Міністерство освіти і науки України
Сумський державний педагогічний університет імені А.С. Макаренка
Фізико-математичний факультет
Кафедра інформатики

УДК 378.016:51]:004

Сасіна Юлія Олександрівна

МЕТОДИЧНІ ОСОБЛИВОСТІ НАВЧАННЯ УЧНІВ 5-9 КЛАСІВ ВІЗУАЛЬНОГО ПРОГРАМУВАННЯ

Галузь знань 01 Освіта/Педагогіка
Спеціальність 014 Середня освіта (Інформатика)
Кваліфікаційна робота на здобуття освітнього рівня «Магістр»

Науковий керівник:
_____ Н.В. Дегтярьова
кандидат педагогічних наук, доцент

Виконавець:
_____ Ю.О. Сасіна

Суми – 2021

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

МП – мова програмування

ВМП – візуальна мова програмування

ВГ – візуальна граматика

АМ – алгоритмічне мислення

ІКТ – інформаційно-комунікаційні технології

ВСТУП

Актуальність. Сучасний етап розвитку суспільства характеризується впровадженням інформаційних технологій в різні галузі людської діяльності. Новітні інформаційні технології спрямовують суттєвий вплив і на галузь освіти. Здійснюються детальні зміни в системі освіти, які викликані новими принципами цілей та освітніх цінностей, а також необхідністю застосування нових технологій навчання. Визначною складовою інтелектуального розвитку дитини є алгоритмічне мислення.

Досі питання навчання школярів старших класів програмуванню здавалася майже не реальною. В першу чергу все через відсутність інструменту, який, з одного боку, був би достатньо простим при використані, а з іншого боку дозволив би створювати справді якісні продукти.

Складні для розуміння програми зробили предмет «інформатика» доступним та зрозумілим лише для вузького колу учнів. Для більшої кількості інших школярів ця галузь так і залишилася непростою.

Відмінність App Inventor полягає в тому, що ця програма для створення додатків для мобільних пристроїв або планшетів з операційною системою Android, а не орієнтована на десктопне вживання.

App Inventor є повністю хмарним додатком. Для початку роботи на ньому, потрібен лише інтернет та браузер. Він вміє «розуміти» дані швидкості мобільного пристрою бачить, як орієнтований телефон в просторі, керувати вбудованою камерою і багато іншого.

Об’єкт дослідження: візуальне програмування.

Предмет дослідження: методичні особливості навчання учнів роботи з App Inventor при створенні Android-додатків учнями ЗЗСО.

Мета: дослідити методичні особливості навчання учнів старших класів візуальному програмуванню за допомогою середовища MIT App Inventor.

Завдання:

1. Проаналізувати наявні матеріали щодо візуального програмування.

2. Схарактеризувати середовище візуального програмування MIT App Inventor.

3. Розробити авторські матеріали для учнів 5-9 класів.

4. Перевірити доцільність впровадження таких завдань у освітній процес.

Методи дослідження: *теоретичні*: аналіз наукової, навчально-методичної, психолого-педагогічної літератури для досліджування візуального програмування; аналіз нормативних документів у сфері освіти, навчальних програм, підручників, узагальнення педагогічного досвіду навчання програмуванню в закладах загальної середньої освіти та вищої освіти; *емпіричні*: спостереження, тестування, анкетування, бесіди з вчителями, викладачами, учнями та студентами для виявлення рівня сформованості цифрових компетентностей з програмування, педагогічне дослідження.

Апробація результатів дослідження: результати роботи опубліковані на міжнародній науково-практичній конференції “Modern problems in science” (Прага, Чехія, 09-12 листопада 2020); Дегтярьова Н.В., Мигаль В.О., Сасіна Ю.О. Особливості навчання візуальному програмуванню учнів старших класів в середовищі MIT App Inventor. Фізико-математична освіта. 2020. Випуск 2(24). Частина 2. С. 16-20.

Теоретичну значущість. Основу дослідження становить положення загально дидактичної теорії навчання, основні положення особистісно-орієнтованого підходу в освіті, положення теорії та методики навчання інформатики.

Практична значущість. Являє собою розроблені матеріали та завдання до теоретичних та практичних задач, для навчання візуальному програмуванню учнів старших класів.

Структура роботи: дана робота складається зі вступу, двох розділів, висновків, списку використаних джерел. Загальний обсяг роботи розміщено на 56 сторінок.

РОЗДІЛ І. ВІЗУАЛЬНЕ ПРОГРАМУВАННЯ НА СУЧАСНОМУ ЕТАПІ

1.1. Розвиток алгоритмічного мислення

Одне з найголовніших дидактичних завдань освіти є інтелектуальний розвиток учня, важливою складовою якого є алгоритмічне мислення.

Найбільшим потенціалом для створення алгоритмічного мислення здобувачів освіти посеред природничих дисциплін має інформатика – це одна з основних сфер наукового знання, що генерує системно-інформаційний підхід до синтезу навколишнього світу, який вивчає інформаційні процеси, методи і засоби передачі, зберігання, перетворення, отримання і використання інформації.

Загалом роль викладання інформатиці в розвитку мислення зумовлена новітніми розробленнями в області методики проектування та моделювання, найбільше в моделюванні об'єктно-орієнтованому. Здатність для кожної предметної області виокремити систему понять, описати алгоритми дій, зобразити їх у вигляді групи атрибутів та дій і схеми послідовного висновку. Тобто те, що здійснюється при інформаційно-логічному моделюванні, вдосконалює орієнтацію людини в даній предметній області та демонструє її розвинене мислення.

Наприклад, в процесі вивчення теми «Алгоритмізація та програмування» учні повинні вміти розробляти план рішення задачі, висувати і доводити гіпотези, прогнозувати результати рішення, аналізувати і знаходити найкращі засоби та інше. Ці розумові вміння характеризують рівень розвитку алгоритмічного мислення.

Алгоритмічним мисленням нахивають пізнавальний процес, який зумовлюється наявністю чіткої, доцільною послідовності здійснюваних розумових процесів з властивою деталізацією і оптимізацією збільшених блоків, усвідомленим закріпленням процесу отримання кінцевого результату, представленого в формалізованому вигляді на мові виконавця з прийнятими семантичними і синтаксичними правилами.

Під спроможністю алгоритмічно міркувати розуміється здатність розв'язувати завдання несхожого походження, яке вимагає розробки плану дій для здобутку бажаного результату.

Оскільки алгоритмічне мислення протягом життя удосконалюватиметься під дією зовнішніх чинників, то в процесі додаткового впливу можливе підвищення рівня його розвитку. Необхідність пошуку нових ефективних прийомів розвитку алгоритмічного мислення у здобувачів освіти обумовлена його значимістю для подальшої самореалізації особистості в інформаційному суспільстві.

АМ має свої загальні та специфічні властивості в порівнянні з іншими стилями мислення. У число загальних властивостей алгоритмічного мислення входять цілісність і результативність, що допомагають побачити поставлену проблему в цілому вигляді і передбачають створення попереднього способу результату вирішення поставленої проблеми. До специфічних властивостей відносяться дискретність, абстрактність і усвідомлена закріплених в мовних формах. Ці властивості є покрокове виконання, виконання алгоритму, дають можливість абстрагуватися від точних вихідних даних, перейти до вирішення завдання в загальному вигляді та зробити алгоритм завдяки деякій формалізованій мові. Компонентами алгоритмічного мислення є вміння формалізувати задачу і розбивати її на окремі складові логічні блоки.

Алгоритмічне мислення відзначається черговими компонентами:

1. Аналіз потрібного результату та вибір за допомогою даної основи вихідних даних для вирішення проблеми;

2. Акцентування операцій, необхідних для вирішення;

3. Вибір виконувача, спроможного реалізовувати ці операції;

4. Впорядкування операцій та побудова моделі процесу рішення;

5. Здійснення процесу рішення і співставлення результатів з тим, що варто було отримати в кінці;

6. Коректування вихідних даних чи системи операцій в разі розбіжності одержаного наслідку з передбачуваним.

Уміння розбивати завдання на частини вважають структурним стилем мислення. Особливості зазначеного стилю: простота і ясність; використання тільки базових (основних) конструкцій; відсутність багатоцільових функціональних блоків і т. д. Відзначимо, що комп'ютер, система програмування не є метою навчання, вони - інструмент реалізації цілей, хоча при цьому пізнається і сам інструмент.

З інформатики в методичній літературі відзначено багато різних засобів формування АМ здобувачів освіти:

- проведення систематичного і спрямованого застосування принципів структурного підходу (А. Гейн, В.Н. Ісаков, В.В. Ісакова, В.Ф. Шолоховіч);
- підвищення рівня мотивованості завдань (В. Ісаков, В.В. Ісакова);
- постійна розумова робота (Я.М. Зайдельман, Г.В. Лебедєв, Л.Е. Самовольнова).

В.В. Левитес запропонував певну систему завдань та прийомів для персональної роботи з здобувачами освіти з розвитку алгоритмічного та логічного мислення. Метою даної системи є формування і розвиток простих логічних дій або прийомів розумової діяльності завдяки використанню логічного конструювання на образному математичному матеріалі за допомогою безпосередньо предметної діяльність з речовим матеріалом: конструктивну діяльність з моделями фігур, конструктивно-графічну - з використанням спеціальної рамки-трафарету з геометричними прорізами, логіко-графічну, яка супроводжує вирішення всіх запропонованих завдань.

А.І. Газейкіна виділяє такий методичний прийом, який розрахований на розвиток алгоритмічного мислення:

1. Створення нового алгоритму, його запис, перевірка і виконання самим учнем або обраним виконавцем;

2. Засвоєння алгоритмів вирішення основних типових задач;

3. Пошук і виправлення синтаксичних і семантичних помилок в алгоритмі;

4. Оптимізація готового алгоритму, тобто його спрощення і поліпшення.

Для кожного етапу з урахуванням психолого-фізіологічних вікових здібностей визначається мінімально можливий рівень кожного з основних елементів алгоритмічного мислення і максимальний рівень. У цих межах для кожного етапу виділяються інформаційний і практичний блоки. В інформаційний блок входять всі знання, якими повинен опанувати школяр після проходження етапу. Практичний блок являє собою систему завдань прикладного характеру, які виконуються школярам на даному етапі.

І.М. Слинкина пропонує використовувати програмні комплекси на основі якісної зміни можливостей сучасних ЕОМ, а також такі комплекси дій, за допомогою яких реалізується алгоритмічне мислення:

1. Створення нового алгоритму, його перевірка і виконання.
2. Проведення синтаксичного аналізу тексту.
3. Трасування алгоритму.
4. Виконання готового алгоритму.
5. Оптимізація готового алгоритму.

А.Г. Гейн вважає в розвитку мислення важливим етапом освоєння системи базових знань, що відображають внесок інформатики в формування наукової картини всесвіту, роль інформаційних процесів в соціальних, біологічних і технічних системах.

У роботах були визначені три основні рівні розвитку алгоритмічного мислення: операційний (володіє деякими розрізненими операціями, але не може поєднувати їх, не володіє структурою їх вкладеності), системний (знає деякі способи поєднання операцій конструкцій створення цих поєднань, вміє вирішувати стандартні завдання на застосування алгоритмічного мислення), методологічний (вміє використовувати вже наявні розумові схеми вирішення деяких алгоритмічних проблем, може перетворити їх в умовах, що змінюються або трансформувати наявні).

Відповідно до даних рівнях були виділені вміння, що характеризують кожен етап розвитку АМ:

- розв'язувати задачі алгоритмічного характеру;

- робити аналіз задачі;
- створювати алгоритм;
- записувати алгоритм;
- виробляти синтаксичний аналіз створеного чи запропонованого алгоритму;
- втілювати алгоритми;
- проводити оптимізацію алгоритму;
- виробляти розумові операції.

На основі цих рівнів виділяються вимоги до розвитку АМ:

• Операційний рівень характеризується тим, що кожен здобувач освіти має певне уявлення про алгоритм.

• Системний рівень характеризується тим, що учень має уявлення про алгоритм, його властивості, становить невеликі лінійні алгоритми або з найпростішими розгалуження та циклом; володіє конкретними операціями класифікації; знає засоби рішення конкретного класу алгоритмічних задач; має бачення про виконавця та систему команд виконавця.

• Методологічний рівень характеризується тим, що учень має уявлення про алгоритм, ознайомлений з його властивостями, може складати та записувати формальні і неформальні алгоритми лінійної структури, з найелементарнішими розгалуження та циклами; має бачення про виконавця, системі команд виконавця.

Основні принципи побудови навчання, спрямованого на розвиток алгоритмічного мислення зводяться до наступних: систематичність роботи, спрямованої на розвиток алгоритмічного мислення; системність, повнота та всебічність розгляду окремих дій, що входять в структуру алгоритмічного мислення; можливість співвіднесення отриманих результатів з еталоном. Для виконання всіх цих умов доцільно і необхідно використання ПК.

1.2 Візуальне програмування

Хід створення багатьох проектів включає в себе побудову візуальних, системних і геометричних з'єднань між усіма необхідними елементами. У більшості випадків переважне створення даних сполучень застосовується за допомогою робочих процесів, у яких перехід від концепції до кінцевого результату створюється завдяки застосуванню правил. Таким чином, хід роботи (можливо, ненавмисно) генерується за алгоритмічним правилом, в основі якого є покроковий набір дій, що відтворює шаблонну логіку: уведення даних, їх обробка і, нарешті, висновок. Інструменти програмування дозволяють формалізувати дані алгоритмічні процеси та підвищити ефективність їх застосовування.

Візуальним програмуванням називають вид програмування, яке призначене для створення програм на основі наочних засобів, тобто на основі оперуванням графічними об'єктами, а не написанням програмного коду в текстовому виді [1; 2]. Такий різновид програмування часто репрезентують як наступну стадію розробки програмування текстових мов. Підґрунтям такої роботи є конструкція розв'язання необхідної задачі використовуючи візуальні заготовки, які включатимуться у форму, надання значень їхнім атрибутам і виготовлення чи вживання необхідних для рішення даної методів задачі. Останніми роками почало надаватися більше уваги ВМ у зв'язку з розвитком смартфонів та планшетів. Користуються цим щоб створити програми які містять графічний інтерфейс для операційних систем з графічним інтерфейсом користувача [3].

Фронтальною метою ВМП є перетворення програмування в щось набагато доступніше для початківців і підтримати програмістів на 3 різних рівнях:

– ВМП користуються значками/блоками, формами та діаграмами, намагаючись зменшити або, зовсім усунути потенційні синтаксичні похибки,

виручаючи з розміщенням примітивів програмування для виробництва досить непогано сформованих програм.

– ВМП мають змогу приділити механізми для розкриття значення примітивів програмування. Довідкові функції допомагають зрозуміти інформацію щодо елементів, які є вбудованими в мову програмування.

– ВМП підтримують дослідження практичних питань програмування, в тому числі оцінку роботи програми в контексті точних ситуацій. Такий рівень опори дає можливість користувачам розміщати об'єкти, створені за допомогою ВМП, в певний стан, щоб дослідити, як програма буде реагувати на цей стан [4; 9].

Осередок розробки ВМП неодноразово об'єднується з спеціалізованим середовищем реалізації, тому всі користувачі мають змогу без проблем і блискавично запускати свою програму і бачити результати. Це не обов'язково безпосередньо пов'язано з самою ВМП, проте даний фактор є важливим і це спрощує запуск першої програми.

Також ВП дає можливість спростити бачення тотальної картини. Програма визначається її формою. Кольори та форми теж застосовують візуальне сприйняття користувача більшою мірою, ніж відступи і забарвлення коду, щоб створити вихідний код більше читабельним. Це допомагає краще описати програму.

Варто розпізнавати:

- ВМП це по-перше мова програмування, яка має власний синтаксис;
- ВЗ розробки найчастіше говорять, що це засоби проектування інтерфейсів або якусь CASE-систему для швидкої розробки додатків або SCADA-систему для програмування мікроконтролерів.

