Програмуємо граючи

НАВЧАННЯ ШКОЛЯРІВ ОСНОВ ПРОГРАМУВАННЯ ЗА ДОПОМОГОЮ СЕРЕДОВИЩА MINECRAFT

ЗМІСТ

ВСТУП

Для сучасної молоді все більш актуальним стає вивчення мов програмування для подальшого працевлаштування. Тому дуже важливо розвивати інтерес школярів до цієї галузі. Це можна робити за допомогою впровадження в освітній процес різних ігрових елементів. Зокрема, при вивченні базових принципів програмування можна використовувати онлайн-тренажери або ігри, наприклад Scratch, CodeMonkey, Tynker, Kodu чи Codecombat. Але підліткам важливо поєднувати ігрові технологім з навчанням. З цією метою у нагоді стане середовищеMineCraft, яке на даний час має потужну освітню компоненту і надає різні можливості для впровадження у навчальний процес. Minecraft - це гра жанру «пісочниця», де немає сюжетної лінії, але є нескінченний світ, в якому можна творити - створювати, будувати, взаємодіяти з іншими гравцями.. Зараз Minecraft це не тільки гра, але й освітня платформа, яку використовують більше ніж в 1000 шкіл по всьому світу: в США, Фінляндії, Швеції, Австралії і навіть Росії. Школярі у цьому середовищі досліджують кораблі, пишуть твори, здійснюють вимірювання, вивчають тривимірну систему координат, будують моделі атомів та молекул, знайомляться з джерелами енергії тощо. Завдяки своїй гнучкості гра легко підлаштовується під різні дисципліни. Гармонійне поєднання ігрових та навчальних технологій дає змогу учням сприймати складний навчальний матеріал у ігровій формі, що більш цікаво та комфортно для них.

На сучасному етапі розвитку шкільної інформатики статус об'єктно-орієнтованого програмування в базовому курсі досі повністю не визначений. Основна мета вивчення об'єктно-орієнтованого програмування - моделювання об'єктів реального світу й опис взаємодії між ними. Об'єктно-орієнтоване програмування є провідним підходом у програмуванні і реалізується сьогодні вже практично в будь-якій сучасній мові програмування. Вивчення даної парадигми програмування передбачає розгляд основних понять об'єктно-орієнтованого програмування (клас; екземпляр класу; характеристики, методи, властивості, індексатори класу; інтерфейси; абстрактні класи), роботи з

великим числом вже створених бібліотек і створенням персональних компонентів.

Широке застосування ідей об'єктно-орієнтованого підходу вимагає глибокого знайомства з його ідеями, а це наштовхує на цілу низку труднощів. По-перше, знайомство з об'єктно-орієнтованим програмуванням зазвичай супроводжується одночасним вивченням нових для учня конструкцій мови програмування. По-друге, часто дуже складно знайти відповідні ретельно продумані матеріали з обговорюваної теми, а також програмні підтримки для закріплення вивченого матеріалу. Існують й інші фактори, що ускладнюють викладання основ об’єктно-орієнтованого програмування. Незважаючи на існування різних ідей з приводу навчання основ об'єктно-орієнтованого програмування, єдина позиція в цьому питанні досі не сформувалася.

Сучасні учителі прикладають багато зусиль для оновлення змісту завдань для школярів, впроваджують інноваційні технології, пристовують завдання до інтересів учнів, використовують ігрові середовища або ігрові завдання. Але наразі бракує методичних розробок з навчання програмуванню, які би враховували інтереси школярів та були доступними для сприйняття. На наш погляд, розробка практичних завдань з навчання мови Python у середовищі MineCraft є цілком доречною і актуальною на даний час.

Об’єкт дослідження: освітній процес у закладах середньої освіти.

Предмет дослідження: навчання школярів основ об’єктно-орієнтованого програмування засобами середовища MineCraft.

Мета дослідження: визначити способи навчання учнів об’єктно-орієнтованому програмуванню, а також розробити практичні завдання з основ об'єктно-орієнтованого програмування для навчання учнів 7-9 класів мови програмування Python у середовищі MineCraft

Відповідно до мети, поставлено такі **завдання**:

1. Розкрити роль ООП у системі інформатичної підготовки учнів 7-9 класів і розглянути методичні аспекти викладання інформатики.

