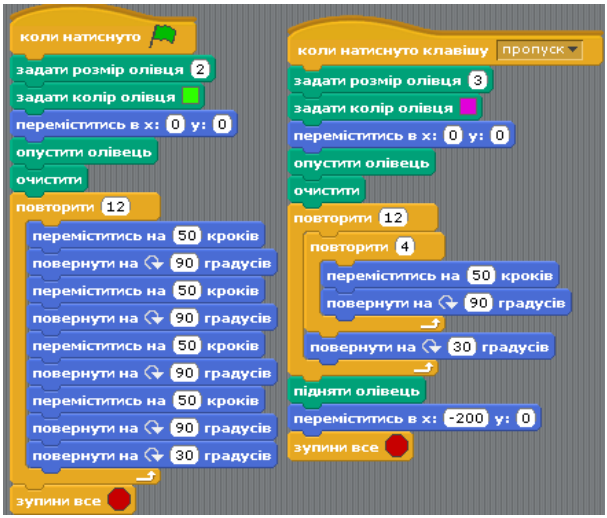
3.1. Вкладені цикли з лічильником. В тілі циклу з лічильником 

використати цикл 

.

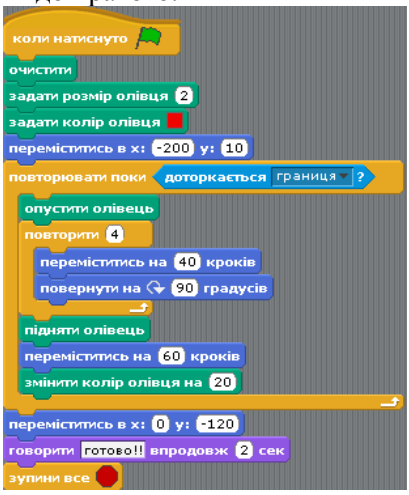
Приклад 8. *Вкладені цикли з лічильником.* Намалювати орнамент з 12 квадратів можна двома способами. В циклі (12 разів) намалювати 4 сторони, повертаючи на 90 градусів (див. перший скрипт), або у внутрішньому циклі(4 рази) намалювати одну сторону і один поворот (див. другий скрипт). В кінці тіла зовнішнього циклу повернути на 360:12=30 градусів.

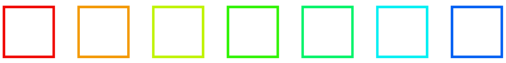
Результат виконання програми:



3.2. Цикли з передумовою у вкладених циклах. У вкладених циклах можна використовувати не тільки цикли з лічильником, але і цикли з передумовою. Вони можуть бути як зовнішніми, так і внутрішніми циклами.

Приклад 9. *Вкладені цикли з передумовою.* Побудувати квадрати різного кольору зі стороною 50 через 10 кроків від лівого кінця сцени до правого.

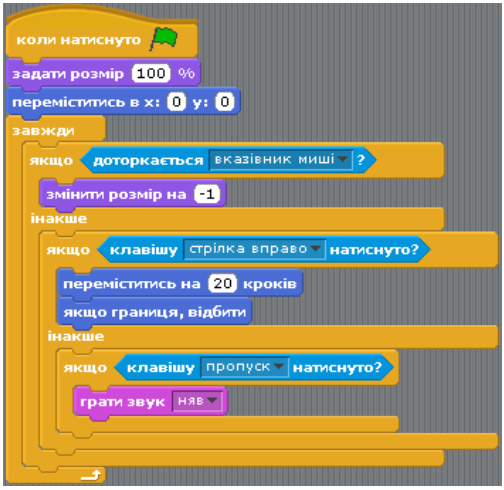




4. Вкладені розгалуження

Вкладені розгалуження – це фрагмент алгоритму, у якому одне розгалуження міститься всередині іншого розгалуження. І зовнішні, і внутрішні розгалуження можуть бути як повними, так і неповними.

Приклад 10. *Вкладені розгалуження.* Скласти проект,у якому виконавець зменшить свій образ на 1, якщо до нього доторкнеться вказівник миші; переміститься вправо на 10, якщо натиснуто стрілку вправо; скаже «няв», якщо натиснуто клавішу пропуск.



Приклад 11. *Піаніно.* Клавіші від **я** до **ь** у нижньому рядку клавіатури відповідають нотам від «до» до «сі». Коли натискаємо вказану клавішу, то звучить відповідна нота, а кіт називає назву ноти.

5. *Вкладені цикли та розгалуження*

5.1. Розгалуження, вкладене в цикл. Кожен раз при виконанні тіла циклу виконується команда розгалуження. Такий прийом є у прикладі 6 і 7.

5.2. Цикл, вкладений у розгалуження.

Приклад 13. *Вибір фігури.* Якщо натиснути цифру 3, то малювати трикутник; якщо цифру 4, то квадрат; якщо 5 – п’ятикутник, а 6 – шестикутник. Тут є кілька вкладень.

Практична робота № 5

Використання циклів та розгалужень у проектах

1. Використання циклів та вкладених циклів.
2. Використання розгалужень та вкладених розгалужень.
3. Використання вкладених циклів та розгалужень.

Завдання

1. Створити проект ***«(Прізвище) Квадрати»*** для малювання квадратів за зразком. Використати вкладені цикли.

2. Створити проект ***«(Прізвище) Олімпійські кільця»*** для малювання кілець на прапорі олімпійських ігор. Використати вкладені цикли.
3. Створити проект ***«(Прізвище) Орнамент»***, у якому виконавець (пелюстка квітки) малює узор з квіток різного кольору. Використати вкладені цикли. Доповнити орнамент іншими деталями.
4. Створити проект ***«(Прізвище) Лінії»***, у якому виконавець намалює синю чи зелену горизонтальну лінію довжиною 40 кроків, якщо натиснуто клавіші «стрілка вліво» чи «стрілка вправо» відповідно. Якщо ж натиснуто клавіші «стрілка вверх» чи «стрілка вниз», то треба намалювати відповідно червону чи жовту вертикальну лінію довжиною 20 кроків. При цьому виконавець має говорити про напрям свого руху. Використовувати вкладені розгалуження.
5. Створити проект ***«(Прізвище) Легкоатлет і художник»*** для двох виконавців. Якщо вказівник миші наведено на першого виконавця, то він бігає по кругу і щось говорить, а якщо на другого, то той малює фігуру і грає музика. Якщо ж відстань між виконавцями менша 100 кроків, то вони в першу чергу здороваються між собою. Використати вкладені цикли та розгалуження.

Тема 5 . Змінні. Вирази. Списки

1. Види та організація даних.
2. Створення змінних, команди присвоювання та зміни значень змінних.
3. Ввід і вивід символьних даних.
4. Вирази, операції та функції.
5. Застосування змінних і виразів для побудови графіків функцій.
6. Застосування Scratch при розв'язанні математичних задач.
7. Списки.

1. Види та організація даних

Види даних у Scratch:

1.Символьні дані (числа і тексти). Їх можна використовувати в багатьох блоках як сталі величини, змінні величини чи елементи списків. Значення можна знаходити (обчислювати) за допомогою виразів і функцій.

2.Графічні дані (спрайти, фони сцени і малюнки, створені за допомогою команд категорії **Олівець**). Спрайти та фони імпортуються або створюються за допомогою графічного редактора.

3.Аудіодані – це звуки, створені блоками категорії **Звуки**, або імпортовані чи записані самостійно звукові файли.

З точки зору організації даних виділяють:

1. **Скалярні дані** (сталі і змінні величини).
2. **Структурні дані** (таблиці, масиви, списки тощо).

Значення **сталих (констант)** задаються безпосередньо в програмі. Це числа і тексти, розміщені у віконечках використаних блоків.

Змінні величини представляються в програмі за допомогою імен, їх значень в програмі не видно. Як правило, значення змінних з'являються в ході виконання програми.

Таблиці, масиви, списки – це набори даних певної структури. В Scratch використовуються лише **списки** – частинний випадок масивів.

2. Створення змінних, команди присвоювання та зміни значень змінних

Змінна – це комірка пам'яті, в яку програма може записувати значення (числові чи текстові) і використовувати (зчитувати) їх пізніше для утворення нових значень.

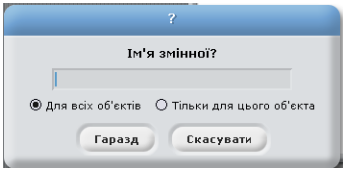
В Scratch змінні можна створювати і використовувати за допомогою блоків категорії **Змінні**. Тут є два блоки: 

та 

.

Якщо вибрати 

, то появиться вікно створення змінної:



У ньому потрібно вказати ім'я змінної, яке використовується в командах для посилання на її поточне значення. Крім того, при створенні задається, чи є змінна доступною для всіх об'єктів (*глобальна змінна*), чи тільки для одного конкретного об'єкту (*локальна змінна*). Далі натиснути **Гаразд**. Після створення змінної появляються блоки для роботи з цією змінною.

Якщо введеної змінної вже не потрібно, її можна вилучити, вибравши відповідну кнопку 

.

Для присвоювання значень змінній використовують блок 

.Тут змінній **а** присвоюється значення 0. Змінній може присвоюватись і текстове значення.

Значення змінної можна відобразити на сцені. Це можна зробити двома способами:

1) клацнувши мишкою на віконечко зліва біля блоку з назвою змінної в категорії **Змінні**, при цьому у віконечку з'явиться галочка 

.
2) використавши блоки 

чи 

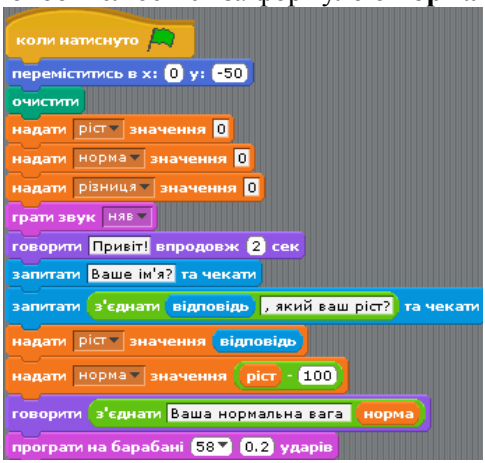
з категорії **Змінні**, щоб назва та значення змінної появлялось чи зникало на сцені.

Для зміни значень використовують блок 

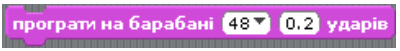
. У результаті до попереднього значення змінної **а** додається число 1. У віконечку можна записати будь-яке число. Якщо воно від'ємне, то від попереднього значення змінної віднімається модуль цього числа. Значення змінної можна змінювати і вручну за допомогою слайдера (бігунця) 

. Він виникає, якщо навести вказівник миші на віконечко на сцені і двічі клацнути лівою клавішею.

Приклад 1. *Тест про вагу.* У проекті визначається нормальна вага людини залежно від її росту. Створено дві змінні **ріст** та **норма**. Значення першої вводить користувач, значення другої обчислюється за формулою **норма=ріст-100**.

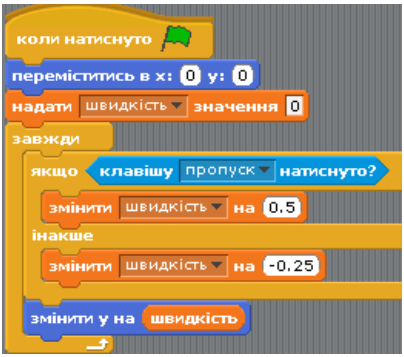


У прикладі символьними даними є числа, слова у віконечках, що використовуються як слова і змінні. Графічні дані – це спрайт. Звукові дані – це звуки, створені командами 

та 

.

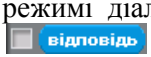
Приклад 2. *Прискорений рух.* Кіт рухається вверх, якщо натиснуто пропуск, і падає вниз у іншому випадку. Рух вверх і вниз відбувається з різною швидкістю. Використовується змінна **швидкість**.



Цей скрипт використовується при створенні ігор на рух спрайта.

3. Ввід і вивід символьних даних

В Scratch ***ввід даних*** можна здійснювати двома способами:

1) за допомогою блоку  з категорії **Датчики**, який дає можливість вводити числа і тексти в режимі діалогу; значення введеного міститься у блоці  з віконечком для відображення на сцені;
2) за допомогою віконечок значень змінних (моніторів даних) з бігунцем (тільки для чисел).

Перший спосіб кращий, бо у другому можна вибрати лише цілі значення.

В Scratch ***вивід даних*** можна здійснювати двома способами:

1) за допомогою блоку  чи  з категорії **Вигляд**, який дає можливість виводити на екран тексти;
2) використовуючи віконечко із значенням змінної на сцені.

Перший спосіб кращий, бо у другому значення заокруглюються до першого знаку після коми.

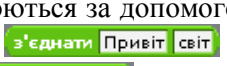
Приклад 3. *Вік.* Обчислити вік людини по року народження. Змінна **рік** є роком народження і вводиться з клавіатури, а змінна **вік** обчислюється за формулою **2020-рік.**

4. Вирази, операції і функції

За допомогою виразу можна задати правило обчислення значення деякої величини.

Вираз складається з ***операндів*** та ***операцій***. Операндами можуть бути сталі, змінні, функції, елементи списків. Операції в Scratch вибирають в категорії **Оператори**.

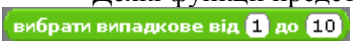
В залежності від операцій вирази можна поділити на такі групи:

1) числові вирази, що містять операції додавання  , віднімання  , множення  , ділення  ;
2) текстові вирази, що утворюються за допомогою операції з'єднання двох слів в одне  та операції виділення букви з слова  ;
3) порівняння з використанням операцій  ;
4) логічні вирази:  .