Є складності для сприймання ВМП загальними користувачами:

- ВМ потребують точності, конкретного формування потоку керування;
- Всі типи програмування вимагають від користувачів знайомства з загальними, а також специфічними для мови програмування концепціями;

- користувачам, які ознайомлені з синтаксисом мови програмування та зобов'язані працювати з багатьма бібліотеками чи різнорідними компонентами, переваги ВМП для виявлення синтаксису не є сильно цікавими. Однак як з точки зору представлення виразів текст є дуже компактним водночас і відкритим;
- не зважаючи на те, що деякі ВМП поєднують текст та форми, їх візуальні зображення зазвичай не можуть конкурувати з інформаційною щільністю тексту;
- осередки розробки ВМП деколи є спеціалізованими, пристосованими до обмеженої арени (напр. ігровий дизайн);
- також ВМП мають технічні дефекти, приміром у швидкості їх виконанні.

Мови ВП можуть бути ще раз класифіковані залежно від типу та ступеня візуального виразу на такі типажі:

- мови на основі об'єктів, коли середовище ВП дає графічні або ж символьні елементи, за допомогою яких є можливість керувати інтерактивним засобом відповідно до певних правил;
- мови, в інтегрованому середовищі розробки яких етапи проектування інтерфейсу застосовуються форми, з шансом налаштуванням їх можливостей. Приклади: Delphi, C# та C++ Builder фірми Borland.
- мови схем, основані на ідеї «фігур і ліній», де постаті (прямокутники, овали тощо) розглядаються як суб'єкти та з'єднуються лініями (стрілками, дугами та ін.), які є відносинами. Приклад: UML.

У нинішніх розробках здійснюють спроби інтегрувати підхід ВП з програмуванням потоків даних (англ. dataflow programming), щоб мати прямий доступ до стану програми для налагодження онлайн, або автоматизована генерація і документування програми. Мови потоків даних теж дають змогу робити автоматичний розподіл, який може стати одним із найбільших досягнень програмування в майбутньому.

У ряді робіт підхід ВП пов'язаний з програмуванням потоків даних (англ. dataflow programming). Деякі засоби візуального програмування підтримують налагодження програм, автоматизовану генерацію та документування. Мови потоків даних можуть дозволяти автоматичне розпаралелювання, що може стати великим досягненням програмування.

При цьому можна назвати недоліки візуального програмування. Стаття Майка Хедлоу говорить про фундаментальні обмеження візуального програмування:

- Обмеження візуального інтерфейсу можуть заплутувати розробника навіть більше, ніж текст.
- З підвищенням складності програм програміст починає займатися абстракцією та зниженням зв'язності, і рівень програміста багато в чому визначається тим, наскільки вдало це вийшло. Візуальні засоби рідко мають розвинену підтримку цього процесу.
- Для текстового представлення в даний час існує безліч інструментів: системи керування версіями, авто доповнення та ін.

1.3 Програмування в школі

Основою буму довкола уведення раннього програмування до шкільних програм з’явилася не через необхідності створити просто групу кваліфікованих програмістів для латання «дірок» на ринку праці. Безперечно, відповідність рівня освіти його потребам – ґрунтовний аспект і має глибоке підґрунтя, але процес зайшов значно далі й поступово обріс новими, не менш важливими пріоритетами та цілями.

Велика кількість досліджень підтвердили, що для здобувачів освіти є дуже важливим – вміти працювати з ІКТ з раннього віку, як в освітньому, так і в соціальному плані. Експерти вважають, що дійсно навчання програмуванню дає шанс школярам розвивати свої власні творчості здібності за допомогою використовування цифрових технологій. Окрім цього, в технологічному суспільстві це допомагає учням поступово перейти від ролі «споживача» до

ролі «творця», а також розвинути настільки важливе сьогодні алгоритмічне мислення, що надає краще розуміння, інтерпретацію та оцінку впливу такого мислення на життя людини.

На пристроях, які зазвичай мають школи, складно викладати щось сучасне, тому що воно банально не запуститься. У результаті практична робота перетворюється на муку та перевірку терпіння учнів. Можна, звичайно, поставити якийсь легкий Linux і налаштувати його для необхідних завдань, але найчастіше такої кваліфікації учитель не має.

Проте програмування навчає всіх дітей приймати виважені рішення, бути незалежною та рішучою у світі, де технології повсюдні, допомагає дітям значніше відчути специфіку та сутність самого цифрового світу, в якому всі люди проживають, та в певному сенсі, краще підготуватися до нього.

Об'єднавши ключові навички, переваги та компетенції, яких набувають здобувачі освіти у процесі вивчення програмування, можна отримати місткий список найактуальніших вимог до будь-якої освітньої системи.

Тому, основні переваги вивчення програмування у школі – це:

1. Якісний розвиток індивідуального творчого потенціалу;
2. Розвиток критичного й логічного мислення;
3. Посилення мотивації навчання;
4. Формування навичок застосування технологій в рамках базових шкільних дисциплін;
5. Вміння вирішувати проблеми різної складності – самостійно та колективно;
6. Розширення витривалих комунікацій і вміння працювати в команді;
7. Придбання та закріплення навичок взаємодіяти з ІКТ на рівні впевненого користувача;
8. Виявлення унікальних здібностей дитини та можливість спрямувати їх у потрібне русло;

9. Формування та поліпшення самооцінки й почуття власної компетентності;

10. Здатність працювати з потрібною інформацією та використовувати її з освітньою метою;

11. Стійкість до проблем та труднощів;

12. Формування навичок самоорганізації та планування часу.

На сьогодні практично всі викладачі можуть забезпечити цими перевагами на благо кожного учня. Потрібно тільки використати достатньо прості у використанні й повсюдно доступні інструменти у вигляді навчальних онлайн платформ і додатків.

Колись найбільш популярними мовами програмування у школах світу були Бейсік та Паскаль. Бейсик завжди вважався найпростішою мовою програмування, а Паскаль — найпридатнішою для навчання програмуванню. Але тепер це негаразд. Так, Бейсік простий. Але він створювався в часи, коли людство не мало жодного досвіду створення комп'ютерних систем, і заснований на застарілих принципах, що не виправдали себе. Власне, жодної фундаментальної цілісної ідеї в основі Бейсіку не лежить.

Сьогодні є прості і при цьому наочніше й ідейно замкнуті мови програмування, ніж Бейсік. Паскаль зручний у навчальних цілях; адже для них він і створювався. Здобувачі освіти швидко вчаться вирішувати з його допомогою алгоритмічні завдання. Але так виходить, що вивчати Паскаль корисно лише для того, щоб писати програми на Паскалі. А якщо потрібно створити справжній програмний продукт, Паскаль виявляється незручним. І студентам, які знають лише Паскаль, доводиться переучуватися, що часто складніше, ніж вивчити правильні мови та технології з нуля.

Учні 5-6 класів після освоєння Scratch готові приступити до вивчення «дорослих мов» програмування. З ними вони працюватимуть у майбутньому, якщо вирішать професійно розвиватися у IT-сфері.

Python це перша мова програмування з невізуальних, яку під силу освоїти учням середньої школи. Python простий у вивченні, тому що має простий синтаксис та мінімум службових символів. Відмінний бонус у тому, що його можна вивчати в ігровій формі. Хлопці навчаються програмувати у просторі улюблених всесвітів, наприклад у Minecraft або Turtle. Вони тепер не просто гравці, а й творці гри, можуть просунутися до найвищого рівня та керувати тим, що відбувається.

Учні шостого класу впораються з більш серйозною мовою HTML/CSS. За його допомогою школяр зможе створити свій перший сайт. Знання HTML дозволить вам написати структуру сайту, а CSS призначений для опису зовнішнього вигляду веб-сторінок. HTML/CSS часто зацікавлює учня настільки, що він починає самостійно заглиблюватися в нюанси веб-дизайну.

Потім можна переходити до мови програмування зі складнішим синтаксисом - JavaScript. Після того, як школяр освоїв візуальні мови та Python, зрозуміти структуру цієї мови буде нескладно. Він включає багато службових символів та конструкції з різними дужками. При цьому для роботи з JavaScript потрібно лише браузер або текстовий редактор, який підтримує цю мову програмування. Діти створюють справжні ігри та легко освоюють програмування в ігровій формі.

App Inventor це середовище, яке призначене для візуальної розробки мобільних android-додатків. App Inventor можна освоювати із п'ятого класу. Ця мова не потребує глибоких знань у програмуванні. Побудова програм провадиться у візуальному режимі з використанням блоків програмного коду. Програма розроблена в Google Labs, а потім передана Массачусетському технологічному інституту.

1.4 Види мов програмування. Сучасні візуальні мови програмування для учнів

На сьогоднішній день існує понад 80 мов програмування [1]. У різноманітних джерелах відзначаються різні класифікації мов програмування.

Зокрема, їх класифікують за функціональністю, способом реалізації тощо. За типізацією – процесом перевірки відповідності типів, тобто набору правил за якими присвоюють властивості змінним, виразам, функціям тощо, розподіляють статистичні та динамічні мови програмування. За рівнем звернення та відповідності до машинного коду диференціюють низько рівневі та високо рівневі мови програмування (МП). Останнім часом є тенденція щодо виокремлення надвисокорівненвих мов з високим рівнем абстракції. За аналогічним вказаному вище виміром розрізнюють інтерпретовані та компільовані МП. І також, за способом дії виокремлюють текстові та візуальні мови програмування.

Scratch та його похідні вважаються великою кількістю користувачів, як ВМП. Вона представляє практично граматику класичної імперативної МП, але графічні підказки допомагають зрозуміти, як з'єднувати головні елементи, такі як змінні, умови або регулятори потоку. За допомогою налаштованих блоків є змога розширити мову, які теж можуть бути визначені за допомогою Scratch.

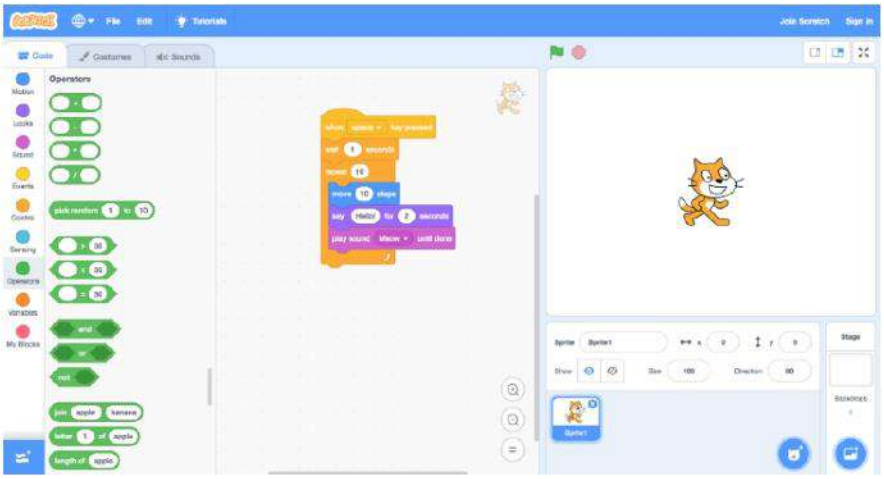


Рис. 1.1. ВМП «Scratch».

Іншим середовищем є ВМП «Кибор». Виходячи безпосередньо з ідеї застосовування «стрілок і прямокутників» для опису програм, ця ВМП застосовує візуальне представлення, яке є близьким до блок-схем для опису головного потоку управління. Вони показують собою цілеспрямовану послідовність виконання між блоками, з потоком, який проходить через блок

до наступного, часто з розгалуженнями, які використовують результат/вихід блоку, щоб вибрати, який блок виконати наступним.

ВМП на основі блок-схем (рис. 1.2).

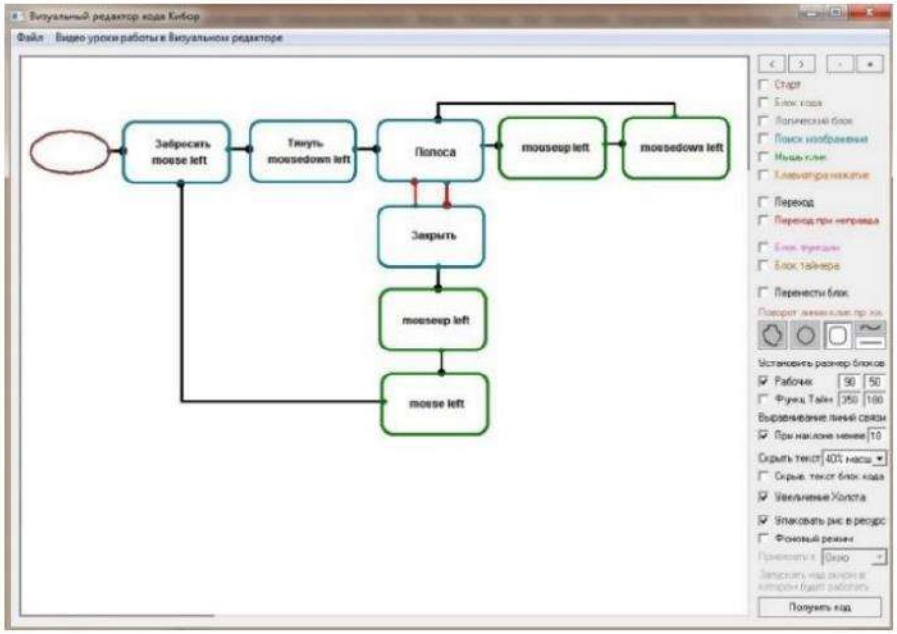


Рис. 1.2. ВМП «Кибор»

Даний фокус на простій та виконанні візуальній граматиці важитиме, що формат нескладний для розуміння, отже вияв синтаксису не є якоюсь складною справою. Проте логічні конструкції, створені просто через такі ВМП, обмежені, багато що залежить від того, що перебуває всередині блоків, що часто є заданим і не може бути змінено з графічного інтерфейсу.

Деякі формати, подібні блок-схемам, користуються розширеною граматикою, яка дає можливість створювати більш складні інструкції безпосередньо з графічного інтерфейсу.

Програмування потоку даних (рис. 1.3).

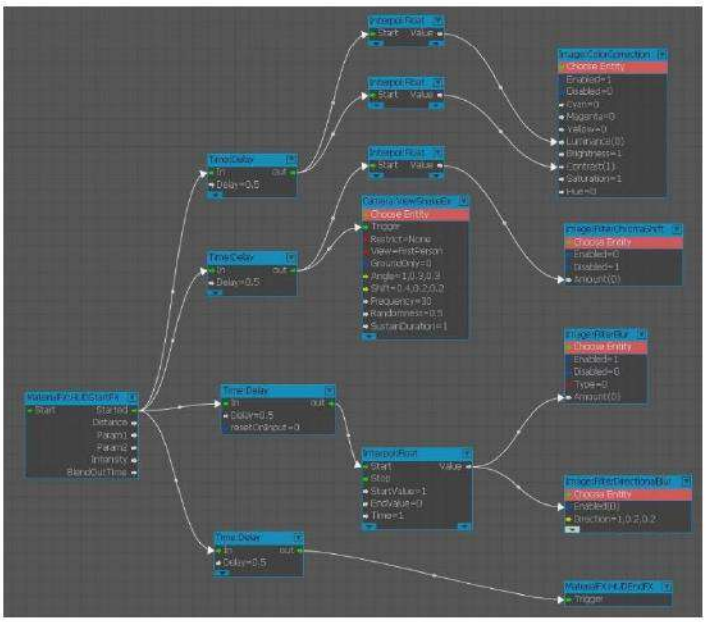


Рис. 1.3. ВМП «CryEngine flow graph»

Такий формат є одним з найбільш розповсюджених ВМП для професійних додатків, орієнтованих на дизайнерів, а не на кінцевих користувачів або початківців програмістів.

При роботі з програмуванням потоку даних блоки являють собою функції, і вони пов’язані один з одним уздовж потоку даних, уздовж входу і виходу. За допомогою зв’язку між вихідним маркером одного блоку і вхідним маркером іншого блоку, користувач з'ясовує потік виконання програми через потік даних [5; 10].