2. Схарактеризувати основні підходи до навчання програмування й описати ООП як сучасну технологію в розробці програмного забезпечення.

3. Розробити практичні завдання з основ об'єктно-орієнтованого мовою Python у середовищі MineCraft.

Методи дослідження: для вирішення поставлених завдань та досягнення мети було використано такі методи дослідження: *теоретичні*: вивчення навчальних програм, підручників, методичних посібників, а також аналіз статей, наукових досліджень, публікацій та метод *проектування*.

Наукова новизна отриманих результатів: *вперше* виділено підходи до навчання програмування (аналіз готових кодів, використання готових блоків програм, використання простих конструкцій, розв’язання комплексних завдань, комбінації підходів)*; уточнено* роль ООП у системі інформатичної підготовки учнів 7-9 класів*; запропоновано* практичні завдання для навчання основ об’єктно-орієнтованого програмування учнів

Практичне значення отриманих результатів полягає в тому, що розроблені практичні завдання можуть бути використаними для навчання об’єктно-орієнтованого програмування на уроках інформатики у 7-9 класах.

РОЗДІЛ 1. СУЧАСНІ ПІДХОДИ ДО НАВЧАННЯ ШКОЛЯРІВ ОСНОВ ПРОГРАМУВАННЯ

1.1. Об’єктно-орієнтоване програмування як сучасна парадигма програмування

Навчання програмування – найбільш розроблена частина методики інформатики й має давню історію. Перші роки після введення в школі інформатики як обов'язкового предмета, школярі вивчали, в основному, саме програмування. Тому існує велика кількість методичних розробок та навчальних програм різних курсів програмування. Ускладненою обставиною для вчителя при виборі такого курсу є те, що існують різні парадигми програмування і необхідно визначитися з вибором мови або системи програмування.

Певний спосіб мислення (парадигма) служить основою для створення мови програмування. Існують різні парадигми програмування - процедурна, об'єктно-орієнтована, функціональна, логічна. Кожна з парадигм використовується для розв’язання певного класу задач. Деякі мови підтримують кілька парадигм, інші ж, навпаки, орієнтовані на реалізацію тільки однієї парадигми. Для кожної мови програмування одна з парадигм є основною, а інші парадигми - додатковими.

Як правило, для початківців першою парадигмою стає процедурна. У межах процедурної парадигми програміст складає послідовність викликів процедур (функцій) для виконання поставленого завдання. При цьому будь-які процедури (функції) можуть використовувати будь-які дані за умови відповідної кількості і типів параметрів. Така ситуація виникає у зв'язку з тим, що дані і процедури (функції) для їх обробки не пов'язані за змістом; отже, неможливо захистити дані від неправильного використання. Крім того, застосування процедурного програмування пов'язане з певними труднощами для створення великих програмних систем (великі програми важко створювати, налагоджувати, супроводжувати).

Об'єктно-орієнтоване програмування (ООП) є результатом розвитку процедурного програмування, проте пропонує інший підхід до розробки програм. В ООП дані і методи об'єднуються в класи, тобто між ними встановлюється зв'язок. На основі класів створюються об'єкти - головні елементи програми. У процесі виконання програми об'єкти взаємодіють між собою, тобто обмінюються інформацією. ООП дозволяє також подолати труднощі при створенні складних програм.

Сьогодні знання ООП визначає успіх у багатьох галузях професійної діяльності. Воно займає провідне місце в розробці сучасних програмних засобів і тому ознайомлення з ним необхідне для учнів, орієнтованих на професію програміста. Вивчення подібного профільного курсу спрямоване на виконання таких завдань:

• засвоєння методології об'єктно-орієнтованого програмування;

• вивчення техніки програмування на одній з мов;

• розширення загального кругозору учнів.

Шкільний курс об’єктно-орієнтованого програмування, з одного боку, повинен бути сучасним, а з іншого - бути елементарним і доступним для вивчення. Поєднання цих двох, багато в чому суперечливих, вимог є складним завданням.

Наразі принципи ООП в основному пояснюються на прикладах фрагментів консольних додатків у тексті підручника. На практиці відразу пропонується розробка віконних додатків. У зв'язку з цим принципи ООП так і залишаються деякою абстракцією, не випробувані учнями. Зрозуміло, що трудомісткість розробки, складність матеріалу і звичайна нестача часу не дозволяють запропонувати учням з різними здібностями розробляти з нуля програми з використанням власних класів.