У виразах можна використовувати різні функції. Їх вибирають з блоку  . Клацнувши на чорний трикутник, одержимо список функцій:

абс (абсолютна величина),
корінь (квадратний),
sin, cos, tan, asin, acos, atan(тригонометричні функції)**,**
ln, log (логарифмічні функції)**,**
e^, 10^ (показникові функції)**.**

Деякі функції представлені окремими блоками:



- генератор випадкових чисел, який вибирає випадково число між двома заданими;

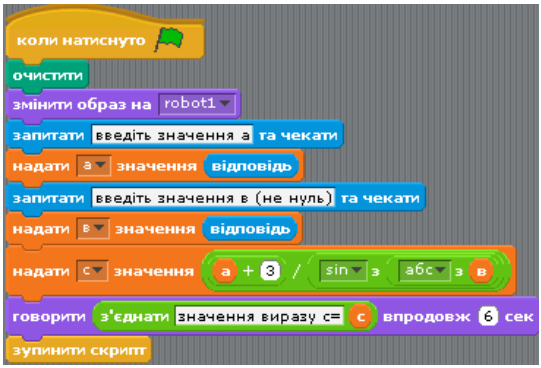
- остача від ділення першого числа на друге;

- округлює число до цілого.

Зауваження. У виразах не використовуються дужки.

Приклад 4. *Степінь числа.* Піднести число до натурального степеня можна, використовуючи цикл.

Приклад 5. *Вираз.* Знайти значення виразу $c = \frac{a+3}{\sin|b|}$ для заданих значень a і b.



Приклад 6. *Округлення.* Округлити введене число до третього знака після коми.

Щоб округлити число ***a*** до третього знака після коми, слід помножити його на $10^3 = 1000$, округлити цей результат до цілих, а потому поділити на 1000. Тобто створити вираз

. Для округлення до *n*-го знаку після коми множать на 10^n.

5. Застосування змінних і виразів для побудови графіків функцій

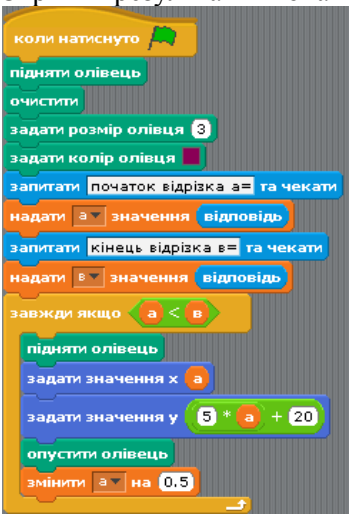
Щоб побудувати графік функції, потрібно спочатку задати межі *a* та *b* зміни аргумента. Далі в циклі будувати точку з координатами *x* та *y*, причому *x* змінюється від *a* до *b* з певним кроком, а *y* обчислюється як значення виразу. Крок та масштаб підбирають «вручну», щоб графік був наочний.

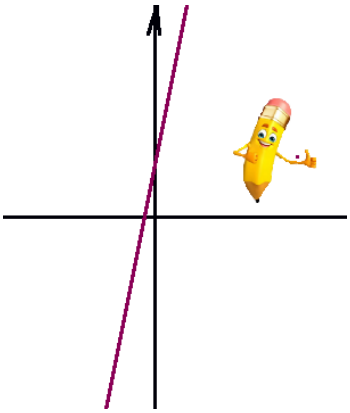
Осі координат можна намалювати по координатах або зобразити як фон сцени (у цьому випадку слід перевірити, чи вони розміщені посередині сцени).

Приклад 7. *Випадкова точка.* Побудувати 20 точок на площині з випадковими координатами.

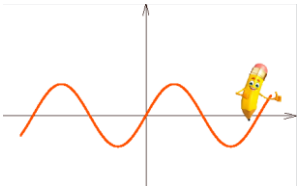
Приклад 8. *Графік функції.* а)Побудувати графік функції $y = 5x + 20$. Запитувати межі зміни аргумента. Крок задавати у програмі (0.5 або 1).

Скрипт і результат виконання:





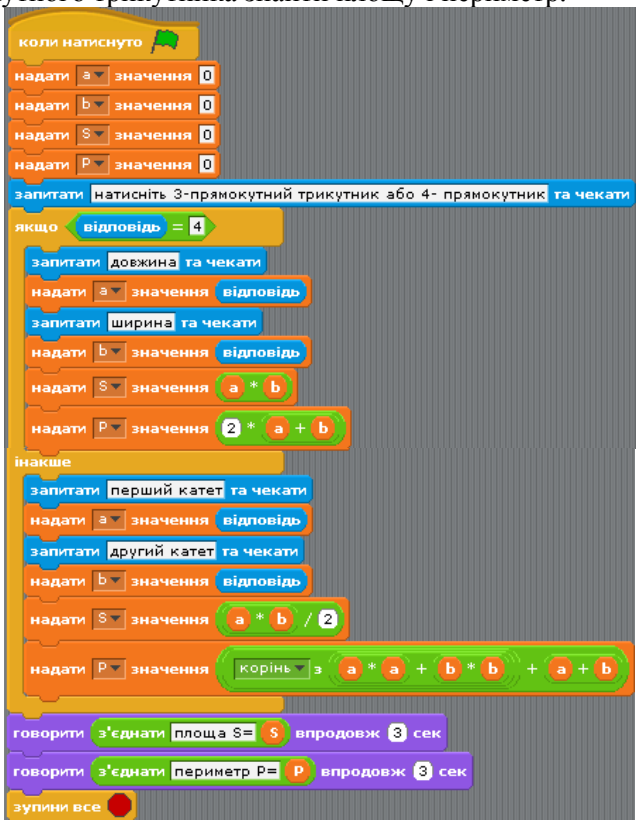
б) Побудувати графік функції $y = 50\sin 2x$. Результат:



6. *Застосування Scratch при розв'язанні математичних задач*

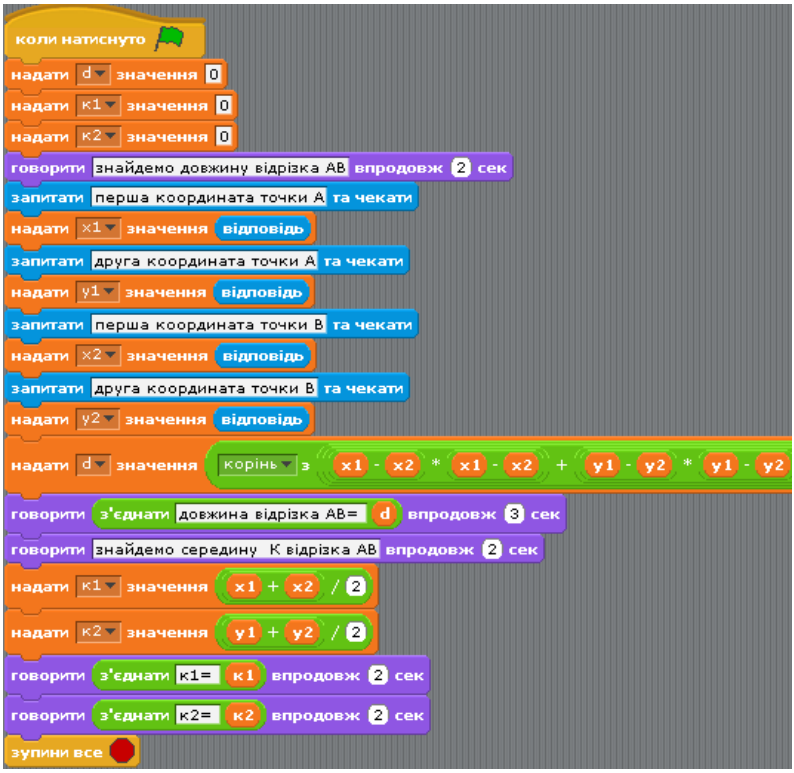
При розв'язанні математичних задач у Scratch широко використовуються і сталі, і змінні, і вирази, і функції, і списки.

Приклад 9. *Площа і периметр.* Для прямокутника чи прямокутного трикутника знайти площу і периметр.

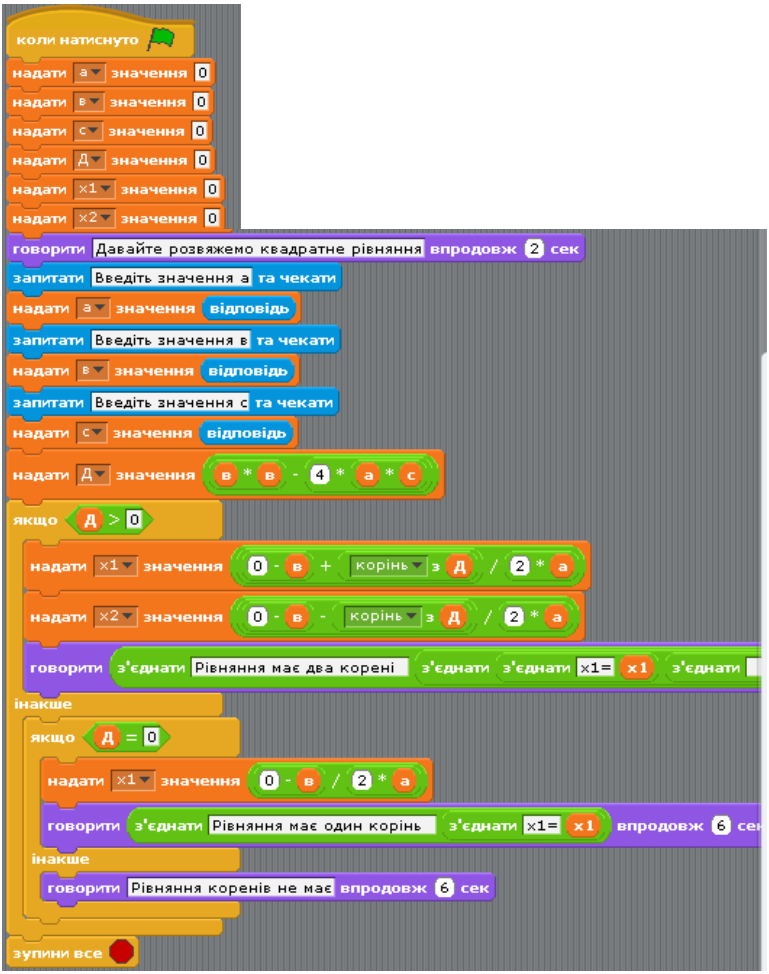


Приклад 10. *Довжина відрізка.* Задано координати двох точок на площині. Знайти довжину і середину відрізка.

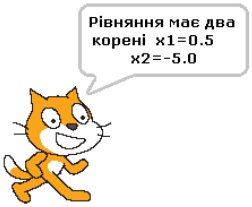
Якщо $A(x_1, y_1), B(x_2, y_2)$, то довжина відрізка $AB = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$, а середина відрізка $K(k_1, k_2)$, $k_1 = \frac{x_1+x_2}{2}, k_2 = \frac{y_1+y_2}{2}$.



Приклад 11. *Квадратне рівняння.* Створити проект для розв'язування квадратного рівняння $ax^2 + bx + c = 0$. Далі наведено скрипт і результат розв'язання квадратного рівняння $2x^2 + 9x - 5 = 0$:


коли натиснуто
надати а значення 0
надати в значення 0
надати с значення 0
надати Д значення 0
надати х1 значення 0
надати х2 значення 0
говорити Давайте розв'яжемо квадратне рівняння впродовж 2 сек
запитати Введіть значення а та чекати
надати а значення відповідь
запитати Введіть значення в та чекати
надати в значення відповідь
запитати Введіть значення с та чекати
надати с значення відповідь
надати Д значення в * в - 4 * а * с
якщо Д > 0
надати х1 значення 0 - в + корінь з Д / 2 * а
надати х2 значення 0 - в - корінь з Д / 2 * а
говорити з'єднати Рівняння має два корені з'єднати з'єднати х1= х1 з'єднати
інакше
якщо Д = 0
надати х1 значення 0 - в / 2 * а
говорити з'єднати Рівняння має один корінь з'єднати х1= х1 впродовж 6 сек
інакше
говорити Рівняння коренів не має впродовж 6 сек
зупини все


а 2
в 9
с -5
Д 121
х1 0.5
х2 -5.0


Рівняння має два корені х1=0.5 х2=-5.0

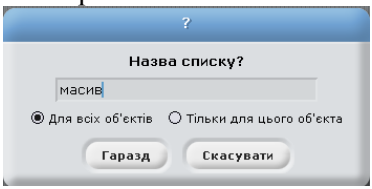
7. *Списки*

Список в Scratch приблизно відповідає одновимірному динамічному масиву. Це впорядкований набір елементів (комірок пам'яті). На елементи можна посилатися за допомогою імені масиву та індексів.

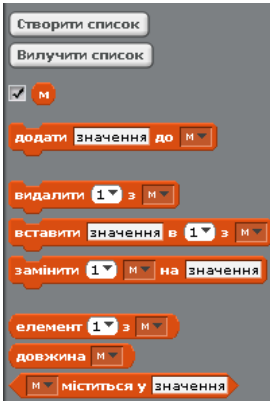
Використовувати списки зручніше, ніж велику кількість однотипних змінних. Наприклад, при обчисленні суми елементів.

Щоб ***створити список***, треба вибрати кнопку 

з категорії **Змінні**. Появиться вікно,



де треба ввести назву списку (наприклад, **масив**) та вказати, чи список буде глобальним, чи локальним; далі вибрати **Гаразд.** Побачимо такі блоки у полі команд:



Якщо поставити галочку зліва від назви списку 

, то список відобразиться у правому верхньому кутку сцени (***монітор списку***):

На моніторі списку буде відображатися назва списку, порядкові номери та значення елементів списку, а також їх кількість (довжина).

Вводити елементи списку можна двома способами:

1) за допомогою використання у програмі блоку 

(причому елементи можна задавати у програмі або питати у користувача);
2) за допомогою монітора списку. Перед виконанням програми вводити елементи на моніторі списку. Для цього натискати мишкою на  , а потім вводити у поле значення першого члена списку і клацнути мишку поза полем. Щоб набрати другий член списку, треба не клацати мишку, а натиснути ***Enter*** або ж знову вибрати  . Одержимо список **масив** з двома елементами, наприклад:

За допомогою блоків категорії **Змінні,** які виникають після створення списку, можна здійснювати такі операції:

– додавати елементи в кінець списку;

– замінити перший (або інший) елемент списку на новий;

– видалити перший (або інший) елемент списку;

– вставити новий елемент на місце першого (або іншого) елементу списку;

– відображає перший (або інший) елемент

списку;

– число, що дорівнює кількості елементів списку;

– відображає всі елементи списку підряд (як одне число чи одне слово);

– блок використовується при перевірці умови, чи задане значення є елементом списку.