Завдяки цьому, візуальна граматика (ВГ) досить проста, користувач в основному вибирає, які блоки використовувати і як їх зв’язати. Це означає, що більша частина того, що визначає програму, насправді знаходиться всередині блоків, які зазвичай пишуться на класичній мові програмування.

РОЗДІЛ ІІ. МЕТОДИЧНІ ОСОБЛИВОСТІ НАВЧАННЯ ВІЗУАЛЬНОМУ ПРОГРАМУВАННЮ УЧНІВ СТАРШИХ КЛАСІВ

2.1 Середовище MIT App Inventor

MIT App Inventor - це середовище для створення веб-додатків. Спершу дана платформа була створена компанією Google Labs, але тепер підтримується Massachusetts Institute of Technology. MIT App Inventor безоплатне програмне забезпечення, яке має відкритий код.

Дане середовище користується графічним інтерфейсом (GUI), вкрай подібний на мови програмування StarLogo і Scratch, який дає змогу користувачам перетягувати візуальні об'єкти для виробництва різноманітних програм. Ці програми призначені для пристроїв Android [6-8; 12]. App Inventor підтримує застосування хмарних даних за підтримкою дослідницького компонента бази даних Firebase # Firebase Realtime.

App Inventor заснований на конструктивістських теоріях навчання, які акцентують, що програмування є способом притягнення потужних ідей через активне навчання [13].

Історія створення. Заявку для створення даного програмного середовища було надіслано в липні 2010 року і теж була опублікована в грудні того ж року. Очолювали команду творців App Inventor професор Хел Абельсон та професор Марк Фрідман. У другій половині 2011 року Google Labs випустила вихідний код, припинила роботу свого сервера і надала фінансування для створення центру мобільного навчання Массачусетському технологічному інституту, очолюваного Хелом Абельсоном та іншими професорами MTI Еріком Клопфером і Мітчелом Резніком. Версія MTI була загалом запущена в березні 2012 року.

У грудні 2013 року був представлений App Inventor 2 від MIT, перейменувавши оригінальну версію середовища «App Inventor Classic». Основні розрізнення полягають в тому, що редактор блоків в автентичній

версії виконувався в ізольованому Java-процесі, застосовуючи бібліотеку Open Blocks Java для створення блоків ВМП.

Open Blocks розходяться програмою Scheller Teacher Education Program (STEP) Массачусетського технологічного інституту і є похідними від магістерських дисертаційних досліджень Рікарози Роке. Професор Ерік Клопфер і Даніель Вендель з програми Шеллера підтримали поширення Open Blocks під ліцензією MTI. ВП відкритих блоків щільно пов'язане з StarLogo TNG, проектом STEP, і Scratch, проектом MTI Media Lab Lifelong Kindergarten Group на чолі з Мітчелом Резніком.

Інтерфейс середовища. Інтерфейс людини, яка використовує дане середовище має два основних редактора: редактор блоків та редактор дизайну. Редактор дизайну, або конструктор (рис. 2.1), призначений для перетягування та розстановки елементів користувальницького інтерфейсу даної програми (User Interface).

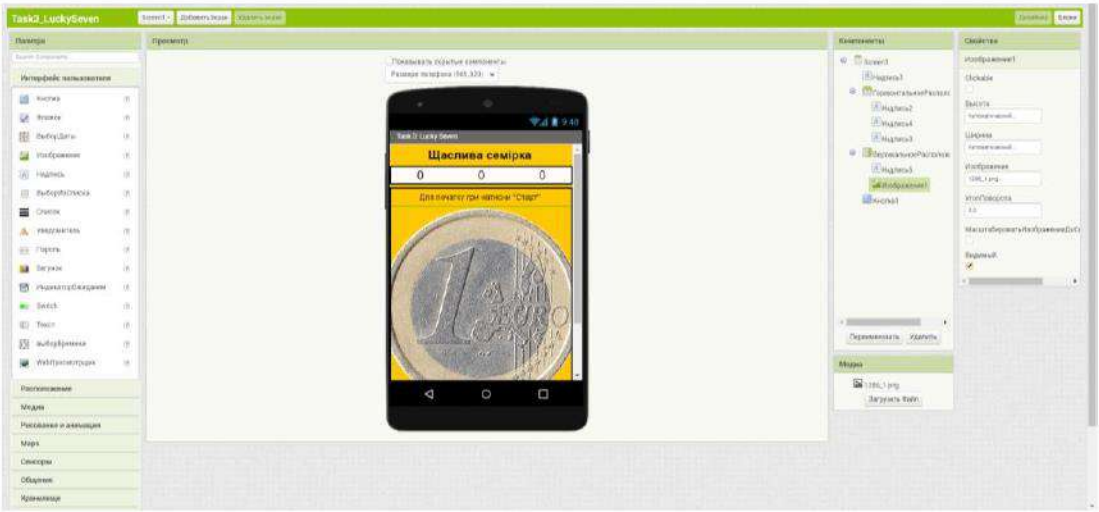


Рис. 2.1. Редактор дизайну MIT App Inventor.

Редактор дизайну можна умовно поділити на 4 частини: «палітра» – вона має всі дозволені компоненти, котрі можна використовувати для створення додатку; «компоненти» – в ній відображаються всі компоненти, які були використані для створення додатку; вікно перегляду програми; «властивості» – в цій частині є шанс побачити атрибути компоненту на натиснули, тут їх безпосередньо можна редагувати.

Компоненти програми розміщуються на екрані у режимі “Дизайн”. Усі компоненти поділені на кілька груп.

- Інтерфейс користувача включає такі компоненти як кнопка, текст, прапорець, напис та інші, які дозволяють застосунку взаємодіяти з користувачем.
- Розташування - компоненти, що відповідають за макетування екрана, дозволяють розміщувати компоненти інтерфейсу користувача горизонтально, вертикально або в осередку таблиці. У середовищі MIT App inventor немає форматування, що дозволяє задавати інтервали між певними компонентами, тому для макета та завдання відстані та простору між елементами використовуються компоненти групи розташування з певними заданими властивостями, наприклад, висота або ширина.
- Медіа - компоненти, що дозволяють задіяти у програмі різні медіа інструменти: пристрої, мікрофони та навушники, камеру, звуки та аудіофайли та інше
- Малювання та анімація – група компонентів, які дозволяють малювати або створювати анімацію у програмі.
- Сховище - компоненти, які дозволяють передавати значення всередині програми та зберігати будь-які дані програми на зовнішньому пристрої.

Редактор блоків (рис. 2.2) – це середовище, в якому розроблювачі додатків мають змогу візуально подавати логіку своїх додатків, використовуючи при цьому кольорові блоки, які об’єднуються разом, як шматочки головоломки, щоб описати всю програму загалом.

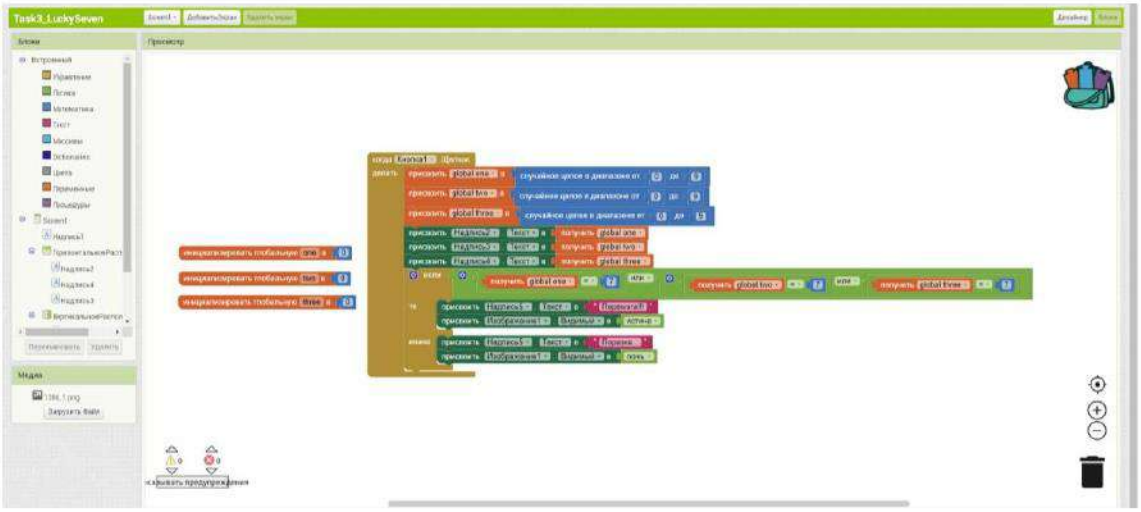


Рис. 2.2. Редактор блоків MIT App Inventor.

Редактор блоків можна поділити на 2 частини: вікно конструювання та перегляду блоків – в ньому саме і створюються блоки команд; «блоки» – в даній частині є вкладка яка містить окрім загальних блоків, має блоки до компонентів, які використані в додатку.

Щоб зарадити в тестуванні і розробці, App Inventor розробив мобільний додаток для смартфонів та планшетів під назвою App Inventor Companion або просто «Компаньйон». Цей додаток розробники можуть використовувати для тестування та налаштування поведінки своїх додатків в режимі реального часу, що є дуже зручним та зрозумілим для будь-кого. Даний додаток є безкоштовним.

При розробці MIT App Inventor головним завданням було введення освітньої мети з розробки мобільних додатків. До його випуску, велика кількість середовищ створення мобільних додатків були важкими, доступними лише для фахівців в галузі системного рівня чи вбудованого програмування, чи всього вище зазначеного разом. Навіть з операційною системою Google Labs Android і мовою програмування Java розробка інтерфейсу користувача була складним завданням. Окрім цього, для вдалого використання платформи треба було знати синтаксис та семантику Java і майстерність нормалізувати помилки компіляції Java. Всі ці проблеми репрезентували собою перешкоди

для входу людей, які не розбираються в комп'ютерних науках, але є цільовою аудиторією App Inventor.

Компоненти є основними об'єктами в MIT App Inventor. Компоненти дають шанс установити взаємодію між механізмом управління даними об'єктами та об'єктами інтерфейсу. Приміром, для застосування глобальної системи позиціонування (GPS) для слідкуванням за рухом, розробник програми додає датчика компонент місцезнаходження, за допомогою якого компонент абстрагує складність і надає API для включення і обробки оновлень місця розташування. Даний процес скорочує понад 629 рядків коду Java до 23 блоків, з котрих тільки два необхідні для здійснення відстеження місця розташування. Це покращення допомагає розробникам додатків зосередитися на головній проблемі та жваво досягти поставленої мети [14-16].

Компоненти складаються з 3 типів блоків: методів властивостей та подій. Станом компонента керують властивості і доступні для читання та/або запису розробником програми. Приміром, властивість enabled датчика розташування залучає в себе функцію, необхідну для налаштування GPS-приймача та управління його станом під час використання певної програми. З декількома параметрами працюють методи і завдяки цьому можуть повертати результат. Події реагують на зміни стану пристрою або програми в залежності від зовнішніх факторів. Наприклад, коли людина змінює своє місце розташування, реагує подія, яка відповідає за зміну положення користувача.

У MIT App Inventor, користувачі кодують поведінку та роботу програми завдяки мові програмування, яка базується на блоках. App Inventor має 2 типи блоків: вбудовані блоки та компонентні блоки. Вбудована бібліотека блоків дає доступ до базових функцій та операцій, які за часту є доступні в інших мовах програмування, такі як логічні значення, числа, списки, рядки, математичні оператори, оператори порівняння та оператори потоку управління. Розробники використовують блоки компонентів для реагування на системні та призначені для людини події, взаємодії з апаратними засобами пристроїв і налаштування візуальних, та компонентів поведінки аспектів.

Уся програмна логіка будується на 3 типах блоків верхнього рівня: визначення глобальних змінних, визначення процедур і обробники подій компонентів. Величезні змінні надають слоти для зберігання обставин програми. Процедури установлюють загальну поведінку, яка має шанс бути викликаною з декількох місць в коді. Коли на пристрої відбувається подія, вона запускає відповідну програмну поведінку, яка миттєво пропонується в блоці подій. Блок обробника подій часто має здатність посилатися на процедури або на глобальні змінні.

Розробники середовища App Inventor при проектуванні враховувала ряд обмежень:

– отримуєш те, що бачиш (What You See Is What You Get): редактор дизайну для App Inventor дозволяє розробникам бачити, як додаток буде відображатися на екрані пристрою, і налаштовувати форм-фактор візуалізованого пристрою. Це може бути як телефон, так і планшет. Коригування властивостей візуальних компонентів, таких як, колір чи розмір фону, конкретно відображаються в реальному часі;

– App Inventor звужує створення нових сутностей під час виконання. Це дає має велику перевагу. По-перше, при розміщенні всіх компонентів в додатку, людина може бачити його чітко та ясно, а не міркувати про речі, які не будуть існувати до певного часу. По-друге, це робить шанси менше для користувачів щоб вводити циклічні залежності пам'яті в інтерфейсі користувача, що в цілому приведе до переобтяження пам'яті. Це змушує розробників мобільних додатків задумуватися, як треба структурувати свої програми та повторно застосовувати компоненти;

– натуральне числення: система числення в App Inventor передбачає початкове значення 1. На відміну від більшості мов програмування, які більш узгоджені з архітектурою машини і тому починаються з 0.

Швидка ітерація та проектування з використанням «Компаньйона». Головною особливістю MIT App Inventor є живе середовище розробки для мобільних додатків. App Inventor забезпечує це завдяки супутньому додатку,

встановленого на мобільному пристрої людини. Веб-інтерфейс App Inventor відправляє код в супутній додаток, за допомогою якому інтерпретується додаток та відображається розробнику в режимі реального часу (рис. 2.3). Завдяки цьому, всі користувачі можуть змінювати інтерфейс і поведінку програми в режимі реального часу. Наприклад, здобувач освіти, який розробляє гру з кулею, де куля може відскочити від краю ігрового майданчика. Однак початкова версія додатку може не враховувати того, що куля наскочитиме на стіну, і тому вона може зупинитися. Тестуючи додаток і коригуючи його програмування у відповідь на нежадану поведінку, школярі чи інші, можуть досліджувати його більш вільно.

Рис. 2.3. Вигляд «MIT Companion»

На зображені показано вигляд самого MIT Companion (рис. ліворуч). Після того, як відбулося з'єднання з середовищем то додаток завантажується в «Компаньйона» і ми бачимо його вже на екрані пристрою та можемо взаємодіяти з ним (рис. праворуч).

Обмеженням App Inventor є те, що його дизайн не дозволяє повторне використання коду. Повторне використання коду є головним концептом обчислювального мислення в фрейм ворку Бреннана і Резніка [17]. Багато текстових мов дають певну підтримку бібліотек коду та управління

залежностями, які дозволяють розробникам додатків краще опиратися на роботи різних додатків один одного. Хоча App Inventor надає галерею для публікації завершеного вихідного коду програми, людям ще треба покращити деталізацію бібліотек, які поширені в багатьох інших мовах програмування. Це дає змогу продовжувати розвивати платформу і спільноту користувачів.

Говорячи про переваги візуального програмування для мобільних пристроїв, користувачі платформи App Inventor отримують користь від можливості перепрофілювати навички обчислювального мислення, які вони вивчають, щоб взаємодіяти з фізичним простором у зовнішньому світі. Візуальне програмування в App Inventor, а також абстракція і поділ концепцій на компоненти і блоки дозволяють розробнику додатку більше зосередитися на розкладанні своїх проблем на складові. Можливість запуску додатків на мобільних пристроях дозволяє учням сприймати свої власні програми як частину екосистеми, з якою вони щодня взаємодіють і з якою вони добре знайомі. Оскільки така інкапсуляція скорочує час, необхідний для створення додатка, навіть простого прототипу, розробники додатків можуть швидко схоплювати суть і повторювати, не витрачаючи багато часу з точки зору циклу компіляції-завантаження-запуску, який типовий для розробки мобільних додатків.