За класифікацією Ю.К. Бабанського [14], у групі методів навчання для організації і здійснення навчально-пізнавальної діяльності виділяються наступні методи: перцептивні, словесні, наочні, практичні, логічні, гностичні та самоврядування навчальними діями. найбільш оптимальним при вивченні

складних тем з програмування, зокрема принципів ООП, є поєднання наочного, практичного та гностичного методів навчання.

Практичний метод навчання може бути реалізований за допомогою вправ, лабораторних робіт, практичних робіт або пізнавальних ігор. Зазвичай проводять практичні роботи, які спрямовані на формування вміння застосовувати знання на практиці. Практична робота проводиться після вивчення теоретичного матеріалу з основ ООП і включає:

- попередній інструктаж;
- виконання роботи;
- поетапний контроль виконання з озвучуванням необхідного результату;
- перевірку та оцінювання.

Безумовно, що при розгляді технології ООП повинна бути його реалізація в інтегрованому середовищі розробки програмного забезпечення. При використанні візуального середовища з'являється можливість проєктувати деяку частину, наприклад, інтерфейси майбутнього продукту із застосуванням візуальних засобів додавання і налаштування спеціальних бібліотечних компонентів.

1.2. Місце вивчення основ об’єктно-орієнтованого програмування у шкільному курсі інформатики

Зосередимо увагу на освітню програму з інформатики для учнів сьомих класів [13]. На цьому етапі не ставиться завдання глибокого вивчення ІКТ, акцент зроблено на набутті навичок практичного застосування інформаційно-комунікаційних технологій, а також на розвивальній спрямованості навчання. На вивчення інформатики у 7 класі передбачено 35 годин, які розподіляються на три великі блоки: служби Інтернету, опрацювання табличних даних, алгоритми та програми. Зупинімося більш детально на останньому блоці «Алгоритми та програми». В освітній програмі зазначено, що учень повинен уміти пояснити такі поняття:

- знати, що таке величина, змінні, вказівка присвоювання;
- знати, як створювати алгоритми і програми з використанням змінних і різних алгоритмічних структур (лінійних, розгалужень і повторень);
- уміти використовувати різні алгоритмічні структури та змінні для розв'язання навчальних і життєвих задач.

Відповідно до освітньої програми восьмого класу [13] на уроки інформатики виділяється вже 70 годин. Але й кількість блоків маємо більшу: кодування даних та апаратне забезпечення, опрацювання текстових даних, створення та публікація веб-ресурсів, опрацювання мультимедійних об'єктів, алгоритми та програми. У восьмому та дев'ятому класі відбувається повноцінне формування предметних ІТ-компетентностей. У восьмому класі учень повинен:

- розуміти призначення мови програмування та основних її елементів, а також уміти наводити приклади сучасних мов програмування;
- знати відмінність між змінними та константами;
- уміти порівнювати особливості різних середовищ програмування;
- розуміти поняття об'єкта в мові програмування, його властивостей і методів;
- уміти пояснювати структуру програми;
- знати функції елементів графічного інтерфейсу та користуватися ними;
- уміти розрізняти властивості і методи елементів управління;
- уміти планувати процес розв'язування задачі з використанням програмування;
- створювати і налагоджувати програми, зокрема подійно-орієнтовані та об'єктно-орієнтовані;
- використовувати в програмах вирази, коректно добирати типи даних;
- уміти розв'язувати задачі з використанням усіх базових алгоритмічних структур, змінних та констант;
- обґрунтовувати вибір типів даних для розв'язування задачі.

У дев'ятому класі учні вже мають більш глибокі знання з блоку «алгоритми і програми», а також знайомляться з новим блоком «бази даних та

системи управління базами даних». На вивчення інформатики передбачено 70 годин,і складається з п’яти блоків [13]: програмне забезпечення та інформаційна безпека, 3D-графіка, опрацювання табличних даних, бази даних та СУБД, алгоритми та програми. У дев’ятому класі учень має:

- уміти пояснювати принцип організації даних за допомогою одновимірних масивів;
- знати та вміти пояснити поняття масиву, елемента масиву, індексу та значення елемента;
- уміти описувати алгоритми опрацювання елементів масиву, що задовольняють певній умові;
- уміти описувати алгоритм знаходження підсумкових величин масиву;
- описувати алгоритми знаходження підсумкових величин масиву.