Для посилання в циклі на елемент з номером ***i*** треба створити нову змінну (лічильник), якій надавати відповідного значення ***i***.

Якщо список вже не потрібний, його видаляють, вибравши кнопку 

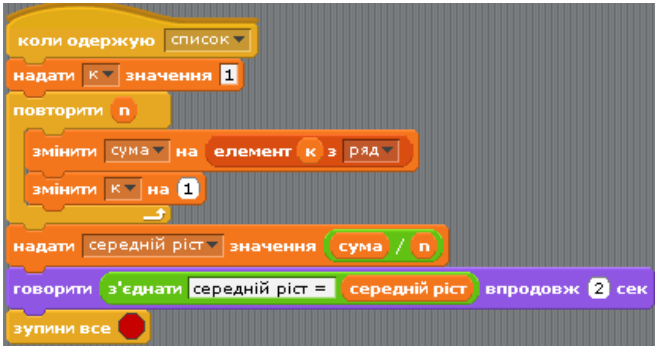
.

Приклад 12. *Список зріст.* Створити список зросту учнів класу і знайти середній зріст.

Введемо змінні ***n*** – кількість учнів у класі, **сума** – сума зросту всіх учнів класу, **середній ріст** – це частка від ділення **суми** на ***n***, ***к*** – лічильник циклу.

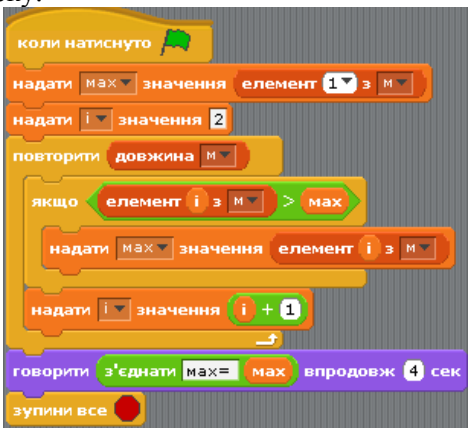
Спочатку запишемо скрипт, у якому створимо список **ряд**, де запишемо зріст кожного учня згідно з порядковим номером. На початку очистимо попередній список. Значення змінної ***n*** та елементів списку вводить користувач з клавіатури.

У наступному скрипті знайдемо середній зріст:



Приклад 13. *Список max.* Знайти максимальне значення серед елементів списку.

Спочатку створимо список **м** на моніторі списку. Змінна **max** містить найбільший елемент, а змінна **і** грає роль лічильника циклу. На початку змінній **max** надаємо значення першого елемента з списку. Далі перевіряємо, чи наступний елемент списку більший за **max**. Якщо так, то тепер **max** вже дорівнює цьому наступному елементу. І так перевіряємо всі елементи списку.



Практична робота № 6

Використання змінних, виразів та списків у проектах

1. Ввід та вивід даних.
2. Використання змінних.
3. Складання виразів.
4. Побудова графіків функцій.
5. Створення, використання та перетворення списків.

Завдання

1. Створити проект ***«(Прізвище) Вираз»***, де передбачити задання чисел a та b з клавіатури, обчислити значення заданого виразу та вивести результат.

Задача 1

1. $(a^2 - 5a + 2)(\cos(2b))^2$.
2. $(ab + 2b^2)\sqrt{|a - 7|}$.
3. $(8a - 5b)e^{4+ab}$.
4. $\sin(a + 2)/(a^3 + b^2 - 1)^2$.
5. $tg((ab - 1)/2a^2)$.
6. $ln|a|ln|b - 2|(ab + 3)$.
7. $(3a^3 + 2a^2 + b + 9)arcsin(ab)$.
8. $(b^3 - \sin a)(a^2 + 3ln|b|)$.
9. $(9 - 3a + b^2)/e^{a-7}$.
10. $\cos((4b - 2a + 7)/-e^a)$.
11. $10^a \times b^2 - \sqrt{\sin(a + 1)}$.
12. $arccos\left(\frac{a}{b}\right) + ln|1 - a|$.
13. $4tg^2\left(\frac{1}{ab} + 5\right)$.
14. $2(ab - a^2 + b^2)^{-1}|\sin a + 5|^{-1}$.
15. $(a^2 + 4b + 9)(\sin(ab))^3$.
16. $(a + 2b + b^2)/e^{3-2a}$.
17. $(a - b^2)\sqrt{|a + 1|}$.
18. $cos(a + b)/(b^2 - 1)^2$.
19. $ln|a|ln|b|/(a + 3)$.
20. $(a^2b - 9)(\ln(ab))^3$.
21. $(ab - b^3)\sqrt{|2b - 7|}$.

22.$(b^2 - \ln(\sin a))a^2$.

2. Створити проект ***«(Прізвище) Графіки»***, у якому в одній системі координат різними кольорами побудувати графіки заданих функцій 2(а) та 2(б). Проміжок зміни аргумента ввести з клавіатури. Систему координат на початку малює виконавець.

Задача 2(а)

1. $y = 10x + 5$.
2. $y = -6x + 20$.
3. $y = -x - 15$.
4. $y = 14 - 8x$.
5. $y = 5 + 16x$.
6. $y = 2x + 10$
7. $y = 25 - x$.
8. $y = 6x + 14$.
9. $y = -5 - 9x$.
10. $y = 8x + 18$.
11. $y = 4x - 21$.
12. $y = 20x + 5$.
13. $y = -x + 23$.
14. $y = 3x - 20$.
15. $y = 5 - 8x$.
16. $y = 9x - 4$.
17. $y = 5x + 15$.
18. $y = 2x - 20$.
19. $y = -2x + 5$.
20. $y = 10 - 8x$.
21. $y = 7x + 4$.
22. $y = 15 - x$.

Задача 2(б)

1. $y = 4x^2 + 2x - 3$.
2. $y = 4x^2 - 2x - 3$.
3. $y = -4x^2 + 2x + 3$.
4. $y = 4x^2 + 2x + 3$.
5. $y = -4x^2 - 2x - 3$.

6. $y = 4x^2 - 2x + 3.$
7. $y = -4x^2 + 2x - 3.$
8. $y = -4x^2 - 2x + 3.$
9. $y = 4x^2 + 2x.$
10. $y = 4x^2 + 3.$
11. $y = -4x^2 - 3.$
12. $y = -4x^2 - 2x.$
13. $y = 4(x^2 + x - 3).$
14. $y = 3x - 4x^2.$
15. $y = x^2 + 4x + 5.$
16. $y = -2x^2 - x - 4.$
17. $y = 2x^2 - x - 4.$
18. $y = -2x^2 + x - 4.$
19. $y = 2x^2 - x$
20. $y = 6x^2 - 3x - 14$
21. $y = -3x^2 + 6x - 4.$
22. $y = x^2 - 4.$

3. Створити проект ***«(Прізвище) Формула»***, у якому буде розв'язана поставлена задача з геометрії.

Задача 3

1. З квадратного листка паперу із стороною 10 см вирізали прямокутник із сторонами 3 і 5 см. Яка площа куска, що залишився?
2. Цеглина має розміри 30х15х7 см. Яка площа стіни товщиною 7 см, складеної зі 100 цеглин?
3. Шиба на вікні має розміри 100х50 см. У будинку 36 однакових вікон з однією шибкою. Скільки потрібно скла для всіх вікон у будинку?
4. Електрична опора має форму літери А. Відстань між закопаними стовпами 2м 40см. Яка довжина горизонтальної перетинки, якщо вона прикріплена на половині висоти опори?
5. Скільки потрібно кружева, щоб обшити навколо прямокутну серветку розміром 80см на 150см?

6. Прямокутну кімнату розміром 4х6 м застелили двома однаковими килимами, довжина яких 2м 50см, а ширина 1м 25см. Яка площа підлоги, що залишилась незастеленою?
7. Один квадратний метр облицювальної плитки вартує 250 грн. Скільки треба заплатити за плитку для стін і підлоги кімнати, довжина якої 2м 50см, ширина – 2м, а висота – 2м 80см? Кімната без вікон, двері розміру 1х2 м.
8. Скільки квадратних метрів фанери потрібно, щоб змайструвати куб, ребро якого має довжину 80см?
9. Огорожею довжиною 800м обгороджено трикутну ділянку, дві сторони якої 220м і 310м. Скільки коштує огорожа третьої сторони, якщо вартість всієї огорожі 96000 грн?
10. Схил гірки для катання на сноутюбах має кут нахилу 30^0 , висоту 7 м та ширину 3 м. Скільки треба штучного покриття для схилу?
11. Треба утеплити бічну стінку будинку, на якій немає вікон. Ширина стінки 15м, а висота 17,5 м. Скільки треба листів пінопласту розміром 1х1.5 м?
12. Треба обгородити прямокутну ділянку для випасу худоби трьома рядами колючого дроту. Скільки треба дроту, якщо розміри ділянки 350х400 м?
13. Знайти площу трикутника за формулою Герона.
14. Знайти сторону трикутника за двома іншими сторонами і кутом між ними (за теоремою косинусів).
15. Знайти площу паралелограма за двома сторонами і кутом між ними.
16. Знайти сторону ромба за двома діагоналями.
17. Знайти гострі кути прямокутного трикутника за двома діагоналями.
18. Знайти площу кругового сектора за радіусом кола та центральним кутом.
19. Знайти площу кругового сегмента за радіусом кола та центральним кутом.
20. Знайти площу правильного n-кутника (n=3;4;5).
21. Знайти радіус описаного навколо правильного n-кутника кола за стороною n-кутника (n=3;4;5).

22. Знайти радіус вписаного в правильний n-кутник кола за стороною n-кутника (n=3;4;5).
23. Знайти кут між векторами, заданими координатами.
24. Обчислити визначник 3-го порядку.
25. Знайти проекцію одного вектора на напрям іншого вектора, якщо задані координати векторів.
26. Перевірити, чи два задані координатами вектори ортогональні.

4. Створити проект ***«(Прізвище) Список»*** для роботи із створеним списком. При необхідності використовувати допоміжні змінні.

Задача 4

1. Знайти суму елементів списку, які не дорівнюють 6.
2. Знайти всі від'ємні елементи списку.
3. Знайти кількість додатних елементів списку та їх суму.
4. Знайти середнє арифметичне від'ємних елементів списку.
5. Знайти кількість елементів списку, які менші за 5.
6. Знайти елементи списку які лежать в межах від 1 до 10.
7. Посортувати елементи списку у порядку зростання (створити новий список).
8. Замінити від'ємні елементи списку їх квадратами.
9. Додатні елементи списку збільшити удвічі, а нульові вилучити.
10. Непарні елементи списку поділити на 3 і заокруглити.
11. Парні елементи списку замінити на протилежні числа.
12. Якщо останній елемент у списку менший за перший, то поміняти їх місцями. Інші елементи зменшити на 2.
13. Вилучити із списку всі елементи, кратні 3.
14. У списку є числа від 1 до 10. Вилучити з нього всі прості числа.
15. Замінити додатні елементи списку числами -5 та знайти їх кількість.
16. Знайти суму та кількість парних додатних елементів списку.
17. Знайти суму елементів списку, які стоять на парних місцях і є додатними.

18. Знайти найменший елемент списку і збільшити його на 1.
19. Якщо перший елемент списку від'ємний, то замінити його нулем. Інші елементи списку збільшити вдвічі.
20. Вилучити із списку всі непарні елементи, знайти їх кількість.
21. Додатні елементи списку замінити їх кубами і порахувати кількість перетворених елементів.
22. Знайти суму елементів списку, які стоять на непарних місцях і є від'ємними.

Тема 6 . Створення вікторин, тестів, загадок, ігор в Scratch

1. Планування проекту.
2. Використання випадкових факторів.
3. Застосування лічильника.
4. Організація і використання таймера.
5. Створення вікторин, тестів, загадок.
6. Приклади ігор.

1. Планування проекту

Засобами Scratch можна створювати різноманітні проекти, зокрема, тести, вікторини, загадки, ігри.

Для створення гри чи іншого проекту потрібно:

1) придумати суть проекту (мету гри, тематику і завдання тестів чи вікторин, набір загадок);
2) визначити правила гри;
3) вибрати чи створити виконавців, фони сцени та звуки;
4) створити скрипти для виконавців та фонів;
5) редагувати проект.

Правила гри для користувача (як грати гру, як виконувати тести чи вікторини) записують одним із таких способів:

a) вказують правила в інформації про проект у полі ***«Про цей проект»*** при збереженні проекту, тоді їх читають при відкриванні гри;
b) вказують назву проекту та правила гри на першому фоні;
c) залишають правила на сцені протягом всього користування проектом.

Найважче придумати суть гри та реалізувати її методами Scratch.

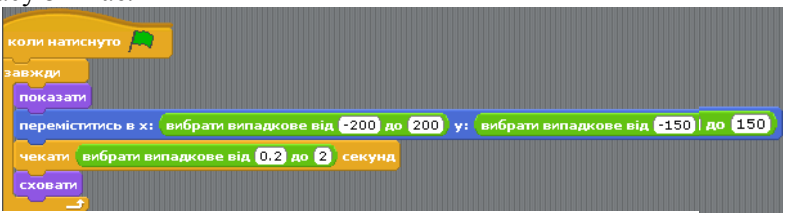
У наступних трьох пунктах розглянемо деякі прийоми, що використовуються у проектах.

2. Використання випадкових факторів

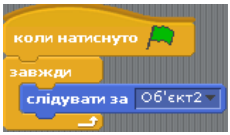
В проектах часто використовується випадково вибране число. Для цього використовують блок 

з категорії **Оператори**. В обох віконечках вибираються довільні числа (і цілі, і дробові, і додатні, і від'ємні), причому якщо обидва цілі, то і випадкове буде ціле, а якщо хоч одне дробове, то випадкове дробове.

Приклад 1 . *Поява у випадковій точці.* Деякий об'єкт з'являється у випадковій точці і через випадковий проміжок часу зникає.