Способи завантаження програми на пристрій:

- у вихідному коді (файл з розширенням .aia)

Вихідний код у форматі .aia дає змогу редагувати програму. Вихідний код створюється зі сторінки проекту меню Проекти / Експортувати обрані проекти (.aia) на свій комп'ютер.

- у вигляді виконуваного файлу (файл з розширенням .apk)

Файл .apk генерується в App Inventor в меню Побудувати - Програма (зберегти .apk на комп'ютер). Файл .apk є програмою, яка збується та працює на пристрої.

- у вигляді QR-коду програми

Створюється завдяки команди меню Побудувати - Програма (створити QR код для завантаження .apk).

Висновки про MIT App Inventor. Проект MIT App Inventor далі продовжує розширяти кордони освіти в контексті створення мобільних додатків. Зведення складної логіки до компактних уявлень і його абстрагування апаратних здібностей дає змогу користувачам даного програмного забезпечення ефективно і швидко розробляти проекти. Функціонування в схожих середовищах удосконалює алгоритмічний тип мислення людини. Тому можна узагальнити, що вивчення цього середовища буде корисним для учнів, майбутніх студентів, які хочуть і далі розвиватися в сфері програмування, і тим, хто не захоче продовжувати роботу в цій сфері.

2.2 Тематичне планування навчального матеріалу. Розробка уроків

В ході дослідження було розроблено курс «Мій крутий додаток».

Інформаційне суспільство має надавати інформатизовані продукти, адже інформація стала невід'ємною частиною життя людини. Технології весь час оновлюються і ІКТ технології в тому числі. З'являється маса середовищ для програмування, які цілеспрямовані на формування та підтримку зацікавленості сучасної молоді до дослідження програмування та інформатики, а також підвищення престижності спеціальностей, які пов'язані з програмуванням, в очах здобувачів освіти. «MIT App Inventor» - одне з відомих середовищ створення мобільних додатків. Для створення додатку для свого мобільного пристрою, знадобиться смартфон з операційною системою Android, обліковий запис Google та осередок для програмування MIT App

Inventor. Завдяки даній платформі, учні зрозуміють, що програмування є досить доступним, актуальним та цікавим.

Мета навчання курсу: ознайомитись з поняттями програмування та здобуття прикладного досвіду в даній сфері; розвиток алгоритмічного та об'єктного способу мислення; сформування мотивування для здобуття освіти у сфері програмування, за допомогою організації практичної діяльності.

Завдання курсу:

- освоєння середовища програмування MIT App Inventor;
- отримання початкового практичного досвіду;
- розвиток логічного мислення та творчих здібностей учнів;
- розвиток уміння вибудовувати гіпотезу та зіставляти з отриманим результатом;
- вміння висловлювати свої думки у чіткій та логічній послідовності, захищати свою точку зору, аналізувати ситуацію та самостійно знаходити відповіді на питання шляхом логічних міркувань;
- стимулювати пізнавальну та дослідницьку діяльності учнів;

Це унікальне середовище, через призму діяльності в якому, будь-який учень вчиться можливості осмислити сутність і природу таких базових понять інформатики як «програма», «алгоритм», «виконавець», «підпрограма», «модель» та ін. об'єктно – орієнтованого програмування, знайомлячись із поняттями «об'єкт», «клас», що є теоретичним фундаментом освоєння базових понять «інкапсуляція», «поліморфізм», «успадкування».

Велика кількість позитивних аспектів реалізації дидактичного потенціалу в середовищі для розробки мобільних додатків «MIT App Inventor» дозволяють зробити висновки, що введення даного середовища в освітній процес продукує об'єктивні умови для ранньої профілізації учнів формування

мотивації у тих, хто навчається до отримання ІТ – освіти через здобуття практичного досвіду.

Говорячи про форми організації позаурочної діяльності, слід зазначити, що інноваційні освітні моделі, такі як, факультативи та інші можуть бути ефективно використані у процесі викладання даного курсу.

Програма враховує вікові особливості усіх здобувачів освіти. Розроблене тематичне планування передбачає значне збільшення активних форм роботи, спрямованих на залучення учнів у діяльність, на забезпечення розуміння ними матеріалу та розвитку інтелекту, набуття практичних навичок, умінь проводити міркування. З цією метою допускається пересування класом у ході виконання групових завдань та участі в ігрових ситуаціях.

Зміст програми

Дане тематичне планування розраховане на здобувачів освіти, які навчаються в 10-11 класах. Програма має призначення як для проведення регулярних щотижневих урочних чи позаурочних занять зі школярами, так і можливість організовувати заняття великими блоками.

Заняття можуть проводитись у вечірній, канікулярний час, чи у вихідні дні.

№	Назва розділу	Всього	Теорія	Практика	Форми контролю
1	Основи створення програм для мобільних пристроїв. Введення в середовище програмування програм для мобільних пристроїв MIT App Inventor. Основні структурні блоки програмування.	4	3	1	Знайомство, бесіда, спостереження

2	Основні компоненти програми. Дизайн програми та програмування компонентів.	5	1	4	Спостереження, виконання завдань на ПК
3	Екрани програми та передачі даних між ними.	7	2	5	Спостереження, виконання завдань на ПК
4	Кольори у додатку.	2	1	1	Спостереження, виконання завдань на ПК
5	Малювання. Компонент «Полотно».	8	3	5	Спостереження, виконання завдань на ПК
6	Анімація об'єктів у мобільних додатках.	4	-	4	Спостереження, виконання завдань на ПК
7	Використання сенсорів у програмі.	7	1	6	Спостереження, виконання завдань на ПК
8	Розпізнавання мови.	7	1	6	Спостереження, виконання завдань на ПК
9	Масиви та списки у додатку.	9	3	4	Спостереження, виконання завдань на ПК
10	Підсумковий проект. Розробка та налагодження мобільного додатка.	6	-	6	Спостереження, виконання завдань на ПК, презентація проектів, змагання
11	Загальна кількість годин: 50		15	35	

Реалізація програми курсу забезпечується такими матеріальними ресурсами: наявністю проектора, інтерактивної дошки, комп'ютерів із виходом до Інтернету.

У процесі викладання цієї програми «Мій крутий додаток» важливим компонентом є такі засоби навчання:

- друковані посібники (роздавальний та дидактичний матеріали);
- наочні посібники (плакати, таблиці, інфографіка);
- електронні освітні ресурси (мультимедійні засоби навчання).

Апаратні засоби:

- персональний комп'ютер;
- проектор;

Програмні засоби:

- Google або Google Apps обліковий запис;
- додаток MIT AI2 Companion App;
- ПЗ App Inventor Setup Software.

2.3 Конспекти уроків

РОЗШИРЕНИЙ КОНСПЕКТ УРОКУ ІНФОРМАТИКИ №1

Тема: Знайомство із середовищем програмування MIT App Inventor

Мета:

А) навчальна: ознайомити з можливостями ВП для мобільних пристроїв; познайомити з візуальним середовищем програмування MIT App Inventor та принципами створення у ньому мобільних додатків; навчити проектувати мобільні програми, створювати програми та виконувати їх налагодження на мобільних пристроях;

Б) розвиваюча: сприяти розвитку інтересу підлітків до програмування та мобільних технологій, сприяти розвитку пам'яті; алгоритмічного та аналітичного мислення;

В) виховна: сприяти профорієнтації підлітків, стимулювати прагнення отримання технічних знань; сприяти отриманню підлітками досвіду співробітництва, колективної взаємодії; навчити оцінювати результати своєї та чужої праці.

Тип уроку: урок засвоєння нових знань.

Обладнання: ПК, конспект, проєктор, смартфон.

ПЛАН УРОКУ

I. Мотивація навчальної діяльності
II. Актуалізація опорних знань
III. Вивчення нового матеріалу
IV. Інтерактивна вправа
V. Підбиття підсумків
VI. Домашнє завдання

Х І Д У Р О К У

І. Етап організації класу

1. Привітання вчителя.
2. Перевірка, чи всі присутні на уроці.
3. Перевірка домашнього завдання.

II. Мотивація навчальної діяльності.

На ринок комп'ютерної техніки, окрім персональних комп'ютерів, впевнено вийшли мобільні гаджети: планшети, смартфони, айфони. Тому зріс попит на кваліфікованих спеціалістів у галузі розробки мобільних застосунків (додатків). ОС Android є найпоширенішою серед операційних систем для мобільних гаджетів.

Програмування мобільних програм на платформі Android може стати в нагоді для розробки дуже широкого кола програмних систем, починаючи від ігрових програм для мобільних телефонів, і закінчуючи професійними системами, що використовують сучасні технології.

III. Актуалізація опорних знань

На минулому уроці ми обговорювали візуальне програмування та його особливості та можливості в сьогоденні. Зараз дайте відповідь на декілька запитань?

- Що таке візуальне програмування?
- Де використовуються ВМ?
- Які програми для роботи з ВМ ви знаєте?

IV. Вивчення нового матеріалу

App Inventor - це середовище візуальної розробки android-додатків, яке вимагає від користувача най мінімальних знань мови програмування [17]. Спочатку розроблено в Google Labs, після закриття цієї лабораторії дане середовище було передано Массачусетському технологічному інституту. На початку березня 2011 року зумовлений інститут створив публічну бета-версію проекту, яка на даний момент є доступною на сайті appinventor.mit.edu. Працює App Inventor безпосередньо з браузера. Отриманий результат можна переглядати на Android-пристрої. Готові додатки можна розміщувати у Play Market. З серпня 2015 року App Inventor 2 починає підтримувати російську мову, українську нажаль досі ні.

В онлайн редакторі MIT App Inventor 2 будь-які додатки будуються за допомогою головних опорних стандартних компонентів, які виступають основним елементом розробки додатків Android.

Блоки App Inventor

Важливі поняття та принципи блоки App Inventor є інструментами для оперування компонентами і виглядають як пазли. Блоки в цьому конструкторі додатків для Android розбиті на дві великі групи за ознакою – на що впливають і до чого відносяться: що стосуються безпосередньо компонентів, що відносяться до додатку в цілому.

Почнемо з блоків, що належать до компонентів.

Їх можна розділити на три типи, які легко розрізнити за кольором:

1. блоки, які описують характеристики елемента зеленого кольору і виглядають так. Цей тип блоку-характеристики можна було б віднести до команд (оброблювачам), оскільки він справді дає команду змінити будь-яку якість елемента, також, і значення полів.

2. блоки-події, тобто ті блоки, відстежують настання якого-небудь події в додатку, приміром, натискання кнопки і далі запускають блок-команду. Вони мають бронзовий колір

3. блок-команда, в App Inventor даний блок часто іменують обробником. Цей блок показує, що необхідно зробити з компонентом, до якого належить блок, який викликає дані з таймера пристрою.

Друга група блоків, що належать до всього додатку, організована дещо інакше.

Для початку ось список підгруп:

- Logic blocks – логічні блоки
- Math blocks – математичні блоки
- Text blocks – текстові блоки
- Lists blocks – блоки для керування списками
- Colors blocks – блоки для керування кольором
- Variables blocks – блоки для керування змінними

- Procedures blocks – блоки процедур.

Всі вони, крім Procedures blocks вбудовуються в інші блоки. Іншими словами, вони не можуть служити вихідним блоком, на відміну від елементів блоків-подій, що належать – всі дії відбуваються при всіх подіях з елементами. Ось тут варто розповісти ще про типи «пазлів». Отже, ви напевно побачили, що є пазли чотирьох видів. З їхньої форми цілком очевидно, що першим видом починається будь-яка ланцюжок у програмі для телефону. Це і цілком розумно, що вона ініціює всі наступні дії. І цей тип не відрізняється від прийнятих в даному архітекторі додатків для Андройд.

При розробці алгоритмів для швидкості та простоти хочеться різні операції помістити в одну функцію, що призведе до швидкого підвищення числа блоків та труднощів з усвідомленням її роботи. Для цього у програмуванні широко використовується традиційне правило: Одна функція (процедура) – одна операція.

V. Інтерактивна вправа

А зараз давайте розглянемо які блоки вам необхідно використати для подальшої вправи.

Учні досліджують, які блоки треба використати і за допомогою яких в створити правильну команду.

VI. Підбиття підсумків уроків

А зараз щоб закріпити новий теоретичний матеріал, пропоную вам по колу відповісти на запитання:

1. Як ви розумієте поняття алгоритм?
2. Як ви розумієте поняття візуальне програмування?
3. В чому полягає різниця алгоритмом та задачею?
4. В яких випадках використовують блок-команда?
5. В яких випадках використовують блок-подія?

А тепер, бажаючі висловлюють власну думку щодо поставлених запитань та аналізують виконані дії.

Закінчити урок я пропоную закінчивши речення:

«Інформація, отримана на сьогоднішньому уроці дозволяє мені зробити висновок, що ….»

VII. Домашнє завдання

В якості домашнього завдання учням пропонується скласти кросворд термінів щойно вивченого матеріалу, мінімальною кількістю 10 слів

РОЗШИРЕНИЙ КОНСПЕКТ УРОКУ ІНФОРМАТИКИ №2

Тема: Створення додатку «Малюй пальцем по екрану»

Мета:

А) навчальна: познайомити з різними функціями візуального середовища програмування MIT App Inventor та принципами створення у ньому мобільного додатку;

Б) розвиваюча: сприяти розвитку інтересу підлітків до програмування та мобільних технологій, алгоритмічного та аналітичного мислення;

В) виховна: стимулювати прагнення отримання технічних знань; навчити оцінювати результати своєї та чужої праці.

Тип уроку: формування навичок і вмінь.

Обладнання: конспект, ПК, смартфон.

ПЛАН УРОКУ

I. Мотивація навчальної діяльності
II. Актуалізація опорних знань
III. Практична робота
IV. Підбиття підсумків
V. Домашнє завдання

ХІД УРОКУ

І. Етап організації класу

1. Привітання вчителя.
2. Перевірка, чи всі присутні на уроці.
3. Перевірка домашнього завдання.

ІІ. Мотивація навчальної діяльності.

Візуальне програмування MIT App Inventor навчить розробляти власні додатки та проекти, які в майбутньому будуть коштувати великих грошей, а людина з потрібними знаннями завжди буде затребуваний;

ІІІ. Актуалізація опорних знань

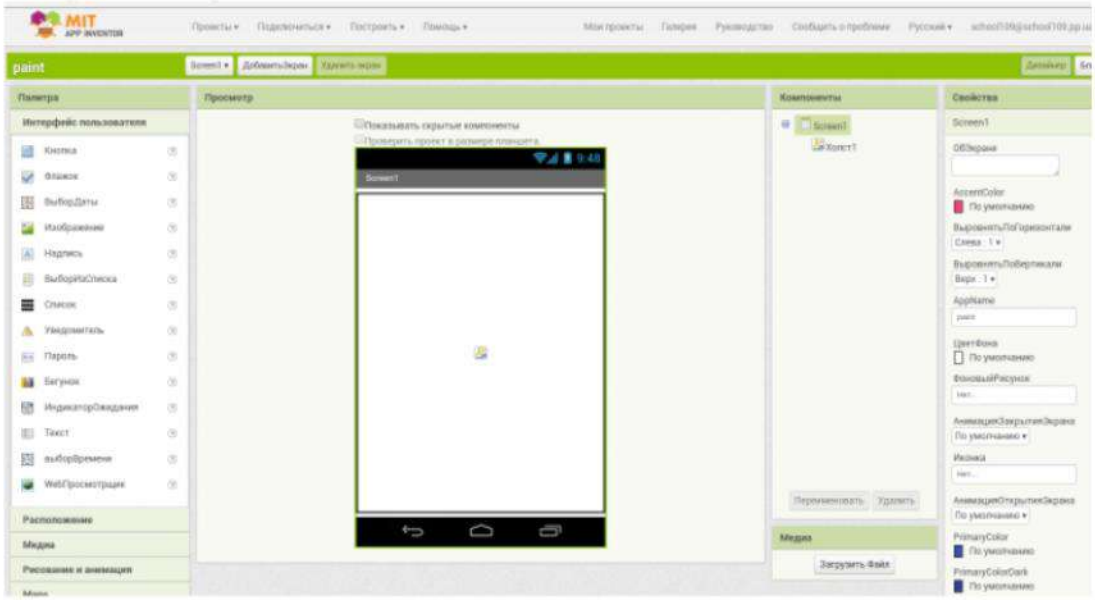
На минулому уроці ми обговорювали візуальне програмування та його особливості та можливості в сьогоденні. Зараз дайте відповідь на декілька запитань?

- Які особливості при використанні MIT App Inventor ви знаєте?
- В яких галузях можна використовувати MIT App Inventor?

IV. Практична робота

1. Невеликий урок у якому зробимо програму для малювання на компоненті Canvas пальцем.

2. Знадобиться компонент Canvas ось власне і все. Розтягуємо Canvas на весь екран.



3. Заходимо у властивості компонента п міняємо розміри Width - Fill Parent,

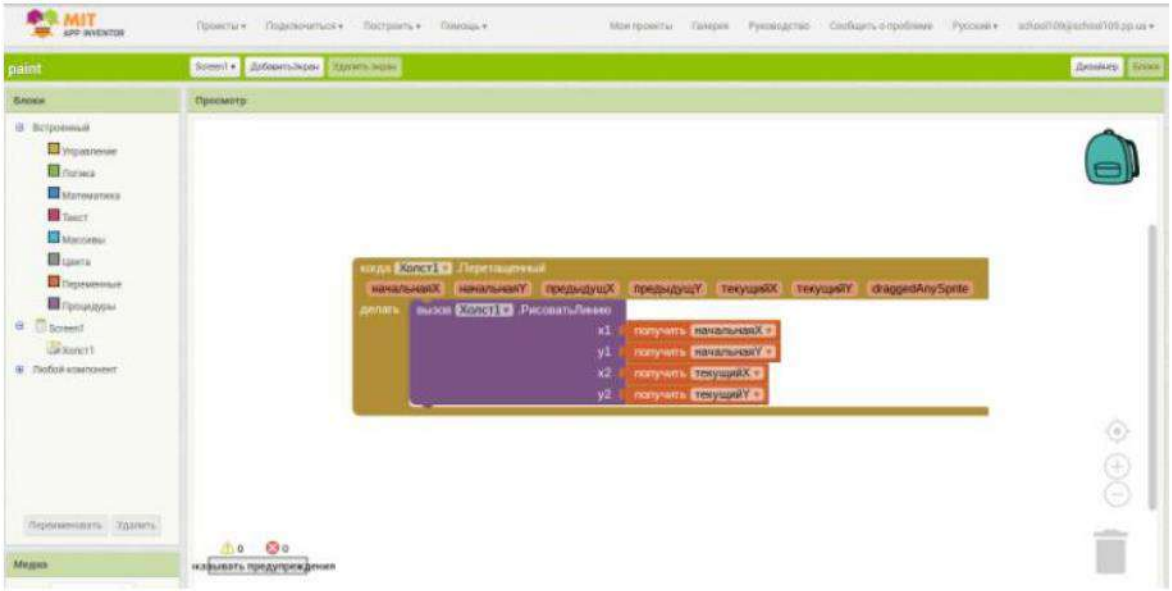
4. Height Fill Parent

5. Збираємо блоки у редакторі блоків.

6. Блок When Canvas1.Dragged відстежує координати дотику до екрану по осях X II У стартові. поточні та попередні

7. До першого блоку додаємо call Canvas1.DrawLine з параметрами x1.y1.x2.y2

8. До цих параметрів додаємо координати дотику.



V. Підбиття підсумків

А тепер для закріплення практичної частини матеріалу пропоную вам відповісти на запитання:

1. Які блоки ви використали?
2. Що таке Canvas?
3. Які кольори для малювання можна використовувати?

А тепер, бажаючі висловлюють власну думку щодо поставлених запитань та аналізують виконані дії.

VI. Домашнє завдання

В якості домашнього завдання учням пропонується створити схожий додаток, тільки з більшою кількістю кольорів та проаналізувати реультат.

2.4 Розробки учнів

Після теоретичної та практичної частини, учням було запропоновано створити власні додатки. Здобувач освіти повинен окрім самого додатку показати і блоки команд. Для цього йому потрібно експортувати із середовища файл додатку (має розширення .aia). Демонстрація готових завдань відбувається у вигляді презентації. Учні по черзі демонструють свої розробки вчителю. Також учень повинен чітко та лаконічно відповісти на запитання вчителя та можливо своїх однокласників, якщо такі будуть. Проводячи перевірку таким методом можна запросто побачити як легко той чи інший учні опанував середовище, як він в ньому орієнтується і чи самостійно він створив свій додаток.

Нижче буде показано та описано деякі з додатків створених учнями загальноосвітньої школи м. Тростянець. Учні брали участь по своєму бажанню, проявляли велике зацікавлення та інтерес.

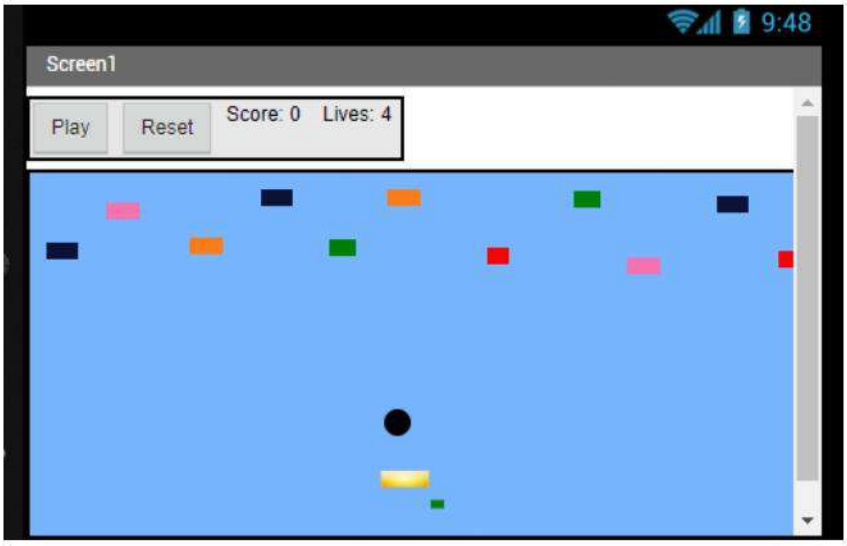


Рисунок 1.1. Екран на смартфоні

Першим розглянемо додаток «Рікошет» (рис. 1.1). На екрані смартфону ми бачимо блакитний фон, на якому розташована куля. Її завдання полягає в знищенні всіх верхніх різнокольорових блоків-перешкод. Під кулею розташований жовтий прямокутник, за допомогою якого куля підбивається та летить догори. Щоб пересунути цей прямокутник, гравцю необхідно

крутити в різні боки свій телефон. Рухи мають біти плавні та обережні.

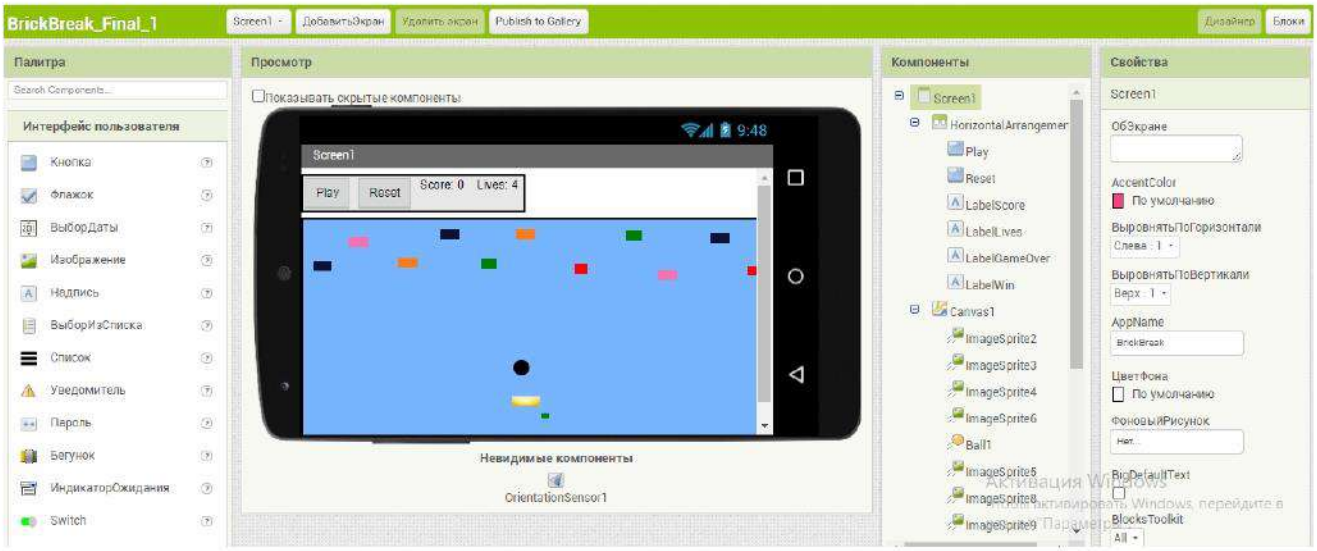


Рисунок 1.2. Екран в редакторі дизайну

На скріншоті (рис. 1.2) можна спостерігати частину медіа файлів (зображення) та компонентів, які застосовувалися при створенні гри. Це такі компоненти як: кнопка, горизонтальне розташування прокрутки та полотно. За допомогою властивостей компоненту кнопка, можна розпочати гру та почати її проходити спочатку в любий момент. Всі кнопки розташовані вгорі та мають однаковий розмір, та гарне масштабування на екрані, що робить все зображення гри задовільним. Це говорить про те, що учень, який створював цей додаток був дуже уважним при роботі та переймався про майбутніх користувачів.

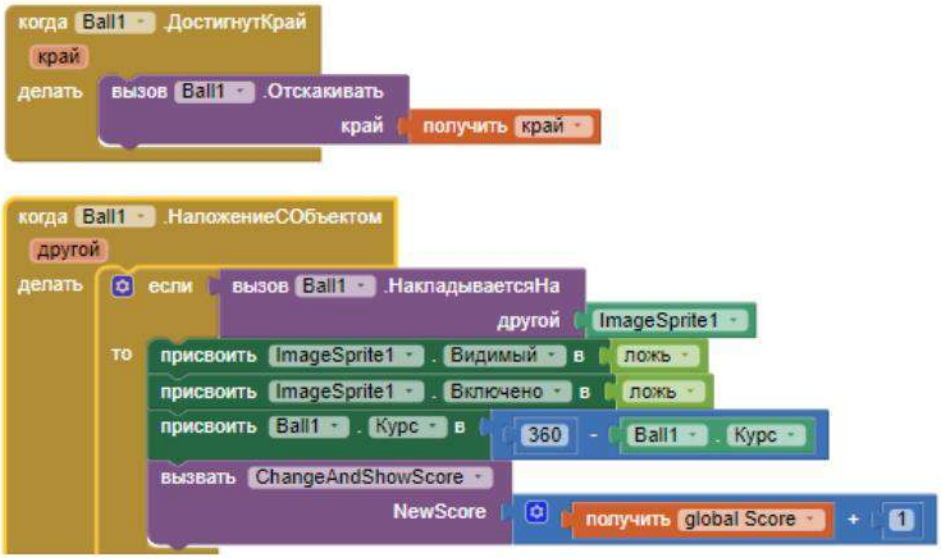


Рисунок 1.3 Блок команд до об’єкту Ball1.

Даний блок команд можна побачити на наступному скріншоті (рис. 1.3). Для об’єкту Ball1 було використано по однаковій команді до кожного зображення-спрайту, який розташований вгорі самої гри. Всі команди виконують одну й туж функцію по відношенню до кулі, тому вони вбудовані та приєднані разом.

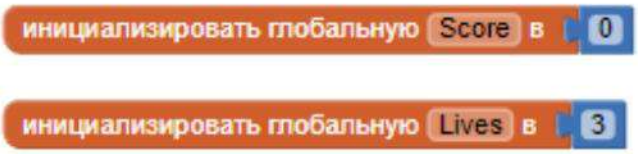


Рисунок 1.4 Блок змінні.

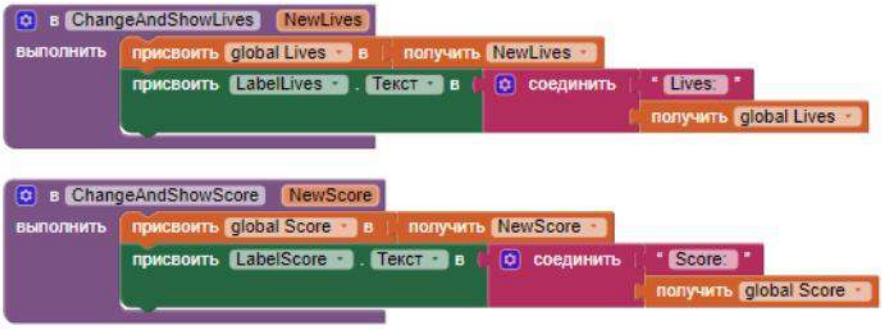


Рисунок 1.5 Блок команд процедури.

Щоб гра не здавалася безкінечно нудною, було створено блоки, завдяки яким, в гравця цієї гри було, лише 3 «життя». Якщо ж всі «життя» були використані, то гра автоматично починається з початку і всі верхні перешкоди з’явилися знову. Дані блоки команд можна побачити на наступних скріншотах (рис. 1.4, рис. 1.5).

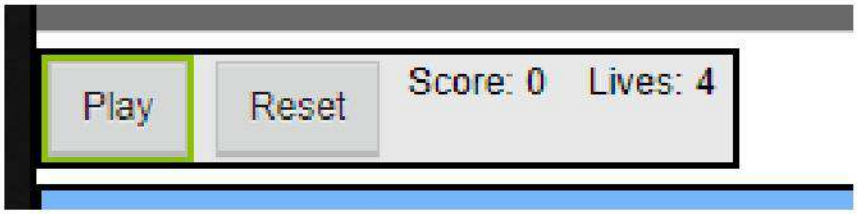


Рисунок 1.6 Верхня панель додатку

Для зручності кнопки розташовані вгорі. При натиску на кнопку «play» гра починається та кулька починає свій рух. Якщо ви хочете почати грати спочатку, то для цього існує в додатку кнопка «reset».

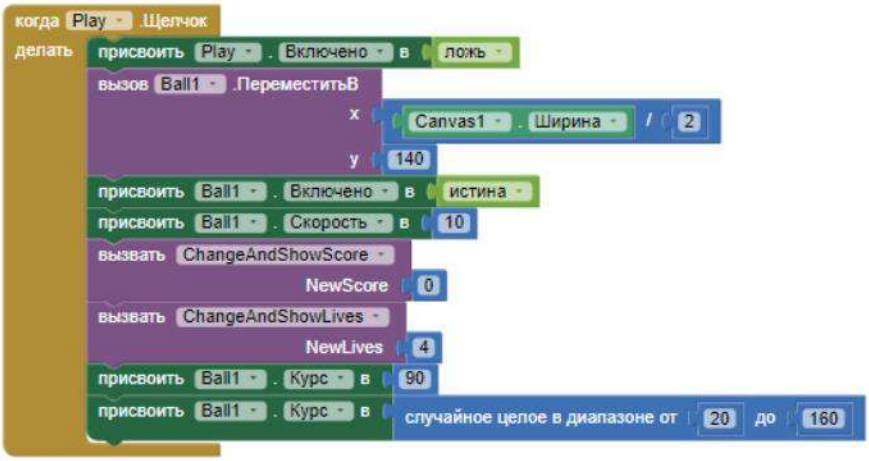


Рисунок 1.7 Блок команд керування.