Котик слідкує за цим об'єктом :

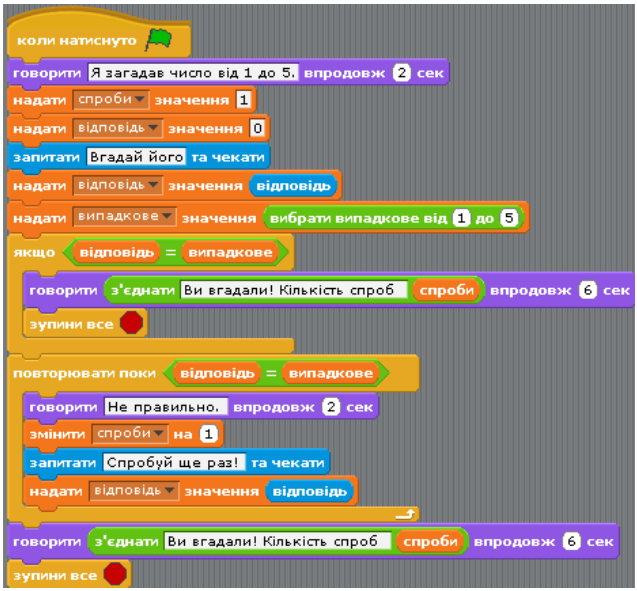


3. Застосування лічильника

Дуже часто в проектах потрібно додавати чи віднімати очки, ставити оцінки, вести рахунок гри тощо. Для цього використовують лічильники. Лічильниками служать змінні. Вводять одну чи кілька змінних, які в процесі виконання програми змінюють та (чи) аналізують.

Приклад 2. *Вгадати число від 1 до 5.* Рахувати кількість спроб.

Введемо три змінних: змінна ***випадкове*** містить задумане число, вибране випадковим чином; змінна ***відповідь*** містить пропоноване користувачем значення числа, а змінна ***спроби*** містить кількість спроб вгадати число.



4. Організація і використання таймера

Багато тестів чи ігор враховують час. Наприклад, переможе найшвидший, гра не може перевищувати певний проміжок часу або дається обмежений час на відповідь тощо. Для врахування часу у Scratch використовується таймер.

Для реалізації цього у категорії **Датчики** є два блоки:  ;  .

При натисканні лівої клавіші мишки на віконечко зліва від блоку  відбуваються такі дії:

- у віконечку появляється галочка  ;
- починається відлік часу;
- на сцені появляється монітор таймера із значенням часу  .

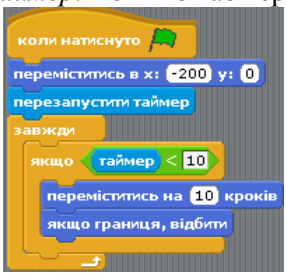
Слід зауважити, що час на моніторі таймера зображується числом з одним знаком після коми. Ціла частина вказує кількість секунд.

З натисканням блоку  починається відлік часу з нуля.

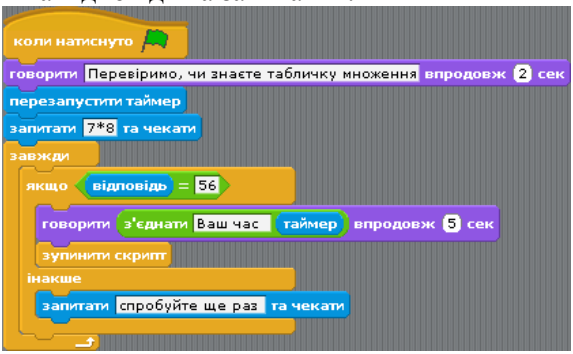
Припинити відлік часу можна, прибравши галочку з віконечка біля блоку 

.

Приклад 3. *Таймер.* Котик бігає перші 10 секунд.



Приклад 4. *Відповідь на швидкість.* Рахується час, затрачений на відповідь на запитання.

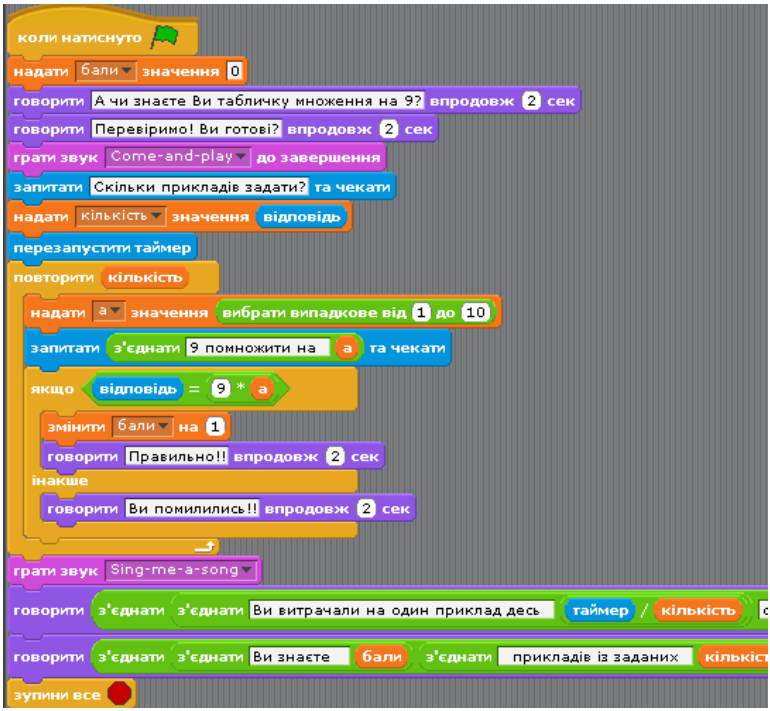


5. Створення вікторин, тестів, загадок

Засобами Scratch можна легко створювати тести, вікторини, загадки, завдання з різних навчальних дисциплін.

Приклад 5. *Тест «Таблиця множення на 9».* Приклади вибираються випадково. Кількість прикладів задає користувач. Рахується кількість правильно розв'язаних прикладів та середній час на виконання одного прикладу. Тест починається і закінчується звуковим сигналом.

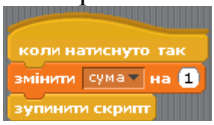
Аналогічно можна зробити тест на перевірку всієї таблички множення.



Приклад 6. *Тест «Чи вмієш ти цінувати час».* У тесті є 7 запитань, на які треба відповідати «так» або «ні». Треба вибирати потрібну відповідь мишкою.

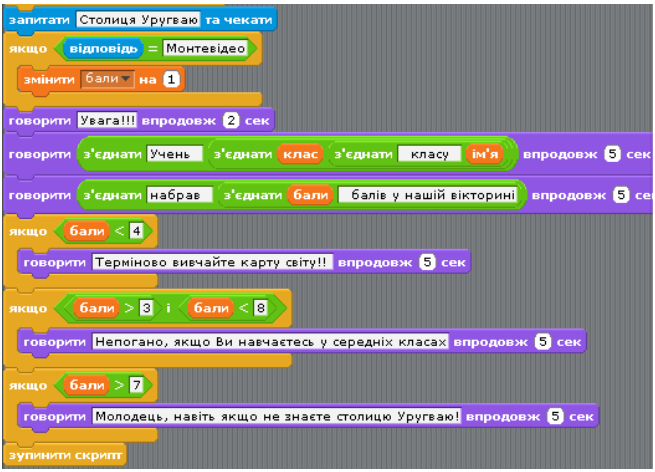
Проект можна організувати так. Кіт прикріплений до мишки:

Відповіді «так» і «ні» є спрайтами. Для спрайта **так** наступний скрипт рахує відповіді «так» при натисканні на кнопку **так:**



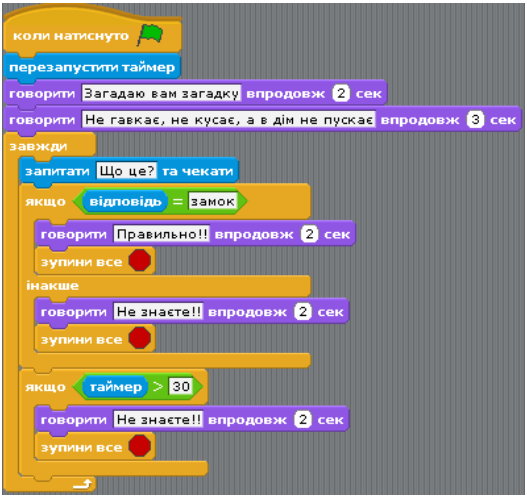
У залежності від кількості відповідей «так» видається висновок.

Приклад 7. *Вікторина «Столиці».* Вгадати 9 столиць держав, отримати по одному балу за кожну правильну відповідь і коментар про рівень своїх знань.



Приклад 8. *Відгадай загадку.* Кіт загадує вам загадку. Якщо немає відповіді протягом 10 секунд, то мишка дає підказку. Далі дається ще 20 секунд на відповідь.

Скрипт для кота:

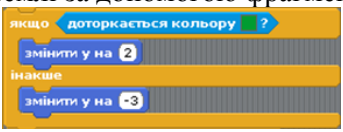


Скрипт для мишки:

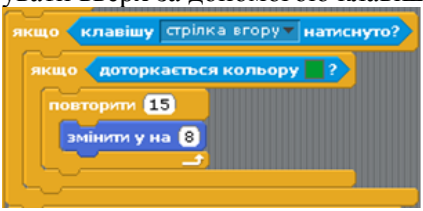
6. *Приклади ігор*

Приклад 9. *Гра «Кіт йде додому».* Котик шукає свій дім. Але на дорозі є ділянки з вулканічною лавою. Якщо він потрапить у лаву, то загине. А якщо дійде до свого будиночка, то зрадіє, тоді гра закінчується перемогою.

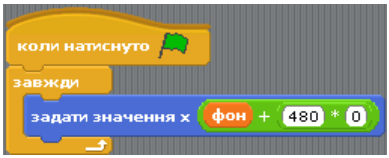
Рух котика організовано таким чином. Він рухається лише по поверхні землі за допомогою фрагмента

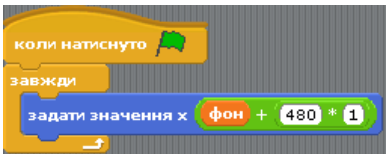


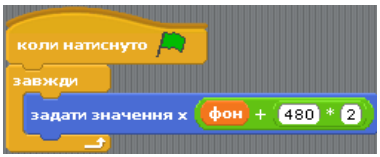
і може підстрибувати вверх за допомогою клавіші *вгору*:

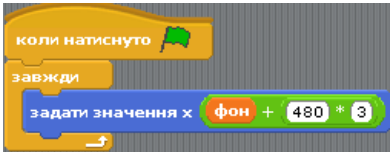


Якщо натискати клавіші *вліво* чи *вправо*, то кіт стоїть на місці, але у протилежному напрямку рухається дорога. Дорога склеєна з чотирьох частин, які є виконавцями. Скрипти для цих частин:



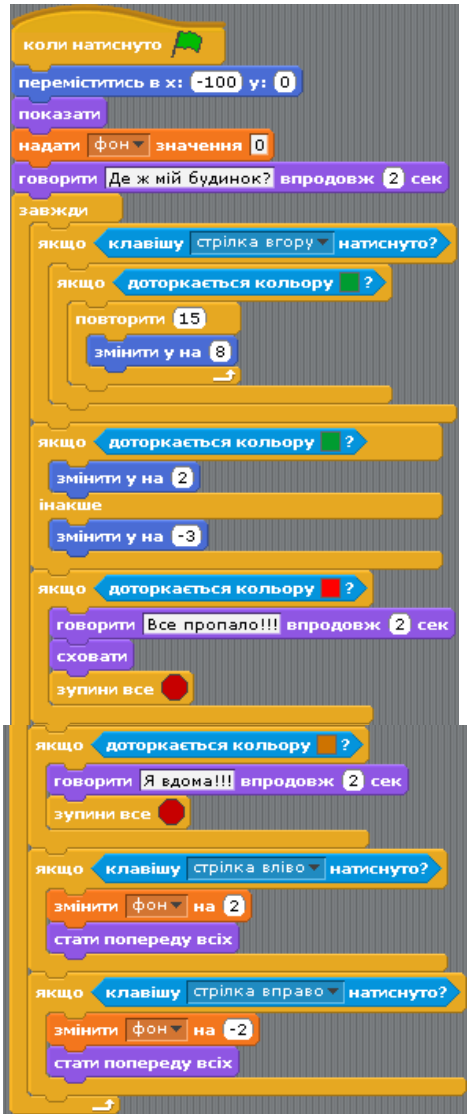






Вигляд сцени на початку і в кінці гри:

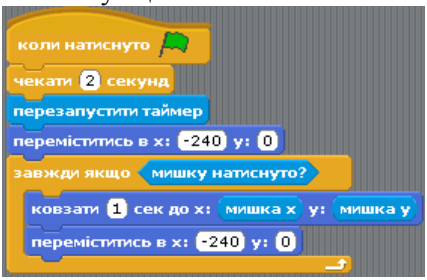
Скрипт для кота:

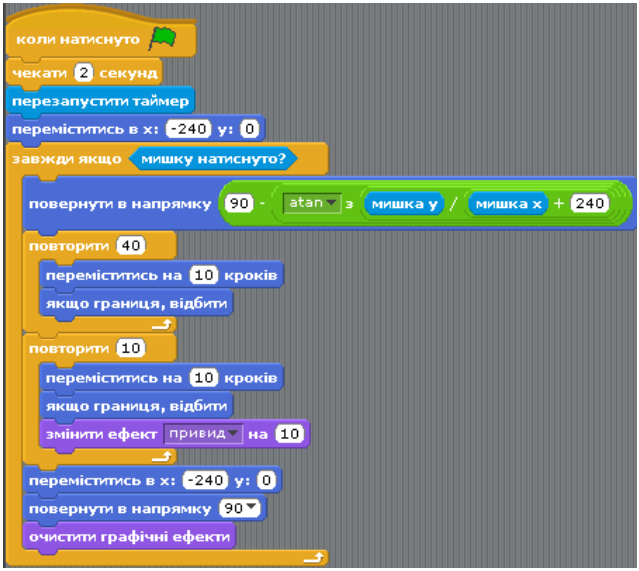


Приклад 10. *Гра «Котобоулінг».* Треба порозганяти котів м'ячем. М'яч вилітає зліва і летить в точку, де клацаємо мишкою. Якщо м'яч зачіпає кота, то кіт зникає. Гра закінчена,

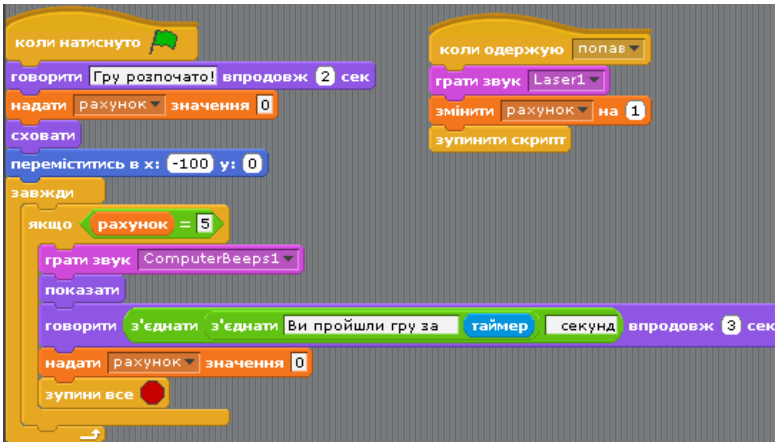
якщо всі п’ять котів, що рухаються хаотично, зникли. Суддя починає і закінчує гру, а також об’являє час гри.