Щоб кулька вільно рухалася після натиску на кнопку «play», було використано команду присвоєння та виклику. Цей об’єкт тепер без перешкод пересувається сам по заданій площині.

Ця гра допомагає розвинути увагу, кмітливість та моторику рук. Гра не є візуально досконалою, проте робота над блоками, змушує прикрити очі на ті зовнішні зображення додатку та насолодитися самою грою.

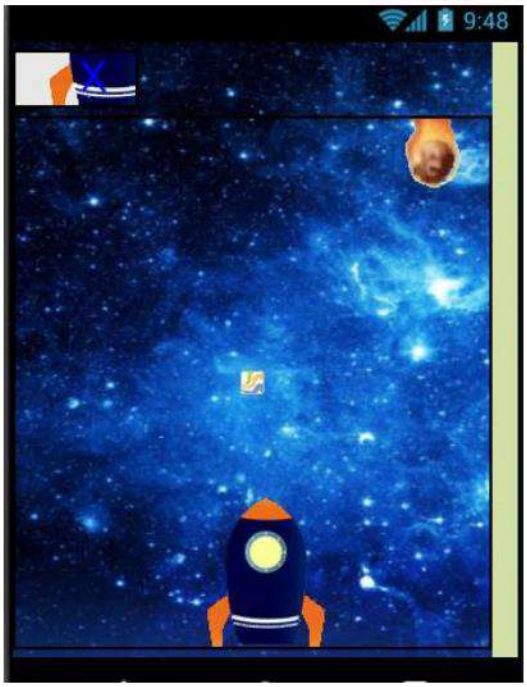


Рисунок 2.1. Екран на смартфоні

Другим розглянемо додаток «Космічна подорож» (рис. 2.1). На екрані можна побачити гарний космічний фон, на якому розташований космічний корабель та різні перепони у вигляді метеоритів. Завдання якого полягає в

проході всіх метеоритів, які заважають руху транспорту. Щоб рухати цей космічний корабель, гравцю необхідно натиснути на нього, плавно та обережно пересувати по поверхні екрану смартфону.

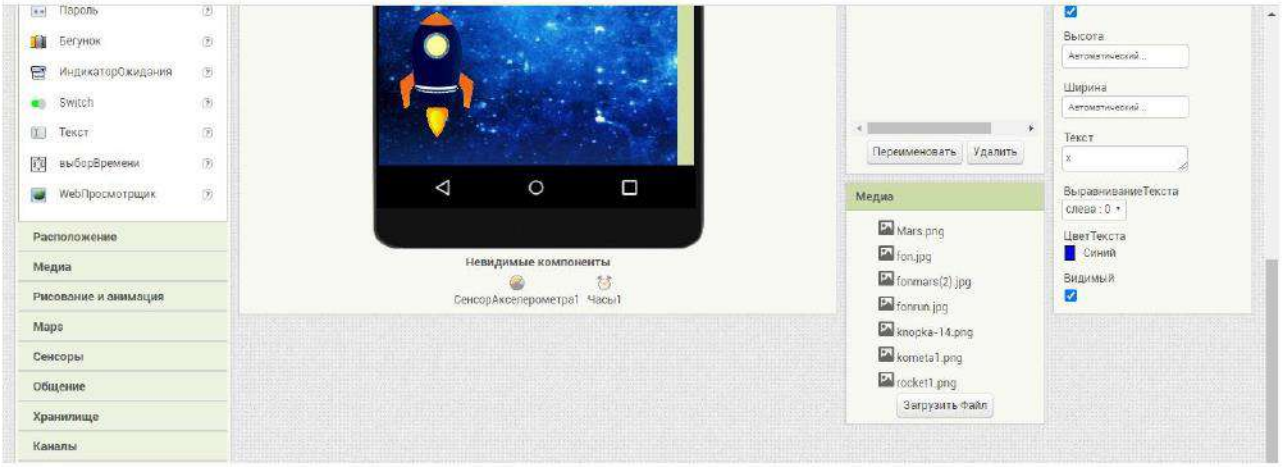


Рисунок 2.2 Екран в редакторі дизайну

На скріншоті (рис. 2.2) помітно медіа файли та компоненти, які застосовувалися при створенні додатку. Це такі компоненти як: кнопка, годинник, сенсор акселерометра, надписи та картинки.

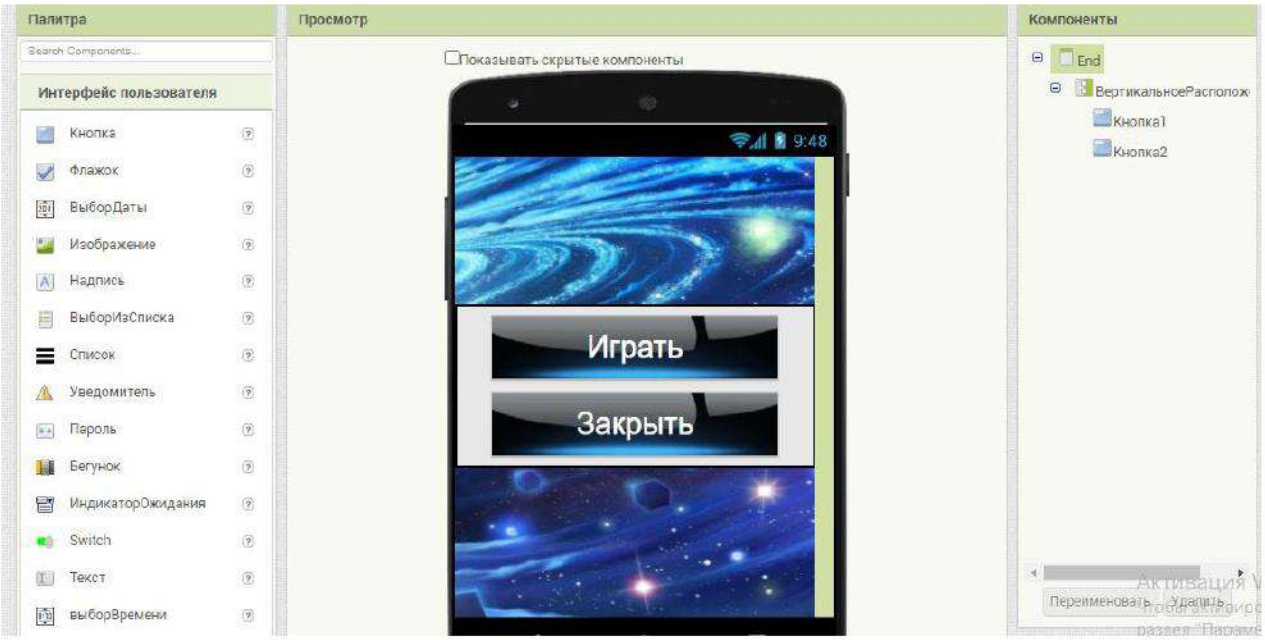


Рисунок 2.3 Екран початку гри

За допомогою властивостей компоненту кнопка1, можна розпочати гру, а при натисканні на кнопку 2 можна вийти з додатка. Всі кнопки розташовані

вгорі та мають однаковий розмір, та гарне масштабування на екрані, що робить все зображення гри задовільним. Це говорить про те, що учень, який створював цей додаток був дуже уважним при роботі та переймався про майбутніх користувачів.

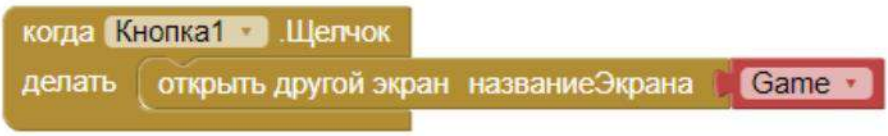


Рисунок 2.4 Блок для кнопки 1

Для того щоб почати гру використовують влаштований блок «керування», а в кінці додають текстову строку, де вказується назва екрану для гри.

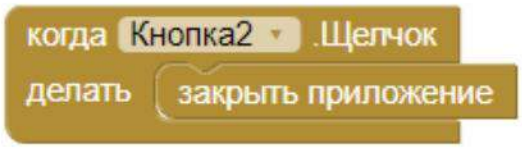


Рисунок 2.5 Блок для кнопки 2

Для того щоб вийти з додатку використовують влаштований блок «керування».

Рисунок 2.6 Блоки для сенсора акселерометра

Щоб зробити об’єкт рухомим використовуються блоки які показані на рис. 2.6. Також є блоки для годинника та існує певне обмеження.

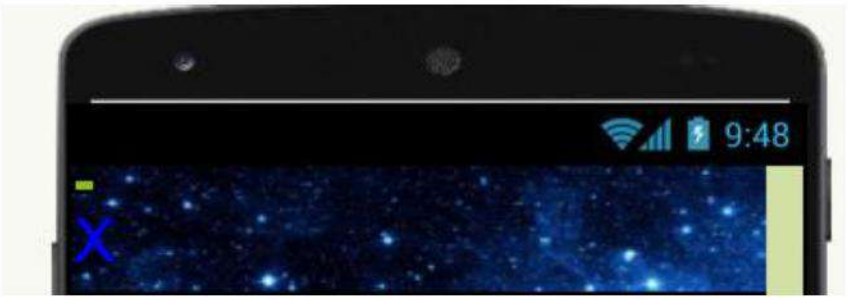


Рисунок 2.7 Верхня панель додатку

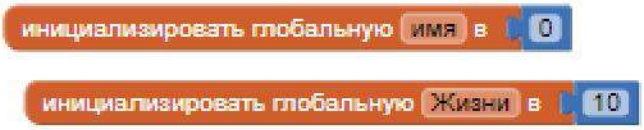


Рисунок 2.8 Верхня панель додатку

Вгорі додатку можна побачити синій хрестик, який інформує про кількість втрачених «життів». Гравцю під час гри дуже зручно контролювати цей процес, адже позначка велика і помітна. Для цього використовують блоки «змінні», як для ім’я, так і контролю життя.

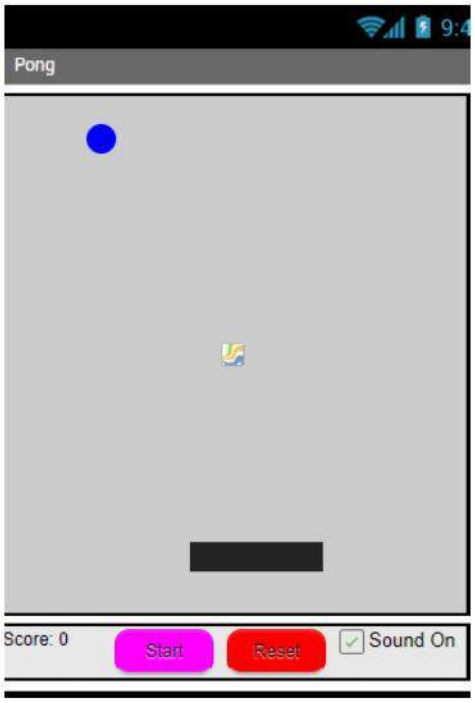


Рисунок 3.1 Екран на смартфоні

Третім розглянемо ще один додаток «Simple ricochet» (рис. 3.1). Ця гра за принципом дії схожа на перший додаток, проте її зовнішній вигляд дещо простіший. На екрані можна побачити звичайний сірий фон, на якому розташований чорна плита для відбивання та синя куля. Завдання плити - не пропустити кулю нижче себе. Щоб рухати цей блок, гравцю необхідно натиснути на неї, плавно та обережно пересувати по поверхні екрану смартфону. Також в даному додатку можна заробляти бали за влучне відбивання кулі.

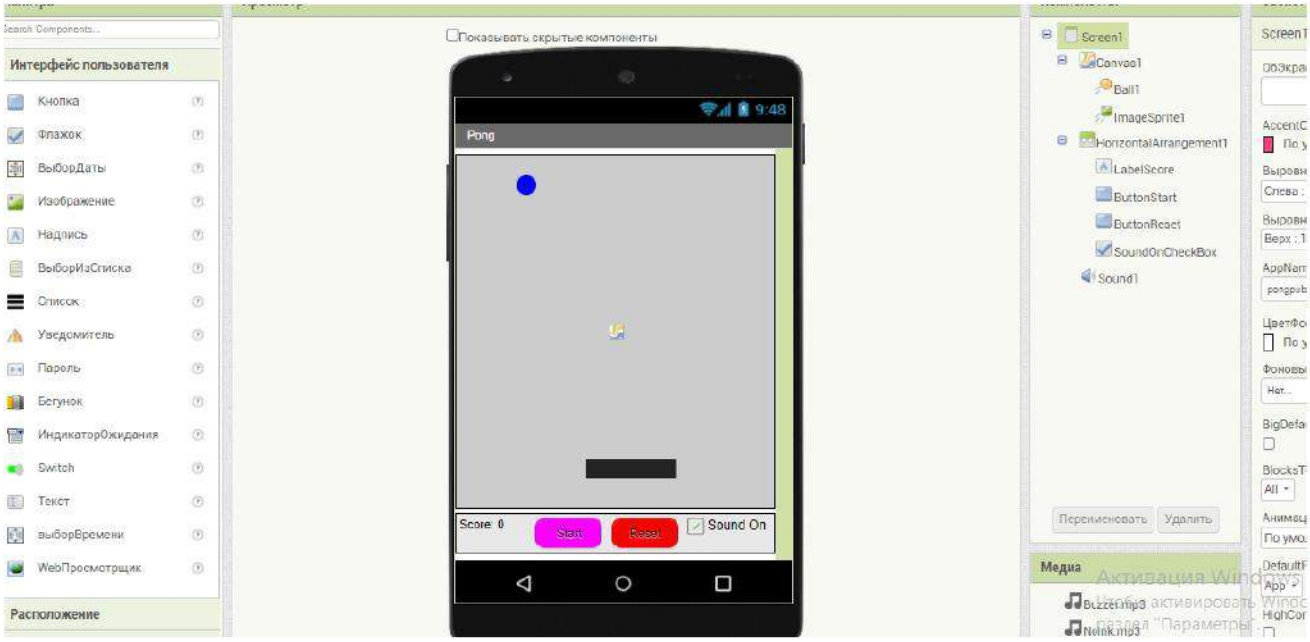


Рисунок 3.2 Екран на смартфоні

На скріншоті (рис. 3.2) можна спостерігати частину медіа файлів та компонентів, які застосовувалися при створенні гри. Це такі компоненти як: дві кнопки, звук, куля, горизонтальне розташування прокрутки та полотно. За допомогою властивостей компоненту кнопка, можна розпочати гру та почати її проходити спочатку в любий момент. Для зручності кнопки розташовані внизу. При натиску на кнопку «start» гра починається та кулька починає свій рух. Якщо ви хочете почати грати спочатку, то для цього існує в додатку кнопка «reset».

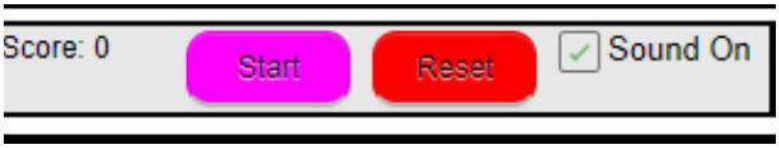


Рисунок 3.3 Екран на смартфоні

Дані кнопки мають різні кольори, це робить їх використання більш зручними. Зліва можна побачити кількість зароблених балів за гру.

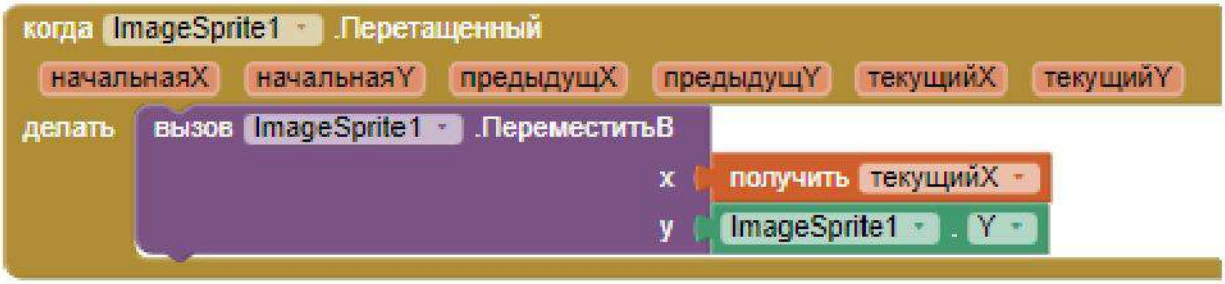


Рисунок 3.4 Блок для плити відбивання

Щоб плита для відбивання функціонувала, створюється блок з використанням блоків «керування», «процедури», «логіки» та «змінної».

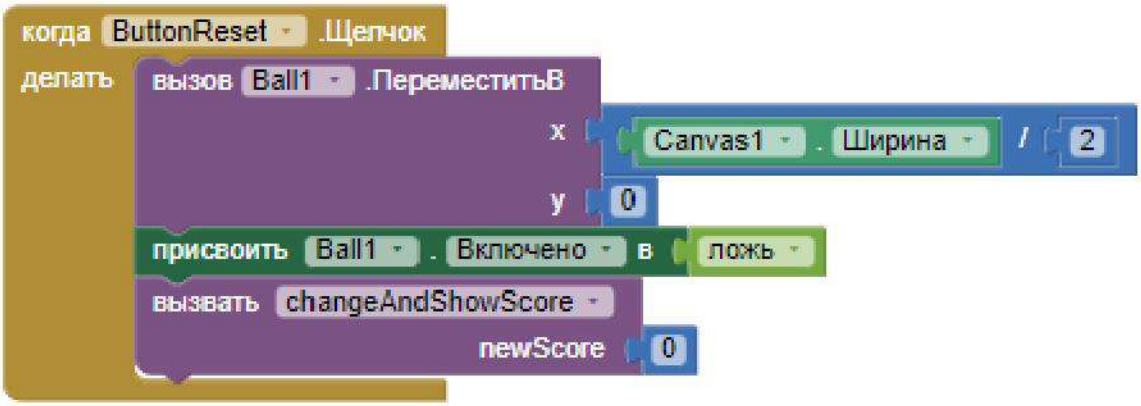


Рисунок 3.5 Блок для кнопки «reset»

Для перезапуску гри в будь-який момент застосовується кнопка «reset». Тільки гра починається знову, то за допомогою блоків «математика», «процедури» та «логіка».

Рисунок 3.5 Блок для кнопки «start»

Для початку гри застосовується кнопка «start». Після натискання на неї синя куля починається рухатися і задача гравця набрати як найбільше балів. Це відбувається то за допомогою блоків «математика», «процедури» та «логіка». Принцип дії такий, як і в кнопки «reset».

Рисунок 3.6 Блок кулі

Щоб куля могла вільно рухатися, було створено вище зображений блок. Куля відскакує за допомогою блоку «процедури» та «змінної» яку зазначили, як край.

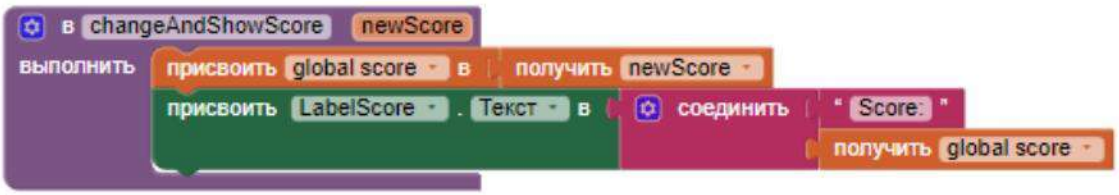


Рисунок 3.7 Блок балів

Кількість балів підраховується автоматична за допомогою блоку «процедури», який поєднує в собі «змінні» та «логіку».

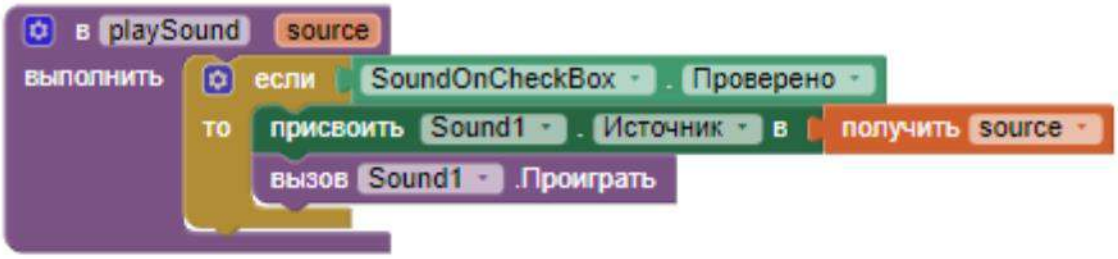


Рисунок 3.8 Блок звук

Для звуку був створений вище зазначений блок.

ВИСНОВКИ

Аналіз матеріалів з теми візуальне програмування надає такі результати:

- Візуальне програмування дозволяє створювати додатки та програми для різних операційних систем не за допомогою написання коду – а взаємодією з візуальними компонентами.

- Візуальне програмування варто пропонувати до вивчення текстових мов програмування, створення лістингу програм учнями;

- Вивчення та робота у візуальних середовищах програмування розвиває алгоритмічний стиль мислення. Даний тип мислення дозволяє розв'язувати задачі, що виникають у будь-якій сфері діяльності людини, не тільки в програмуванні. Добре розвинений алгоритмічний стиль мислення, вміння мислити точно, стає одними з важливих ознак загальної культури людини в цьому цифровому світі.

- Розвиток навичок у візуальному програмуванні буде розвивати навички програмування. Тому навчання візуального програмування є досить корисними.

Ознайомившись та вивчивши середовище візуального програмування MIT App Inventor 2 можна зрозуміти наступне:

- MIT App Inventor 2 це середовище візуального програмування для розробки додатків для операційної системи Android.

- Для роботи в середовищі MIT App Inventor 2 не обов'язково відмінно володіти текстовими мовами програмування.

- Середовище App Inventor 2 дозволяє, за допомогою супутнього додатку MIT App Inventor 2 Companion, швидко тестувати та редагувати створені додатки в режимі реального часу.

- Робота в цьому середовищі буде корисна як учням, які далі хочуть розвиватися в програмуванні, також і тим здобувачам освіти, які не мають бажання працювати в цій сфері.

Стосовно навчання візуального програмування у закладах освіти різного рівня, то проаналізувавши наукові та методичні матеріали з теми можна зробити висновки:

- Вивчення візуального програмування на даний момент не дуже поширене.

- Теоретичний матеріал, що викладається, потрібно закріплювати практично, тому вчитель має обрати один з двох варіантів: на теоретичному занятті безпосередньо запропонувати учням за інструкцією виконати дії зі створення вивченої програми. Другим варіантом може бути роздатковий матеріал, на якому інструкції зі скріншотами допоможуть учням самостійно вдома виконати таке ж завдання.

- При розробці курсу для учням потрібно оволодіти середовищем в якому вони будуть працювати, пояснити важливість та актуальність вивчення даної теми.

Запропоновані розробки методичних матеріалів до теоретичних та практичних робіт дають змогу стверджувати, що візуальне програмування доступне для опанування учням, студентам з різною базовою підготовкою і спеціальної освіти чи підготовки щодо отримання навичок роботи з MIT App Inventor не потребують.

Таким чином за допомогою середовища MIT App Inventor можна ознайомити та навчити учнів візуальному програмуванню, що стане у нагоді в майбутньому вивченні програмуванні. В цілому поставлені завдання роботи виконані і мета є досягнутою.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Абрамик М.В., Лещук С.О., Олексюк В.П. Використання хмарних технологій у процесі навчання майбутніх учителів інформатики основам програмування. Фізико-математична освіта. 2018. Випуск 4(18). С. 7-11.

2. Алгоритмічний стиль мислення URL: https://studfile.net/ preview/7517256/page:6/

3. Атаманюк С.І., Шишенко І.В., Семеніхіна О.В. Інновації в освіті та специфічні принципи підготовки майбутніх фахівців їх використовувати. Фізико-математична освіта. Суми, 2020. Вип. 4(26). Ч. 2. С. 13-16.

4. Бобровицька С.Ф., Семеніхіна О.В. Стан розробленості проблеми підготовки майбутніх учителів початкової школи до застосування електронних освітніх ресурсів у професійній діяльності. Педагогіка та психологія. 2019. Вип. 62. С. 23-29.

5. Будянський Д.В., Друшляк М.Г., Семеніхіна О.В., Харченко І.В., Горбачук В.О., Чашечникова О.С. Типологія електронних ресурсів у формуванні риторичної культури фахівця. Інформаційні технології і засоби навчання. 2021. 81(1), С. 82-96. https://doi.org/10.33407/itlt.v81i1.4292

6. Вакал Ю.С., Шамоня В.Г. Організація педагогічного експерименту із використанням сучасних інформаційних технологій: навч. посіб. Суми: СумДПУ імені А. С. Макаренка, 2020. 156 с.

7. Візуальне програмування/ URL: https://uk.wikipedia.org/wiki / Візуальне програмування

8. Ворожбит А.В., Рибак О.С. Огляд курсу за вибором «основи верстки та веб-програмування». Фізико-математична освіта. 2018. Випуск 1(15). С. 20-27

9. Дегтярьова Н., Петренко С. Актуальні питання формування цифрових компетентностей вчителів різних дисциплін під час підвищення кваліфікації. Актуальні питання гуманітарних наук: міжвузівський збірник наукових праць молодих вчених Дрогобицького державного педагогічного

університету імені Івана Франка. Дрогобич: Видавничий дім «Гельветика», 2020. Вип. 27. Том 2. С. 167-170.

10. Дегтярьова Н.В., Петренко С.І. Змішане навчання як чинник формування навичок самоосвіти у майбутніх вчителів інформатики. Вісник Вінницького політехнічного інституту. 2(143). 2019. С. 117-122.

11. Дегтярьова Н.В., Руденко Ю.О., Вернидуб Г.О. Формування вміння у майбутніх учителів працювати над науковим текстом. Педагогіка формування творчої особистості у вищій і загальноосвітній школах: зб. наук. праць. Запоріжжя: КПУ, 2020. Вип. 68. Т.1. С. 240-243.

12. Дегтярьова Н.В., Руденко Ю.О., Шамоня В. Г., Семеніхіна О.В. Методика вирішення нечітких багатокритеріальних задач вибору варіантів. Збірник наукових праць Національного університету кораблебудування імені адмірала Макарова, 2020. № 3 (481). С. 124-128. https://doi.org/10.15589/znp2020.3(481).16

13. Друшляк М. Г., Юрченко А. О., Розуменко А. М., Розуменко А. О., Семеніхіна О. В. Ефективні форми підвищення кваліфікації вчителів у галузі комп'ютерної анімації. Відкрите освітнє е-середовище сучасного університету, 2021, 10 (1), С. 77-88. https://doi.org/10.28925/2414-0325.2021.108

14. Кобильник, Т., Когут, У., & Жидик, В. (2021). МЕТОДИЧНІ АСПЕКТИ ВИВЧЕННЯ ОСНОВ АЛГОРИТМІЗАЦІЇ І ПРОГРАМУВАННЯ МОВОЮ PYTHON У ШКІЛЬНОМУ КУРСІ ІНФОРМАТИКИ У СТАРШИХ КЛАСАХ. *Фізико-математична освіта*, *31*(5), 36–44. https://doi.org/10.31110/2413-1571-2021-031-5-006

15. Ковальчук М.Б. Змістові аспекти алгоритмічного мислення. *Фізико-математична освіта.* 2018. Випуск 3(17). С. 61-66.

16. Кузьменко А.В. Огляд навчальних програм з інформатики для учнів старших класів загальноосвітнього навчального закладу. Фізико-математична освіта. 2017. Випуск 3(13). С. 93-99.

17. Мартиненко О., Чкана Я., Удовиченко О. Управління самостійною роботою майбутніх учителів математики у віртуальному навчальному

середовищі через використання електронної версії робочого зошиту. Педагогічні науки: теорія, історія, інноваційні технології. 2020. № 2 (96). С. 144-153.

18. Одінцова О.О. Особливості створення математичних моделей задач, що вивчаються в лінійному програмуванні. Фізико-математична освіта. 2016. Випуск 1(7). С. 105-113.

19. Острога М.М., Шамоня В.Г. Модель формирования готовности будущих бакалавров среднего образования к использованию цифровых технологий в профориентационной деятельности. Science and Education a New Dimension. Pedagogy and Psychology, IX (97), Issue: 246, 2021. P.25-28.

20. Павленко Л.В., Павленко М.П., Хоменко В.Г., Хоменко С.В., Скурська М.М. Інноваційні підходи до вивчення статистики майбутніми ІТ-фахівцями на основі використання мови програмування R. Фізико-математична освіта. 2020. Випуск 1(23). С. 97-105.

21. Петренко С., Петренко Л. Модель формування інформатичної компетентності майбутніх учителів інформатики в процесі фахової підготовки. Педагогічні науки: теорія, історія, інноваційні технології. Суми: СумДПУ імені А. С. Макаренка, 2020. № 2 (96) С. 154-164. DOI 10.24139/2312-5993/2020.02/154-164

22. Петренко С., Петренко Л. Формування готовності майбутніх учителів інформатики до професійної діяльності. Педагогічні науки: теорія, історія, інноваційні технології. Суми: СумДПУ імені А. С. Макаренка, 2019. № 10 (94). С. 95-105. DOI 10.24139/2312-5993/2019.10/095-106.

23. Петренко С.І. Аналіз проблеми безпечної роботи учнів початкових класів у мережі Інтернет // Петренко С.І. / Вісник університету імені Альфреда Нобеля. Серія «Педагогіка і психологія». Педагогічні науки. 2020. № 1 (19) С. 85-92. DOI: 10.32342/2522-4115-2020-1-19-9

24. Петренко С.І., Дегтярьова Н.В. Формування ІКТ-компетентності викладачів на курсах підвищення кваліфікації. Наукові записки Серія:

Педагогічні науки Випуск 186 - Кропивницький: РВВ ЦДПУ ім. В. Винниченка, 2020. с. 150-155.

25. Прошкін В., Хоружа Л., Семеніхіна О. Теорія і практика професійної підготовки майбутніх учителів математики та інформатики засобами цифрових технологій. Теоретичні та практичні аспекти використання математичних методів та інформаційних технологій в освіті й науці: моногр. / за заг. ред. О. Литвин. К.: Київ. ун-т ім. Б. Грінченка, 2021. 332 с. С.48-74.

26. Руденко Ю. О., Дегтярьова Н. В., Юрченко А. О., Семеніхіна О. В. Використання елементів нечіткої логіки у гуманітарних дослідженнях. Збірник наукових праць Національного університету кораблебудування імені адмірала Макарова, 2020. № 1 (479). С. 130-134. https://doi.org/10.15589/znp2020.1(479).17

27. Руденко Ю.О., Дегтярьова Н.В. Електронні ресурси та сервіси інтернет в контексті реалізації електронного навчання. Професійна підготовка вчителя в умовах цифрового освітнього середовища / за заг. ред. О.В. Семеніхіної. Суми, 2020. С.56-86.

28. Семеніхіна О. В., Прошкін В. В., Друшляк М. Г. Використання прийомів мнемотехніки в процесі навчання математики. Математика в рідній школі. 2020. №5 (219). С. 2-7.

29. Семеніхіна О., Юрченко А. Професійна підготовка фахівця: організація онлайн-опитування для визначення потреб у зміні освітньої програми. Освіта. Інноватика. Практика. 2019. Issue 2(6). P. 36-43.

30. Семеніхіна О., Юрченко А., Удовиченко О. Формування умінь візуалізувати начальний матеріал у майбутніх учителів фізики: результати педагогічного експерименту. Професійна підготовка вчителя в умовах цифрового освітнього середовища / за заг. ред. О.В. Семеніхіної. Суми, 2020. С. 99-117.