Можна по-різному вираховувати траєкторію м’яча. Далі наведено два варіанти скрипта для м’яча. У першому м’яч летить у точку, де клацнули ліву клавішу мишки. У другому вираховують напрям польоту м’яча за координатами x і y мишки за формулою $90^0 - arctg\frac{y}{x+240}$, причому м’яч через певну відстань поступово зникає (ефект *привид*), а потім знову з’являється з лівого боку сцени.

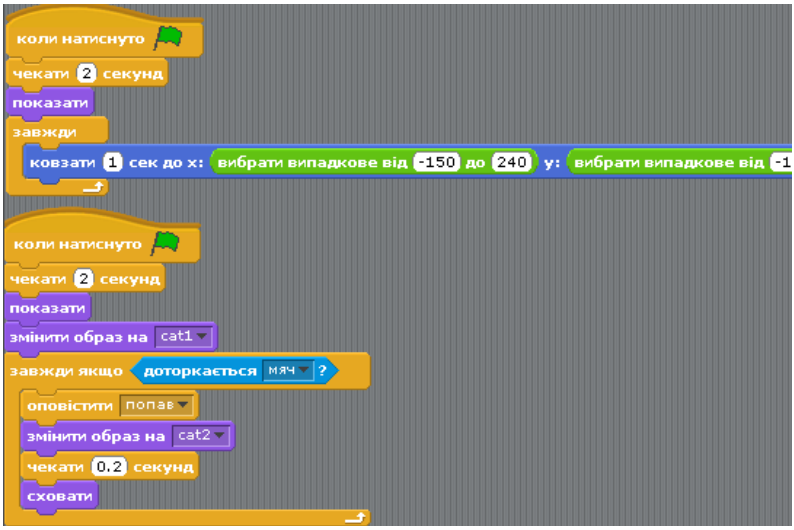




Скрипти для судді:



Скрипти для кожного з п’яти котів:



Так виглядає сцена на початку гри:

Практична робота № 7

Створення тестів та ігор у Scratch

1. Використання лічильників.
2. Використання таймера.
3. Створення тестів.
4. Створення ігор.

Завдання

1. Створити проект ***«(Прізвище) Тест»***, у якому користувач вибирає одну з відповідей на питання тесту і отримує висновки на підставі своїх відповідей.
2. Створити проект ***«(Прізвище) Вікторина»***, у якому треба вводити відповіді на запитання з якогось шкільного навчального предмету. Виставляється оцінка за результатами відповідей.
3. Створити гру ***«(Прізвище) Стрілялка»***, у якій Об’єкт1 стріляє у Об’єкт2. Об’єкт1 управляється клавішами вліво і вправо та рухається вздовж нижньої сторони сцени. Об’єкт2 з’являється зверху у випадковому місці і стоїть там випадкову кількість секунд. Об’єкт1 має встигнути дійти до місця з тою ж координатою, що і Об’єкт2, і стрілити вверх у Об’єкт2 (можна стріляти якимось Об’єктом3). При попаданні Об’єкт2 змінює образ і зникає, а Об’єкт1 отримує

одне очко. Треба рахувати кількість влучень. Гра триває 30 секунд. Зробити висновки про влучність. Щоб було цікавіше грати, виберіть фон і звуки.

4. Створити гру ***«(Прізвище) Обхід»***, у якій Об'єкт має за найкоротший час пройти по лабіринту (трубі, тунелі тощо) до виходу. Об'єкт управляється клавішами вверх і вниз. Клавіші вліво і вправо рухають фони (4шт), що є спрайтами (як у прикладі 9). Але випадково появляються перешкоди, торкання до який забирає 1 життя з трьох. Втрата всіх трьох життів веде до програшу, а доторкання до виходу з лабіринту є виграшем. Вкінці гри об'явити кількість витрачених життів та час проходу лабіринту.

Тема 7. Інші версії Scratch

1. Модифікації середовища Scratch 1.4.
2. Версія Scratch 2.0.
3. Версія Scratch 3.0.

1. Модифікації середовища Scratch 1.4

На основі вихідного коду Scratch 1.4 було створено декілька модифікацій мови. Опишемо коротко деякі з них.

BYOB (Snap!) розроблено в Університеті Берклі (Каліфорнія, США). Основним розширенням мови, яке було введено в BYOB, стала можливість побудови користувацьких блоків — аналога процедур звичайних мов програмування. Підтримуються рекурсія, замикання і лямбда-вирази. Також додано налагоджувач і можливість компіляції у виконувані файли, вкладені спрайти, багатомірні списки, покращена робота зі скролінгом і компіляція виконуваних файлів. Починаючи з версії 3.1 в BYOB додана підтримка об'єктно-орієнтованого програмування — спрайти BYOB тепер дозволяють наслідування на основі прототипів. Для цього введено механізм клонування спрайтів.

Panther розширює Скретч підтримкою drag-n-drop керування спрайтами, можливістю клонування спрайтів (об'єктів), і підсистемою CYOB (яка, як і BYOB, дозволяє створювати власні командні блоки, але вимагає для цього знання мови Squeak (сучасний діалект Smalltalk, на якому написано Скретч 1.4 і його модифікації).

Механізм Mesh. І BYOB, і Panther включають підтримку механізму mesh, що забечує взаємодію скретч-програм через мережу за допомогою роздільних змінних і посилання широкомовних повідомлень.

Slash доповнює можливості BYOB 3.0 новими можливостями Panther: клонуванням і перетягуванням спрайтів (об'єктів).

StarLogo TNG. У 2008 році в Массачусетському технологічному інституті розроблено навчальну мову програмування StarLogo TNG, яка розширює StarLogo можливостями тривимірної графіки і мовою візуального

блокового програмування. На відміну від оригінального OpenStarLogo і MIT Scratch — StarLogo TNG не є програмним продуктом з відкритим вихідним кодом.

Scratch 2.0. У лютому 2011 року була випущена перша онлайнова бета-версія Скретч 2.0, в який планувалося включення деяких з можливостей BYOB (таких, як створення призначених для користувача процедур), векторної графіки, клонування спрайтів, можливість групової роботи над проектами і т. д. Офіційно бета-версія вийшла в реліз 9 травня 2013.

Scratch 3.0 є третьою і поточною основною версією Scratch. Це повна переробка і повторна реалізація Scratch, написана на HTML5 і JavaScript. Він має новий, сучасний вигляд і дизайн, а також сумісний з багатьма мобільними пристроями і не вимагає Flash. Випущений 2 січня 2019 року.

Ще одним середовищем програмування, який використовує візуальні блоки команд є **App Inventor** — експериментальна система візуального програмування для платформи Android.

Більш прямим аналогом мови Scratch на Android є мова **Catroid**, яка розробляється в Інституті Технологій Програмного забеспечення Технічного Університету Граца в Австрії.

Однією з найновіших модифікацій Scratch 3.0 є **Turbo Warp**. Тут є багато додаткових можливостей, зокрема, запис відео виконання проекту, можливість перегляду на гаджетах без завантаження програмного середовища, прискорення виконання проектів тощо.

2. Версія Scratch 2.0.

2.1. Історія

Scratch 2.0 — друга мажорна версія Scratch після Scratch 1.4 . Вона працює на основі технології Adobe Flash. Оновлення змінило вигляд сайту і включає в себе онлайновий редактор проектів і автономний редактор для роботи без Інтернету. З'явилась можливість створювати власні блоки в межах проектів, а також ряд інших удосконалень.

Scratch 2.0 був анонсований скретчером *andresmh* на форумах Scratch в січні 2010 року. Перший експериментальний випуск Scratch 2.0 від спеціальної частини команди Scratch стався в серпні 2010. Пізніше, в 2011, була представлена версія плеєра на технології Flash, і користувачі могли вибирати між плеєром Java або Flash. У 2012 році плеєр був поміщений в альфа-версію редактора, а також Flash-версія плеєра стала єдиною в жовтні 2012 року.

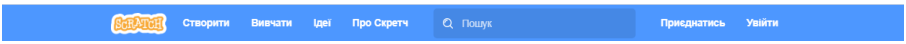
У травні 2011 року з'явилися перші відомості про преальфа-версії редактора проекту, яку показали кільком людям на День Скретчу . Пізніше команда Scratch почала публікувати новини про оновлення в своєму блозі. Новий сайт і перероблений редактор проекту був доступний на сайті альфа-версії Scratch (alpha.scratch.mit.edu) і доступний кілька днів на Дні Скретчу у 2012 і наступному році. Весь 2012 рік люди допомагали тестувати альфа-версію Скретч: модератори, радники спільноти, деякі вчителі, колишні куратори, куратори дизайнерської студії Scratch, куратори ТИ (TBG) і група з 500 добровольців. Деякі користувачі також могли взяти участь через збої системи.

У грудні 2012 року була анонсована бета-версія Scratch. 28 січня 2013 бета-версія стала доступна публічно на сайті beta.scratch.mit.edu . Після доопрацювання сайт Scratch переїхав на колишній домен scratch.mit.edu .

Сайт бета версії і версії 1.4 були прибрані за непотрібністю командою Scratch з 6 по 8 травня 2013 р. Всі проекти були збережені. Проте версію 1.4 можна і досі завантажити за посиланням https://scratch.mit.edu/scratch_1.4/.

2.2. Завантаження версії Scratch 2.0

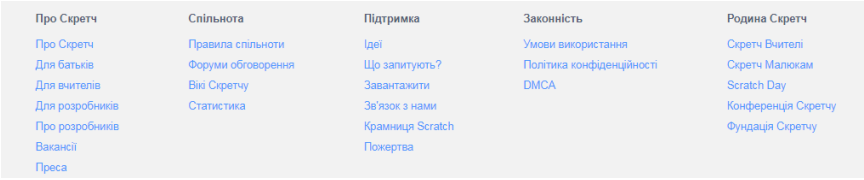
Щоб працювати у середовищі Scratch, потрібно зайти на офіційний сайт scratch.mit.edu. Появиться головне вікно із таким верхнім рядком:



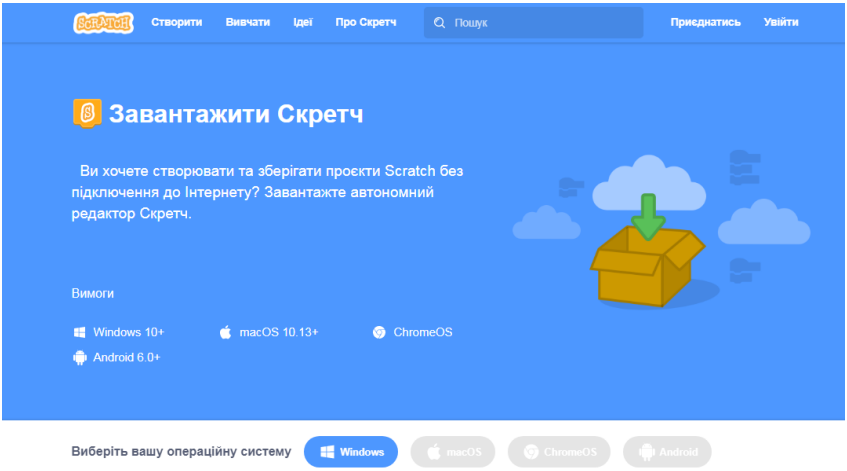
Щоб працювати в ***онлайновому режимі***, треба вибрати кнопку **Створити**. Появиться головне вікно поточної версії Scratch 3.0, де можна працювати.

Якщо вибрати **Увійти**, то пропонують зареєструватися, вибравши ім'я та пароль. Створюється обліковий запис. Це зручно, якщо треба зберігати свої продукти та працювати з ними.

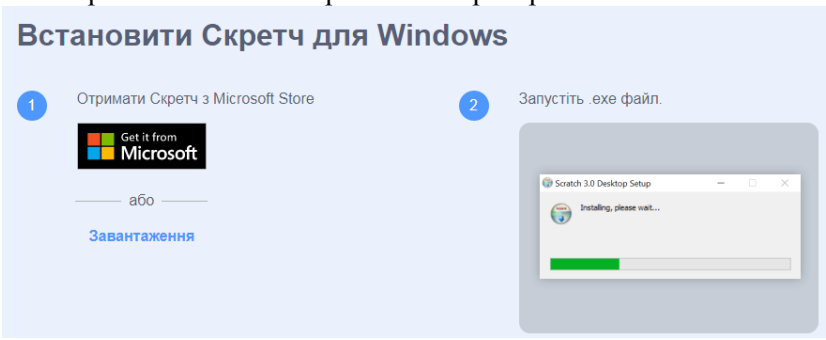
Для встановлення ***автономного (офлайнового) редактора*** Scratch для роботи без інтернет-з'єднання слід перейти на кінець головного вікна сайту.