31. Семеніхіна О.В., Бобровицька С.Ф. Особливості практичної підготовки вчителів до використання ЕОР у початковій школі. Фізико-

математична освіта. 2020. Вип. 1(23). Частина 2. С. 72-77.

32. Семеніхіна О.В., Юрченко А.О., Удовиченко О.М. Формування умінь візуалізувати начальний матеріал у майбутніх учителів фізики: результати педагогічного експерименту. Фізико-математична освіта. 2020. Вип. 1(23). С. 122-128.

33. Семеног О., Семеніхіна О. Медіаосвітні уміння майбутнього вчителя та особливості їх формування у процесі професійної підготовки. Професійна підготовка вчителя в умовах цифрового освітнього середовища / за заг. ред. О.В. Семеніхіної. Суми, 2020. С.118-140.

34. Удовиченко О.М. Критерії та показники рівнів готовності майбутніх учителів інформатики до професійної діяльності. Вісник Черкаського національного університету. Серія «Педагогічні науки». Черкаси, 2020. Вип. 2.2020. С. 142-147.

35. Харченко І.І., Удовиченко О.М. Результати експериментального формування культури професійної комунікації майбутніх фахівців з економіки. Вісник Черкаського національного університету. Серія «Педагогічні науки». Черкаси, 2020. Вип. 1.2020. С. 146-150.

36. Хворостіна Ю.В., Удовиченко О.М., Юрченко А.О. Особливості використання дидактичних ігор на уроках математики. Інноваційна педагогіка. 2019. Вип. 19. Том 3. С. 141-146. https://doi.org/10.32843/2663-6085-2019-19-3-29

37. Чередник І.В., Руденко Ю.О., Семеніхіна О.В. Труднощі навчання учнів системам числення і кодуванню інформації та шляхи їх запобігання. Фізико-математична освіта. 2020. Випуск 2(24). Частина 2. С. 21-27.

38. Шамоня В., Семеніхіна О. Комп’ютерна візуалізація роботи логічних елементів інформаційної системи на базі PROTEUS. Професійна підготовка вчителя в умовах цифрового освітнього середовища / за заг. ред. О.В. Семеніхіної. Суми, 2020. С. 87-98.

39. Шамшина Н.В. Методичні аспекти вивчення СУБД ACCESS: створення інформаційних систем. Професійна підготовка вчителя в умовах

цифрового освітнього середовища / за заг. ред. О.В. Семеніхіної. Суми, 2020. С. 140-178.

40. Юрченко А.О., Семеніхіна О.В., Хворостіна Ю.В., Удовиченко О.М., Петренко С.І. Навчання програмувати в старшій школі крізь призму чинних навчальних програм. Фізико-математична освіта. 2019. Вип. 2(20). Ч. 2. С. 48-55. DOI 10.31110/2413-1571-2019-022-4-021.

41. Юрченко А.О., Семеніхіна О.В., Хворостіна Ю.В., Удовиченко О.М., Петренко С.І. Навчання програмувати в старшій школі крізь призму чинних навчальних програм. Фізико-математична освіта. 2019. Випуск 2(20). Ч. 2. С. 48-55.

42. Юрченко А.О., Удовиченко О.М., Хворостіна Ю.В., Петренко С.І. Дослідження рівня знань майбутніх учителів фізики при використанні цифрових лабораторій. Фізико-математична освіта. 2019. Вип. 4(22). С. 137-141. DOI 10.31110/2413-1571-2019-022-4-021.

43. A History of Visual Programming URL: https://bubble.io/blog/visual-programming/

44. App Inventor Java Bridge URL: http://www.appinventor.org/jbridge

45. App Inventor URL: https://uk.wikipedia.org/wiki/App_Inventor

46. App Inventor URL: https://uk.wikipedia.org/wiki/App_Inventor

47. App Inventor for Android URL: https://en.wikipedia.org/wiki /App_Inventor_for_Android

48. Atamanyuk S., Semenikhina O., Shyshenko I. Theoretical fundamentals of innovation of higher education in Ukraine. Pedagogy and Education Management Review (PEMR). Tallinn, Estonia, 2021. Issue 2(4). P. 30-36.

49. Dehtiarova N., Petrenko S., Rudenko Yu. Pedagogical design in the context of blended learning for future computer science teachers. Modern approaches to the development of knowledge management. Ljubljana. Slovenia. pp. 313-323.

50. Drushlyak M. G., Semenikhina O. V., Kondratiuk S. M., Krivosheya T. M., Vertel A. V., Pavlushchenko N. M. The Automated Control of Students

Achievements by Using Paper Clicker Plickers. MIPRO 2020 : Proceedings of 43 International convention on information and communication technology, electronics and microelectronics, 28 вересня – 2 жовтня 2020, Opatija (Croatia). 2020. P. 688-692.

51. Drushlyak M. G., Shishenko I. V., Borozenets N. S., Nekyslykh K. M., Semenikhina O. V. Computer Probabilistic Models Construction and Analysis of Professional Activity of their Use by Ukrainian Mathematics Teachers. Proceedings of 44 International convention on information and communication technology, electronics and microelectronics "MIPRO 2021", Opatija (Croatia), 28 September – 1 October, 2021. P. 712-717. DOI: 10.23919/MIPRO52101.2021.9596868

52. Drushlyak M., Semenikhina O., Proshkin V., Sapozhnykov S. Training pre-service mathematics teacher to use mnemonic techniques. Journal of Physics: Conference Series. 1840 (2021), 012006. C.1-12 DOI:10.1088/1742-6596/1840/1/012006

53. Kudrina, O., Shpileva, V., Klius, Y., Lavrova, O., Esmanov, O., & Semenikhina, O. Industrial enterprise tax transaction costs planning using digital tools. TEM Journal. 2020. Volume 9(2), P. 619-624. DOI:10.18421/TEM92-26

54. Lazorenko S. A., Semenikhina O. V. Development of Information and Digital Culture of Future Specialists in Physical Culture and Sports as a Modern Problem of Education. Science and Education a New Dimension. Pedagogy and Psychology, VIII (95), Issue 239, 2020 Nov. P. 29-32.

55. MIT App Inventor: Objectives, Design, and Development URL: https://link.springer.com/chapter/10.1007/978-981-13-6528-7_3

56. New clues emerge about Amazon's secretive low-code/no-code project URL: https://www.geekwire.com/2019/aws-everyone-new-clues-emerge-amazons-secretive-low-code-no-code-project/

57. New frameworks for studying and assessing the development of computational thinking URL: http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

58. Okhrimenko O., Semenikhina O., Shyshenko I. Future teachers' readiness for the digital modernization of inclusive education. New challenges in the development of future specialists: collective monograph. Universitatea Dunarea de Jos Galati, Romania, 2021. P. 83-94.

59. Okhrimenko O., Semenikhina O., Shyshenko I. Readiness of future teachers for digital modernization of inclusive education. Innovative Approaches to Ensuring the Quality of Education, Scientific Research and Technological Processes : collective monograph. 2021. No 3.6.15. P. 694-700.

60. Omelyanenko, V., Kudrina, O., Semenikhina, O., Zihunov, V., Danilova, O. & Liskovetska, T. Conceptual aspects of modern innovation policy. European Journal of Sustainable Development. 2020. Volume 9 (2). P. 238-249. DOI:10.14207/ejsd.2020.v9n2p238

61. Ostroha M., Drushlyak M., Shyshenko I., Naboka O., Proshkin V., Semenikhina O. On the use of social networks in teachers' career guidance activities. Smyrnova-Trybulska E. (ed.). (2021) E-learning in COVID-19 Pandemic Time. "E-learning" Series. Vol. 13 (2021) (Pp. 113-124) Katowice-Cieszyn: Studio Noa for University of Silesia.

62. Petrenko S., Dehtiarova N. Increasing teachers' ict-competency level in the after-graduate education process. Інноваційна педагогіка. Вип. 21. Т. 3. 2020. С. 73-77.

63. Repenning, Alexander "Moving Beyond Syntax: Lessons from 20 Years of Blocks Programing in AgentSheets". *Journal of Visual Languages and Sentient Systems*. 2017. С. 68-91 - URL: http://ksiresearchorg.ipage.com/vlss/journal/VLSS2017/vlss17paper_10.pdf

64. Rudenko Yu., Rozumenko A., Kryvosheya T., Karpenko O., Semenikhina O. Online Training during the COVID-19 Pandemic: Analysis of Opinions of Practicing Teachers in Ukraine Proceedings of 44 International convention on information and communication technology, electronics and microelectronics "MIPRO 2021", Opatija (Croatia), 28 September – 1 October, 2021. DOI: 10.23919/MIPRO52101.2021.9596799

65. Rudenko Yu., Semenikhina O. Analysis of distance learning experience in colleges of Sumy region of Ukraine. Education during a pandemic crisis: problems and prospects / Eds. Tetyana Nestorenko & Tadeusz Pokusa Opole, 2020. P. 175-181

66. Rudenko Yuliia, Olha Naboka, Larysa Korolova, Khana Kozhukhova, Olena Kazakevych, Olena Semenikhina. Online Learning With the Eyes of Teachers and Students in Educational Institutions of Ukraine. TEM Journal. Volume 10, Issue 2, P. 922-931. DOI: 10.18421/TEM102-55.

67. Semenikhina O. et al. The Formation of Skills to Visualize by the Tools of Computer Visualization. TEM Journal. 2020. Volume 9(4). P. 1704-1710. DOI: 10.18421/TEM94-51

68. Semenikhina O. V. The Using Interactive Methods In The Formation Of Conflictological Culture Of Specialist. International Scientific Journal «Future Science: Youth Innovations Digest». 2019. Volume 3, Issue 3. P. 44-48

69. Semenikhina O., Drushlyak M., Lynnyk S., Kharchenko I., Kyryliuk H., Honcharenko O. On Computer Support of the Course “Fundamentals of Microelectronics” by Specialized Software: the Results of the Pedagogical Experiment. TEM Journal. 2020. Volume 9 (1). P. 309-316. DOI: 10.18421/TEM91-43

70. Semenikhina O., Drushlyak M., Yurchenko A., Udovychenko O., Budyanskiy D. The use of virtual physics laboratories in professional training: the analysis of the academic achievements dynamics. ICT in Research, Education and Industrial Applications (ICTERI-2020) : 16th International Conference. October, 06-10, 2020. Kharkiv. P. 423-429.

71. Semenikhina O., Proshkin V., Drushlyak M. Mathematical knowledge control automation within dynamic mathematics programs. E-learning and STEM Education / Scientific Editor Eugenia Smyrnova-Trybulska. Katowice–Cieszyn, 2019. P. 571-586. .

72. Semenikhina O., Proshkin V., Naboka O. Application of Computer Mathematical Tools in University Training of Computer Science and Mathematics

Pre-service Teachers. International Journal of Research in E-Learning, 2020, 6(2), 1-23. https://doi.org/10.31261/IJREL.2020.6.2.06

73. Semenikhina O., Yurchenko A., Sbruieva A., Kuzminskyi A., Kuchai O., Bida O. The Open Digital Educational Resources In IT-Technologies: Quantity Analysis. Information technologies and learning tools. V. 75. Issue 1. P. 331-348 https://doi.org/10.33407/itlt.v75i1.3114

74. Semenikhina Olena V., Proshkin Volodymyr V. The main problems of using computer mathematical tools in university education. Інформаційні технології в освіті та науці: Збірник наукових праць. Випуск 12. Мелітополь: ФОП Однорог Т.В., 2021. 204 с. С.9-11.

75. Semenikhina, O., Yurchenko, A., Udovychenko, O., Petruk, V., Borozenets, N., Nekyslykh, K. Formation Of Skills To Visualize Of Future Physics Teacher: Results Of The Pedagogical Experiment. Revista Romaneasca Pentru Educatie Multidimensionala, 2021, 13(2), 476-497. https://doi.org/10.18662/rrem/13.2/432

76. Semenog O., Semenikhina O., Oleshko P., Prima R., Varava O., Pykaliuk R. Formation of Media Educational Skills of a Future Teacher in the Professional Training. Revista Românească pentru Educaţie Multidimensională. 2020. Volume 12. Issue 3, P. 219-245. https://doi.org/10.18662/rrem/12.3/319.

77. Shamonia, V. H., Semenikhina, O. V., Proshkin, V. V., Lebid, O. V., Kharchenko, S. Y., & Lytvyn, O. S. Using the proteus virtual environment to train future IT professionals. CEUR Workshop Proceedings, 2547. P. 24-36.

78. Shishenko I. V., Shamonia V. H., Loboda V. S., Punko V. V., Khvorostina Yu. V. and Voitenko A. A. Studying dynamic mathematics software in the professional training of teachers of computer science, mathematics, and IT specialists. MIPRO 2020 : Proceedings of 43 International convention on information and communication technology, electronics and microelectronics, 28 вересня – 2 жовтня 2020, Opatija (Croatia). 2020. P. 683-687.

79. The maturity of visual programming URL: https://www.craft.ai/blog/the-maturity-of-visual-programming

80. To block or not to block, that is the question: students' perceptions of blocks-based programming URL: https://dl.acm.org/doi/abs/10.1145/2771839.2771860

81. Udovychenko O., Chkana Ya., Yurchenko A., Khvorostina Yu. Introduction of didactic games in the educational process. Фізико-математична освіта. 2019. Вип. 4(22). Частина 2. URL: https://fmo-journal.fizmatsspu.sumy.ua/publ/8-1-0-621.

82. Udovychenko, O. M., Ostroha, M. M., Chernysh, A. E., Kudrina, O. Y., Bondarenko, Y. A., & Kurienkova, A. V. (2020). The use of electronic textbooks in the learning process: A statistical analysis. MIPRO 2020 : Proceedings of 43 International convention on information and communication technology, electronics and microelectronics, 28 вересня – 2 жовтня 2020, Opatija (Croatia). 2020. P. 608-611. doi:10.23919/MIPRO48935.2020.9245146

83. Visual programming language. URL: https://en.wikipedia.org/wiki/Visual_ programming_ language#External_links

84. Voitenko A., Semenikhina O. To the question about inclusive educational space in the training of informatics of children with intellectual disabilities. Education. Innovation. Practice. 2019. Issue 2 (6). P. 6-9.

85. What is visual programming? URL: https://bitspark.de/blog/what-is-visual-programming

86. What Is Visual Programming? URL: https://www.outsystems.com/blog /posts/what-is-visual-programming/

87. Yurchenko A., Drushlyak M., Sapozhnykov S., Teplytska S., Koroliova L., Semenikhina O. Using online IT-industry courses in the computer sciences specialists' training. International Journal of Computer Science and Network Security. Vol. 21 No. 11 pp. 97-104. http://paper.ijcsns.org/07_book/202111/20211113.pdf

88. Yurchenko A., Semenikhina O., Rudenko Yu., Shamonia V. The Digital Technology in IT-Education: the View of Ukrainian University. Збірник наукових праць Національного університету кораблебудування імені адмірала

Макарова, 2020. №4 (482). С. 129-133. https://doi.org/10.15589/znp2020.4(482).15

89. Yurchenko A., Shamonia V., Udovychenko O., Momot R., Semenikhina O. Improvement of Teacher Qualification in the Field of Computer Animation: Training or Master Class? Proceedings of 44 International convention on information and communication technology, electronics and microelectronics “MIPRO 2021”, Opatija (Croatia), 28 September – 1 October, 2021. P. 683-687. DOI: 10.23919/MIPRO52101.2021.9596946

90. Yurchenko A.O., Udovychenko O.M., Rozumenko A.M., Chkana Y.O., Ostroha M.M. (2019). Regional Computer Graphics Competition as a Tool of Influence on the Profession Choice: Experience of Sumy Region of Ukraine. 42nd International Convention on Computers in Education (MIPRO) (May 20 – 24, 2019), Opatija, Croatia, 2019, pp. 909-914.