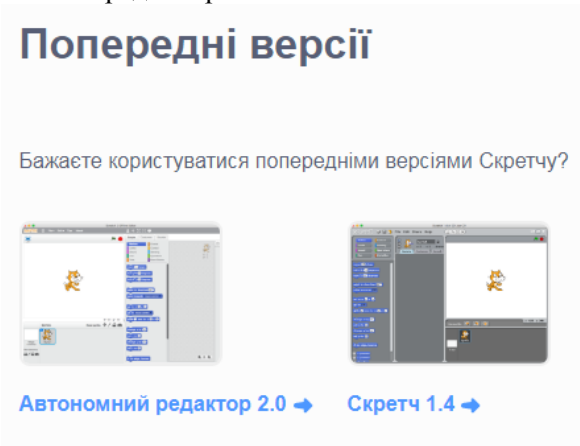
Там у розділі **Підтримка** вибрати **Завантажити**. Появиться вікно



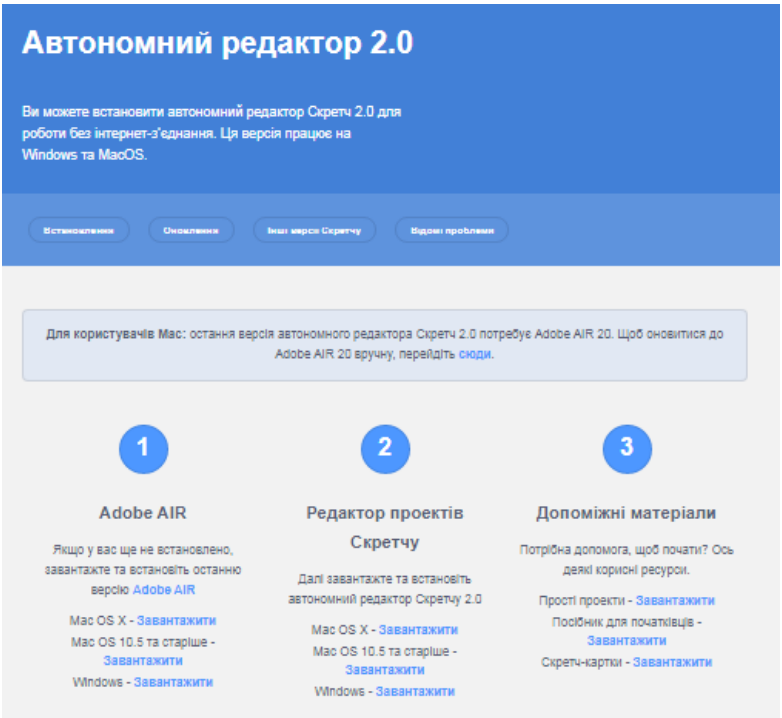
Тут описані вимоги для завантаження поточної версії 3.0. Треба вибрати свою операційну систему. Нижче показано вікно при завантаженні версії 3.0 на пристрій з ОС Windows:



Якщо у вас нижча версія, ніж Windows 10, то є можливість завантажити попередні версії 2.0 та 1.4:



При виборі Автономного редактора 2.0 появиться вікно для завантаження:



Отже, ***для встановлення офлайнового редактора Scratch 2.0*** треба:

1. Завантажити та встановити останню версію **Adobe AIR** для своєї ОС.
2. Завантажити та встановити автономний редактор Scratch 2.0 для своєї ОС, слідуючи інструкціям. На робочому столі з'явиться ярлик Scratch 2.0.

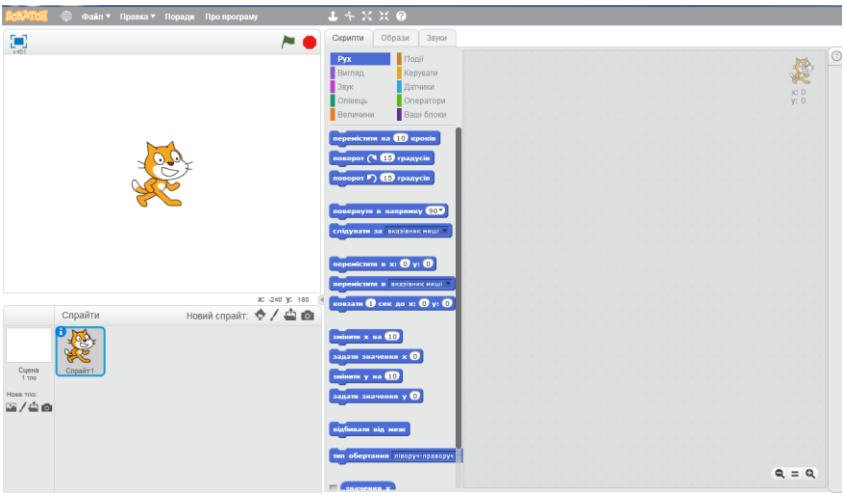
3. Завантажити допоміжні корисні ресурси: прості проекти, посібник для початківців, скетч-картки.

Автономний редактор може самостійно оновлюватися (з дозволу користувача). Він буде перевіряти наявність оновлень при запуску або можна самостійно це робити, скориставшись

командою «Перевірити наявність оновлень» у меню. На даний час поточною є версія 461.

2.3. Інтерфейс Scratch 2.0

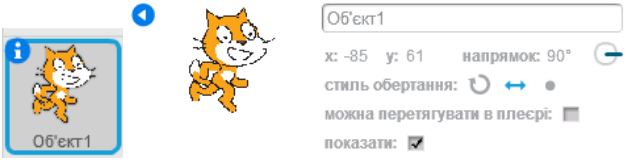
Головне вікно Scratch 2.0 має вигляд



Як бачимо, поля розміщені не так, як у версії 1.4. Проте всі можливості попередньої версії збережені і примножені.

Деякі нововведення у Scratch 2.0:

1. *Вибирати новий спрайт чи фон* (тло) можна чотирма способами: вибирати з бібліотеки, намалювати у вбудованому графічному редакторі, завантажити з файлу, взяти з камери.
2. *Інформація про спрайт* (координати, напрям, стиль обертання) не висвічується зразу, а лише при натисканні на кнопку **інфо (i)** на картинці відповідного спрайту в полі спрайтів:

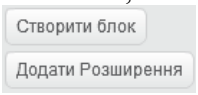


Проте у правому верхньому кутку у полі скриптів є зображення спрайта та його поточні координати (з ефектом привид):

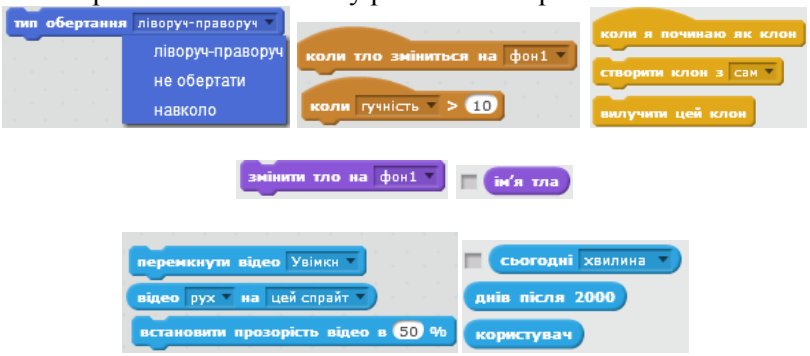
3. Внизу поля скриптів є *кнопки керування розміром скриптів*  , які дозволяють зменшити розмір скриптів і бачити всі скрипти цілком.

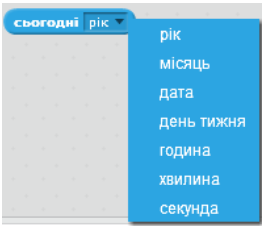
4. У Скретч 2.0 є *10 категорій команд* замість 8 у версії 1.4. Колишню категорію **Керувати** розділили на дві: **Події** та **Керувати.** У категорії **Керувати** залишилися блоки циклів, розгалужень, кінця скриптів. А у категорію **Події** віднесені блоки початку скриптів. Ще одна категорія 

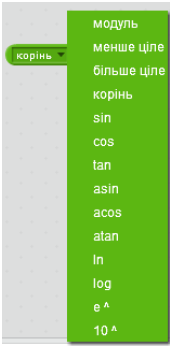
дозволяє створювати свої блоки, містить дві кнопки



5. Створено такі *нові блоки* у різних категоріях:





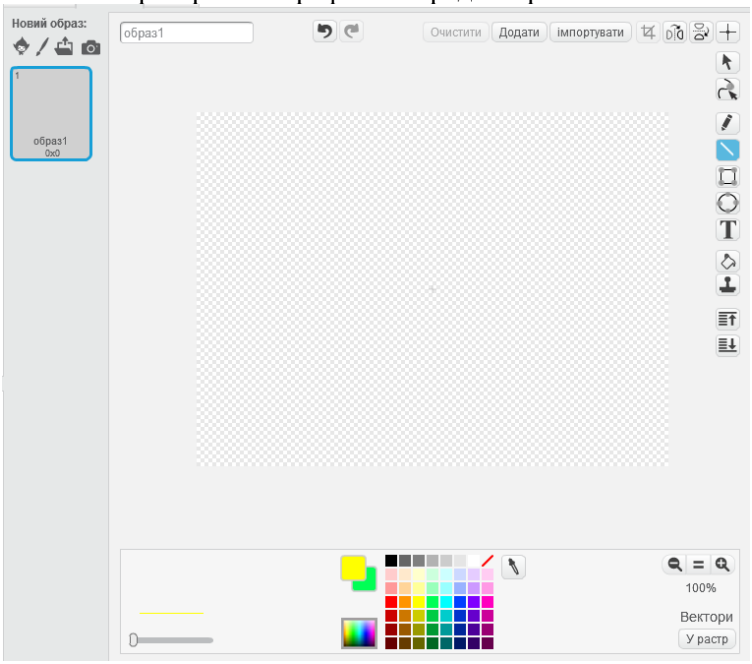


6. З'явилося багато нових спрайтів, звуків та фонів для анімації у відповідних бібліотеках.

7. По-новому подається довідкова інформація. Зокрема, є не тільки довідник блоків та зразки їх застосування, але і приклади виконання проектів.

8. Перероблено *графічний редактор*. Тут передбачені два вікна: для створення векторних та растрових зображень.

Вікно растрового графічного редактора таке:

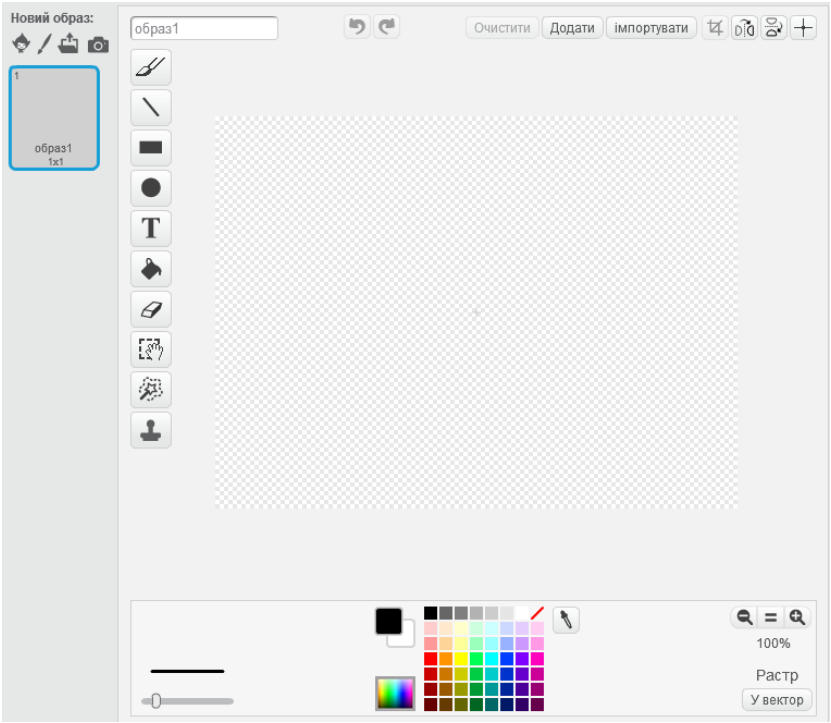


У растровому редакторі є кнопка **Реформувати**  , за допомогою якої за принципом кривих Безьє можна перетворити створене зображення:

Крім цього, є кнопки управління шарами зображення:

Шар наперед  , ***Позаду шару***  .

Вікно для векторного графічного редактора:



Зауважимо, що є можливість переходу від одного до іншого вікна графічного редактора.

3. Версія Scratch 3.0

Бета версію Scratch 3.0 випустили 1 серпня 2018 року, повний реліз був випущений 2 січня 2019. **Scratch 3.0** - офіційна поточна версія Скретч, починаючи з січня 2019 року. У новій версії доступні нові зображення, нові навчальні матеріали та нові можливості програмування. Але найголовніше – тепер програму Scratch можна встановити на планшет і продовжувати заняття вдома.

Проекти на Scratch 3.0 можна створювати в онлайні - на порталі scratch.mit.edu, або на локальному комп'ютері з використанням не мережевого редактора. Його останню оновлену версію можна скачати тут: https://scratch.mit.edu/download . При цьому звертаємо увагу, що для того, щоб встановити цей редактор на комп'ютер потрібно наявність операційної системи Windows 10+ або MacOS 10.13+.

Scratch 3.0 буде працювати на настільних комп'ютерах, ноутбуках і планшетних пристроях (iOS і Android). Крім того, проекти зможуть відтворюватися на мобільних телефонах.

Scratch 3.0 побудований на основі стандартної галузевої технології HTML5 і більше не залежить від Flash. Через це він працює в будь-якому сучасному веб-браузері. А саме, ***на комп'ютері*** підтримуються такі браузери:

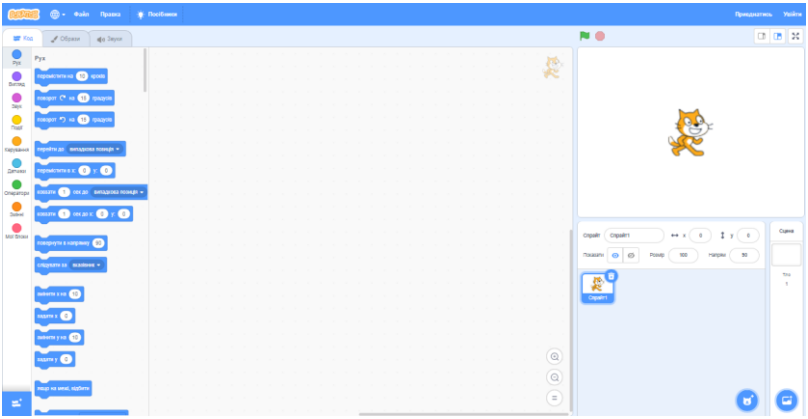
- ✓ Chrome (63+),
- ✓ Edge (15+),
- ✓ Firefox (57+),
- ✓ Safari (11+),
- ✓ Internet Explorer НЕ буде підтримуватися;

***на планшеті*:**

- ✓ Mobile Safari (11+),
- ✓ Мобільний Chrome (62+).

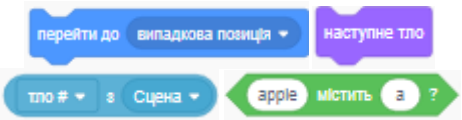
Scratch 3.0 використовує технологію браузера WebGL для рендеринга проектів на сцену. Хоча WebGL підтримується у всіх сучасних браузерах, деякі старі комп'ютери і операційні системи не можуть його підтримувати. Для користувачів, які не можуть запускати WebGL, рекомендується використовувати автономний редактор Scratch 2.0.

Головне вікно Scratch 3.0 має вигляд:

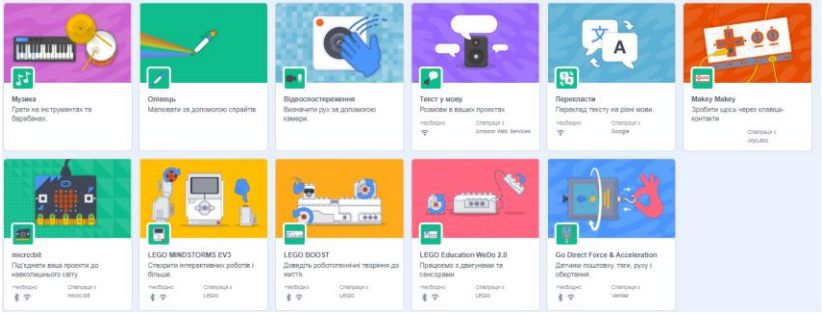


Деякі нововведення у новій версії 3.0:

1. У Scratch 3.0 використано абсолютно *нову базу коду* на основі JavaScript, що складається з декількох компонентів, таких як «Scratch-GUI», тепер базується на бібліотеці від Blockly, «Scratch-VM» для інтерпретації коду, і "Scratch - Render" — механізму візуалізації. Блоки команд генеруються за допомогою Blockly.
2. *Змінений дизайн інтерфейсу*: він схожий на Scratch 1.4 , але більш сучасний (сцена розташована справа, поле команд – зліва, поле скриптів посередині).Кнопки категорій палітри блоків розташовані зліва від них, а палітра безшовно тягнеться вниз. Кнопками можна взагалі не користуватися, так як всі блоки розташовані один за одним.
3. *Добавлені нові блоки*:



4. Деякі блоки (наприклад, «точка в напрямку») мають більш візуальний і інтуїтивний спосіб вибору входів.
5. Всі проекти тепер починаються зі змінною (що називається «моя змінна»), щоб зробити їх більш помітними для початківців.
6. *Змінений колір і розмір деяких блоків*. Блоки в Scratch 3.0 стали ширші, а у деяких змінився колір. Наприклад, категорія блоків **Події** стала жовтою, а категорія **Керування** стала помаранчевою. Додаткові блоки **Мої блоки** стали рожевими.
7. Категорії **Олівець**, **Музика** і **Відео** перенесені в *розширення.*



Скретч використовує так звані «розширення», які надають додаткові блоки та функції, які можна використовувати в проектах. Багато цифрових розширень в Scratch 3.0 раніше були звичайними категоріями блоків, які були переміщені до розділу розширень. До них належать:

- ✓ ***Музика*** — грати на цифрових інструментах (барабани, труби, скрипки, фортепіано тощо).
- ✓ ***Олівець*** — малювати на сцені з різними товщинами ліній та кольорами.
- ✓ ***Відеоспостереження*** — виявлення руху за допомогою камери.

Нові цифрові розширення додані у співпраці з комерційними компаніями. До них належать:

- ***Текст у мову*** — перетворює слова з тексту в голосові повідомлення (технологія Amazon)
- ***Перекласти*** — використовує Google Translate для перекладу тексту з однієї мови на інші.

Користувачі також можуть створювати власні розширення для Scratch 3.0 за допомогою JavaScript.

8. Трохи змінився *графічний редактор*. Тепер у векторному режимі всі команди для малювання розташовані зліва, а не справа, як в Scratch 2.0 . Кольорової палітри більше немає, але зате можна вибирати колір за відтінками.

Ще деякі зміни у графічному редакторі:

- новий макет, який робить доступними інструменти і опції більш помітними;
- нові інструменти, такі як «ластик», який працює в векторному режимі;
- розширені можливості пошуку для вибору і настройки кольорів;
- більше контролю над векторними точками (точками кривої і точковими режимами);
- додаткові елементи управління упорядкуванням шарів («вивести на передній план», «перемістити назад» і т. д.);

- нові елементи управління градієнтом.

9. Додано *новий звуковий редактор*, щоб легше записувати і маніпулювати звуками.

Редактор звуку пропонує ряд нових функцій:

- нова система запису, яка простіша у використанні;
- нова система обрізання звуку, яка простіша у використанні;
- нові звукові ефекти (такі як «швидше», «повільніше», «відлуння» і «робот»).

10. З'явилися *нові звуки, фони і спрайти*. Багато з попередніх спрайтів, звуків і фонів, як і раніше, доступні в бібліотеках.

11. За допомогою Scratch 3.0 можна відтворювати проекти Scratch на своєму телефоні, створювати проекти Scratch на планшеті і управляти проектами Scratch своїм голосом. І є багато інших нових функцій.

12. Підтримується той же *набір світових мов* (з поліпшеною підтримкою мов).

13. Всі існуючі проекти і облікові записи продовжують працювати у новій версії; крім того, в оновленій версії доступні:
- нові матеріали для початківців-користувачів;
- нові відео уроки;
- оновлені карти активності Scratch і керівництва для викладачів і батьків;
- оновлені уроки в Code Club і CS First.

Практична робота № 8

Створення презентації про нові версії Scratch

1. Завантаження і вивчення відповідної версії.
2. Пошук інформації для відповіді на запитання.
3. Створення текстового файлу із відповіддю.
4. Створення презентації у Scratch 1.4.
5. Доповідь на задану тему.

Завдання

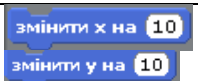
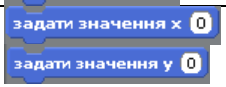
Створити проект ***«(Прізвище) Презентація»*** на запропоновану нижче тему. Повний опис подати у Word, а презентацію – у Scratch 1.4.

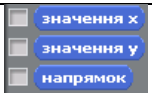
Теми презентацій

1. Категорії блоків та нові блоки у Scratch 2.0, порівняння з Scratch 1.4.
2. Зміни у інтерфейсі головного вікна Scratch 2.0 у порівнянні з Scratch 1.4.
3. Векторний графічний редактор Scratch 2.0.
4. Растровий графічний редактор Scratch 2.0.
5. Вибір, створення та редагування нових образів у Scratch 2.0.
6. Створення та використання звуків у Scratch 2.0.
7. Створення та редагування фонів у Scratch 2.0.
8. Довідкова інформація у Scratch 2.0 і 3.0.
9. Категорії блоків у Scratch 3.0, порівняння з Scratch 1.4 та Scratch 2.0.
10. Зміни у інтерфейсі головного вікна Scratch 3.0 у порівнянні з Scratch 2.0.
11. Зміни графічного редактора у Scratch 3.0 у порівнянні з Scratch 2.0.
12. Розширення в Scratch 3.0. Категорії ***Олівець, Музика*** *і* ***Відео*** в Scratch 3.0, інші розширення.
13. Звуковий редактор у версії Scratch 3.0, порівняння з версією 2.0.
14. Модифікація TurboWarp на базі Scratch 3.0.
15. Мережа клубів Code Club в Україні і світі.
16. Олімпіади та інші змагання по Scratch.

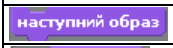
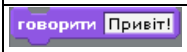
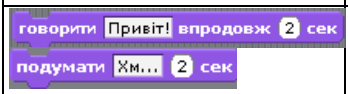
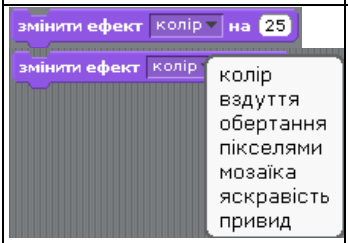
<u>Додаток 1.</u> *Блоки-команди*

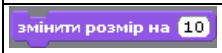
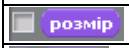
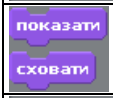
Категорія Рух

Команда	Призначення
	Пройти вказану кількість кроків. Якщо кількість додатна, рухається вперед, якщо від'ємна - назад.
	Повернутися на вказаний кут. Стрілочка вказує, за чи проти годинникової стрілки здійснюється поворот.
	Повернутися в указаному напрямку. Можна вибрати: вверх(0), вниз(180), наліво(-90) чи направо(90).
	Повернутися в напрямку другого спрайта або мишки (вибрати в списку).
	Змінити положення по осі ***x*** або по осі ***y*** на вказане число кроків.
	Установити положення об'єкта по осі ***x*** або ***y***. Використовується декартова система координат. Якщо ***x=0*** і ***y=0*** – об'єкт знаходиться в центрі екрана. Розміри екрана: ***x*** від -240 до 240, ***y*** від -180 до 180.
	Переміститися в точку з вказаними координатами.
	Плавно переміститися в точку з вказаними координатами за вказаний час. Час вказано в секундах.
	Перейти в точку, де розміщений вказівник миші або інший спрайт (за вибором).
	Якщо потрапляєте на край екрана, то відбитись від нього. Це дуже

	корисно, якщо не хочете загубити спрайт.
	Значення координат спрайта по осі ***x*** або ***y*** чи напрямку руху спрайта. Застосовується разом з іншими командами. Відображається на сцені, якщо клацнути мишкою у віконечку.

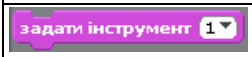
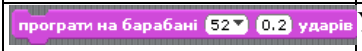
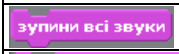
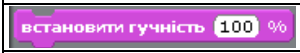
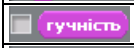
Категорія Вигляд

Команда	Призначення
	Перейти до іншого образу, який вибрати у списку.
	Вибрати наступний образ виконавця.
	Значення номера образу, який в даний момент має наш виконавець.
	Говорити фразу, записану у віконечку команди впродовж усієї роботи скрипту.
	Думати фразу, записану у віконечку команди впродовж усієї роботи скрипту.
	Говорити чи думати записану фразу вказану кількість секунд. Репліка знаходиться поряд з об'єктом. Скрипт призупиняє роботу на вказану кількість секунд.
	Змінити об'єкт по одному з параметрів на вказану величину. Перелік параметрів: ○ *колір* – об'єкт змінює свій колір; ○ *вздуття* – об'єкт стає більш опуклим, якщо число додатне, і більш худим, якщо число від'ємне; ○ *обертання* – об'єкт спотворюється;

	○ *пікселями* – зображення ділиться на малі частинки одного кольору; ○ *мозаїка* – об'єкт розмножується; ○ *яскравість* – об'єкт стає більш чи менш яскравим; ○ *привид* - об'єкт стає більш чи менш прозорим.
	Встановлює вказане значення ефекту вибраного параметра.
	Очистити всі графічні ефекти. Якщо ми здійснювали над об'єктом зміни, то в результаті цієї команди всі вони скасовуються.
	Об'єкт збільшується, якщо число додатне, або зменшується, якщо від'ємне.
	Встановити розмір об'єкта в процентах від поточного.
	Поточний розмір.
	Показати – об'єкт стає видимим. Сховати - він стає невидимим.
	Об'єкт переміщається на перший план, його ніщо не закриває.
	Об'єкт стає позаду вказаної кількості інших об'єктів.

Категорія Звук

Команда	Призначення
	Відтворити вибраний зі списку звук. При цьому звук можна імпортувати з бібліотеки звуків або з файла у форматі ***wav, mp3***. Звук грає і одночасно виконуються наступні команди.

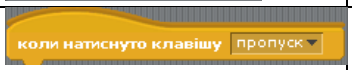
	Спочатку грає звук до кінця, а потому виконуються наступні команди.
	Вибрати інструмент, який буде грати. Інструментів у Скретч 1.4 є більше 120.
	Ударні грають вказану кількість тактів.
	Грати певну ноту вказану кількість часу (в секундах). Ноти записані в цифрах, але напроти кожної цифри стоїть її звучання.
	Зупинити всі звуки.
	Збільшує (якщо число додатне) або зменшує (якщо число від'ємне) поточну гучність.
	Встановлює гучність у процентах.
	Поточне значення гучності.
	Збільшує (якщо число додатне) або зменшує (якщо число від'ємне) поточний темп.
	Встановлює темп.
	Поточне значення темпу.
	Встановлює тривалість паузи.

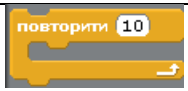
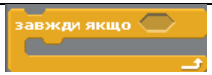
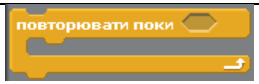
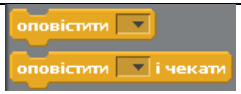
Категорія Олівець

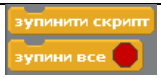
Команда	Призначення
	Очистити екран від усіх слідів, які на ньому залишили об'єкти.
	Опустити олівець. Після цієї команди за об'єктом, що рухається, буде залишатися слід.

	Підняти олівець. При русі об’єкта слід не залишається.
	Вибрати колір для малювання.
	Змінити колір у порівнянні з поточним. Можна використовувати додатні та від’ємні числа. Про нумерацію кольорів читайте у Додатку 2.
	Вибрати числове значення кольору.
	Дозволяє змінити насиченість кольору (0 – найнасиченіший, 100 – найблідіший).
	Встановлює насиченість кольору.
	Змінити розмір олівця по відношенню до поточного.
	Установити розмір олівця.
	Копіює об’єкт на сцені.

Категорія Керувати

Команда	Призначення
	Відбудеться запуск проекту при натисканні на зелений прапорець.
	Відбудеться запуск проекту при натисканні на об’єкт.
	Запускає виконання блока команд у відповідь на натискування вибраної кнопки. Дозволяє передати управління на клавіатуру.
	Запускає виконання блока команд у відповідь на одержане повідомлення.
	Команда очікування. Параметр

	вказує, скільки секунд потрібно чекати.
	Команда нескінченного циклу. Блок команд всередині конструкції буде виконуватися постійно.
	Цикл з лічильником. Параметр вказує, скільки разів потрібно повторити команди, що містяться всередині блока.
	Цикл з передумовою. Дії всередині блоку виконуються доти, поки умова у віконечку правильна.
	Чекати, поки не виконається умова у віконечку.
	Цикл з післяумовою. Дії всередині блоку виконуються доти, поки умова у віконечку не правильна.
	Команда неповного розгалуження. Якщо умова у віконечку виконується, то виконуються команди, що містяться всередині конструкції. Якщо умова не виконується, то ніяких дій не відбувається.
	Якщо умова у віконечку виконується, то виконуються команди, що містяться всередині конструкції *якщо ( )*. Якщо умова не виконується, то виконуються команди, що містяться всередині конструкції *інакше ( )*.
	Передати повідомлення. Передане повідомлення може запускати активність іншого виконавця. Працює разом з командою *коли*

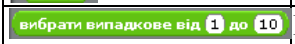
	одержую ( ). При використанні першого блока відправляється повідомлення і виконуються наступні блоки, а у другому очікується реакція.
	Зупинити виконання програми для даного виконавця або всіх програм.

Категорія Датчики

Команда	Призначення
	Значення вказівника миші по осі ***x.***
	Значення вказівника миші по осі ***y.***
	Перевіряє, чи натиснута управляюча клавіша мишки.
	Перевіряє, чи натиснута вказана клавіша.
	Чи доторкається об’єкт до мишки, до межі чи до вибраного спрайта?
	Чи доторкається об’єкт до вибраного кольору?
	Чи доторкається перший вибраний колір до іншого кольору?
	Відстань до вибраного об’єкта чи вказівника мишки.
	Значення координати ***x*** (або координати ***y***, або напрямку, або номера образу, або розміру, або гучності) вибраного об’єкта.
	Значення таймера починає рахуватися від нуля.
	Відображає поточне значення

	таймера.
	Відображає поточне значення гучності.
	Відображає значення вибраного датчика.
	Датчик має значення «істина», якщо на комп'ютері є можливість відображати звуки, та «хибне» в іншому випадку.

Категорія Оператори

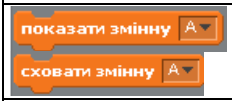
Команда	Призначення
	Додавання.
	Віднімання.
	Множення.
	Ділення.
	Операції порівняння: менше,дорівнює, більше.
	Вибір випадкового числа у межах від () до ().
	Логічні операції перерізу, об'єднання, заперечення. Ці блоки містять вхідні віконечка, куди вставляються відповідні блоки.
	Функція (модуль,квадратний корінь, тригонометричні та обернені тригономентичні функції,логарифмічні та показникові функції)
	Остача від ділення першого числа на друге.
	Округлити число до цілого значення.
	Об'єднати два слова в одне. Замість слова може бути значення змінної (текстове чи

	числове).
	Значення символу із вказаним порядковим номером у слові або числі.
	Кількість символів у слові або числі.

Категорія Змінні

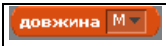
Тут є два блоки:  та  .

Після створення змінної виникають такі блоки:

Команда	Призначення
	Присвоєння змінній вказаного значення.
	Змінити значення змінної на додатне або від'ємне число.
	На сцені з'являється чи зникає монітор з назвою і значенням змінної.
	Блок відображає значення змінної.
	Кнопка дозволяє вилучити створену змінну.

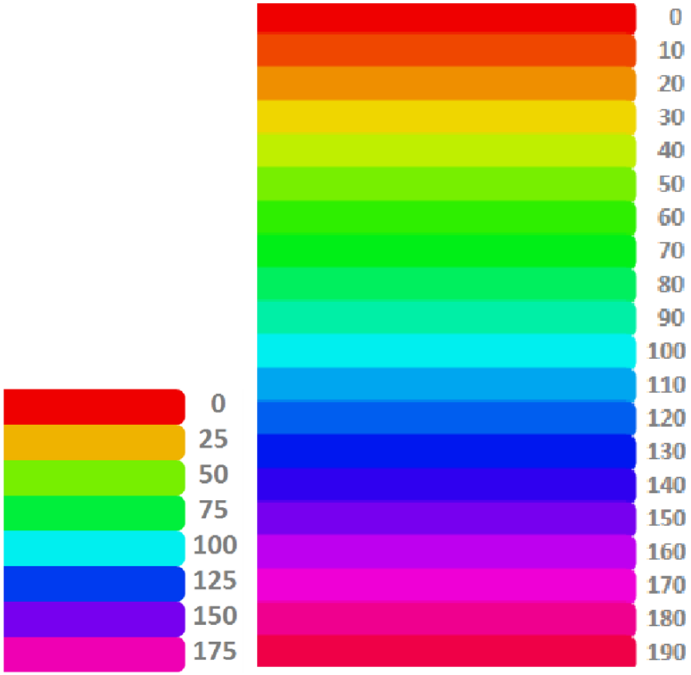
Після створення списку виникають такі блоки:

Команда	Призначення
	Додати зазначений елемент у кінець списку.
	Видалити перший елемент списку (або інший, або останній, або всі).
	Вставити новий елемент на місце першого (або іншого) елемента списку.
	Замінити перший (або інший) елемент списку на новий.
	Відображає всі елементи списку підряд.
	Відображає перший (або інший) елемент списку у інших блоках.

	Відображає число, що дорівнює кількості елементів списку.
	Блок використовується при перевірці умови, чи задане значення є елементом списку.
	Вилучення створеного списку.

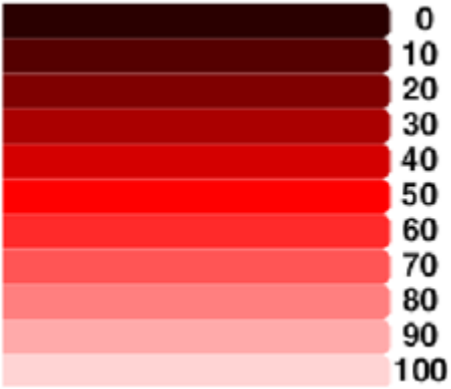
Додаток 2. *Нумерація кольорів*

Нумерація кольорів та їх відтінків використовується при малюванні у блоках категорії Олівець. Кольором з номером нуль є червоний. Колір повторюється через 200. На малюнках наведено таблиці відповідності кольорів та їх номерів з кроком 25 та 10 відповідно.



Всі відтінки кольорів також пронумеровані від нуля (найтемнішого відтінку) до 100 (найсвітлішого відтінку). У

наступній таблиці наведені, як приклад, відтінки та їх номери для червоного кольору (нормальний колір – відтінок 50).



Трохи по-іншому нумеруються кольори при зміні кольору спрайта. Колір спрайта можна змінювати, використовуючи блоки 

та 

з категорії Вигляд. Номер кольору спрайта змінюється від 0 до 200. Якщо значення кольору більше за 200 (наприклад, при 201) спрайт виглядає так само, як і при 1. Початковий колір спрайта відповідає номеру 0, а при зміні ефекту на 25 колір змінюється згідно з попереднім малюнком.

Наприклад, для котика початковий колір оранжевий – він має номер 0. При значенні ефекту 25 кіт стане жовтим, при 50 – зеленим і т.д., при досягненні номера 200 кіт знову буде оранжевим.

Література

1. Офіційний сайт проекту Scratch [Електронний ресурс]. — Режим доступу: http://scratch.mit.edu.
2. Вікіпідручник по Scratch. [Електронний ресурс]. — Режим доступу: http://uk.wikibooks.org.
3. Онлайн-курс «Алгоритми і проекти Scratch» на українській платформі масових відкритих онлайн-курсів “Prometheus” [Електронний ресурс]. — Режим доступу: https://edx.prometheus.org.ua/courses/course-v1:KPI+Scratch 101+2017_T1/about.
4. Шапошнікова С. Введення в Scratch. Цикл уроків по програмуванню для дітей.
5. Рындак В.Г., Дженжер В.О., Денисова Л.В. Проектная деятельность школьника в среде программирования Scratch. Учебно-методическое пособие.— Оренбург, 2009. —116 стр.
6. Патаракин Е.Учимся готовить в Scratch..
7. Голиков Д.В., Голиков А.Д. Программирование на Scratch (в 2 частях).
8. Підручники з інформатики **3 клас**
    1. Корнієнко М.М. Сходинки до інформатики. 3 клас: підруч. для загальноосвіт. навч. закладів / М.М. Корнієнко, С.М. Крамаровська, І.Т. Зарецька. – Х.: Видавництво «Ранок», 2013. – 160 с.: іл..
    2. Ломаковська Г.В. Сходинки до інформатики: підруч. для 3 кл. загальноосвіт. навч. закладів / Г.В. Ломаковська, Г.О. Проценко, Й.Я. Ривкінд, Ф.М. Рівкінд. – К.: Видавничий дім «Освіта», 2013. – 160 с.
    3. Коршунова О.В. Сходинки до інформатики: підруч. для 3-го кл. загальноосвіт. навч. закл. / О.В. Коршунова. – К.: Генеза, 2014. – 176 с.: іл.
9. Підручники з інформатики **4 клас**
    1. Корнієнко М.М. Інформатика: підруч. для 4 класу загальноосвіт. навч. закл. / М.М. Корнієнко, С.М. Крамаровська, І.Т. Зарецька. – Х.: Вид-во «Ранок», 2015. – 160 с.: іл..
    2. Ломаковська Г.В. Інформатика: підруч. для 4 кл. загальноосвіт. навч. закладів / Г.В. Ломаковська, Г.О.

Проценко, Й.Я. Ривкінд, Ф.М. Рівкінд. – К.: Видавничий дім «Освіта», 2015. – 160 с.

3. Коршунова О.В. Інформатики: підруч. для 4-го кл. загальноосвіт. навч. закл. / О.В. Коршунова. – К.: Генеза, 2015. – 176 с.: іл..
4. Левшин М.М. Інформатика: підруч. для 4-го кл. загальноосвіт. навч. закл. / М.М. Левшин, Є.О. Лодатко, В.В. Камишин. – Тернопіль: Навчальна книга – Богдан, 2015. – 192 с.
5. Морзе Н.В. Інформатика: підруч. для 4-го кл. загальноосвіт. навч. закл. / Н.В. Морзе, О.В. Барна, І.О. Большакова, В.П. Вембер. – К.: Видавничий дім «Освіта», 2015. – 192 с.

10. Підручники з інформатики **5 клас**

1. Корнієнко М.М. Інформатика : підруч. для 5 кл. закл. загал. серед. освіти / М. М. Корнієнко, С. М. Крамаровська, І. Т. Зарецька. — Харків : Вид-во «Ранок», 2018. — 144 с.
2. Ривкінд Й.Я. Інформатика : підруч. для 5 кл. закл. загал. серед. освіти /Й.Я. Ривкінд та ін. — Київ: Генеза, 2018. — 208 с.: іл.
3. Коршунова О.В. Інформатика : підруч. для 5 кл. закладів загальної середньої освіти / О.В. Коршунова, І.О. Завадський. – К.: Видавничий дім «Освіта», 2018. – 144 с.
4. Бондаренко О.О. Інформатика : підруч. для 5 кл. закл. загал. серед. освіти / О.О. Бондаренко, В.В. Ластовецький, О.П. Пилипчук, Є.А. Шестопалов. – Харків: Вид-во «Ранок», 2018. — 160 с.
5. Морзе Н.В. Підручник з інформатики для 5 кл. закладів загальної середньої освіти / Н.В. Морзе, В.П. Вембер,О.В. Барна, О.Г. Кузьмінська. — К.: УОВЦ «Оріон», 2018. – 256 с.: іл.

11. Підручники з інформатики **6 клас**

1. Інформатика: підруч. Для 6-го кл. загальноосвіт. навч. закл. / Й.Я. Ривкінд [та ін.] – Київ: Генеза, 2019. – 128 с.: іл.

2. Морзе Н.В. Підручник з інформатики для 6 кл. закладів загальної середньої освіти / Н.В. Морзе, О.В. Барна, В.П. Вембер. — К.: УОВЦ «Оріон», 2019. – 192 с.: іл.
3. Коршунова О.В. Інформатика : підруч. для 6 кл. закладів загальної середньої освіти / О.В. Коршунова, І.О. Завадський. – К.: Видавничий дім «Освіта», 2019. – 144 с.: іл.
4. Бондаренко О.О. Інформатика: підручник для 6 класу ЗНЗ / О.О. Бондаренко, В.В. Ластовецький, О.П. Пилипчук, Є.А. Шестопалов. – К.: «Аспект», 2017.

12. Підручники з інформатики **7 клас**

1. Інформатика : підруч. для 7-го кл. загальноосвіт. навч. закл. /Й.Я. Ривкінд [та ін.]. — Київ: Генеза, 2015. — 240 с.: іл.
2. Морзе Н.В. Інформатика: підруч. для 7 кл. . загальноосвіт. навч. закладів / Н.В. Морзе, О.В. Барна, В.П. Вембер, О.Г. Кузьмінська. — К.: Видавничий дім «Освіта», 2015. – 224 с.
3. Казанцева О.П. Інформатика : підручник для 7 кл. загальноосвіт. навч. закл. / О.П. Казанцева, І.В. Стеценко, Л.В. Фурик. – Тернопіль: Навчальна книга – Богдан, 2015. – 176 с.: іл.
4. Пилипчук О.П. Інформатика : підручник для 7 класу загальноосвітніх навчальних закладів / О.П. Пилипчук, Н.А. Ріпко, Є.А. Шестопалов. – Шепетівка: «Аспект», 2015. – 108 с.
5. Інформатика : підруч. для 7 кл. загальноосв. навч. закл. /А.М. Гуржій, Л.А. Карташова, В.В. Лапінський, В.Д. Руденко. — Львів: Світ, 2015. — 176 с.: іл., табл.