Час, виділений на кожну тему, визначається вчителем відповідно до рівня попередньої підготовки учнів, а також обраної методики навчання. Однак потрібно зауважити, що такі теми, як «моделювання» та «алгоритми та програми», вивчаються не менше 40 % навчального часу в 5–8 класах і не менше 30 % у 9 класі. Послідовність вивчення тем може змінюватись учителем з необхідності.

1.3. Підходи до навчання програмування

На основі аналізу методичних посібників, навчальних підручників, досвіду учителів-практиків, які висвітлюють рекомендації на Інтернет-ресурсах, можна виділити кілька підходів, за якими здійснюється навчання основ програмування:

- аналіз готових фрагментів коду;
- використання готових блоків програм («будівельних блоків»);
- використання простих конструкцій;
- розв’язання комплекстних завдань;
- комбінації підходів.

1. Аналіз готових фрагментів коду.

При підході до навчання програмування, заснованому на аналізі коду, учні читають і опановують логіку програмування, перш ніж писати свою власну. Цей підхід заснований на використанні псевдокоду, тому він не залежить від мови програмування.

Здатність пояснювати логіку програмування та код, мабуть, є необхідною умовою, хоча і не має на увазі здатності писати код. Такий підхід дозволяє розвивати навички розв’язання проблем, які можуть застосовуватися в багатьох галузях. Цей підхід може розчарувати учнів, які хочуть працювати з комп'ютерами, і може не підходити для ситуацій незалежного навчання через відсутність зворотного зв'язку. Підхід до аналізу коду аналогічний навчанню читання перед навчанням письма. Навчання читання розкриває правила граматики мови, що вивчається. Спроба вивчення мови таким чином повинна дозволити учню познайомитися з тим, як компоненти і конструкції мови комбінуються для створення сенсу. А вже потім учень може намагатися будувати свої конструкції.

Реалізація цього підходу може включати надання учням практичних вправ. Ці вправи можуть бути підготовлені із використанням прийнятного псевдокоду. Вправи можуть існувати тільки на папері або відображатися у відповідному середовищі розробки, якщо така існує для реалізації псевдокоду. Однак, як індивідуальний підхід, аналіз коду не вимагає, щоб учні взаємодіяли з кодом у середовищі комп'ютера.

Чотири нещодавних дослідження підтверджують використання підходу до аналізу коду. В Університеті Вікторії в Мельбурні Мілішевська і Тан (2007) [4] змінили дизайн одного з курсів з інформатики першого року, включивши в нього об'єктно-орієнтовану парадигму, але спочатку навчили структурного програмування. Новий підхід до викладання курсу включає вивчення прикладів добре написаного коду. Вони припускали, що студенти будуть учитися, наслідуючи приклади гарної практики.

Цей підхід також підтримують Келлинг і Розенберг (2001) [3], які є прихильниками вивчення коду для засвоєння стилів і використання ідіом.

Кемпбелл і Болкер (2002) [5] погоджуються, виходячи з власного переконання, що студенти дізнаються більше, читаючи програми, написані досвідченими програмістами, ніж починаючи з написання власного коду. Проходження коду і читання коду, обидва елементи конструктивістського навчання комп'ютерного програмування, були використані Луї та ін. (2004) в їх дослідженні студентів, які вивчають більш слабке програмування [4]. Вони припускають, що цикл редагування, компіляції та виконання може виснажити терпіння і впевненість слабших учнів. Вони виступають за те, щоб слабші учні працювали з папером і олівцем.

Звичайно, разом ці результати можуть просто означати, що здатність писати код вимагає певних навичок у поясненні, але ця навичка може не породжувати здатності писати код. Хоча з точки зору учня навчання програмування без використання комп'ютера може здатися дивним, введення аналізу коду із самого початку має свої переваги. По-перше, цей підхід означає, що немає інструментів або середовищ, які потрібно освоїти при підготовці до вивчення логіки програмування. По-друге, можна використовувати будь-яку реалізовану мову програмування, але, можливо, найбільш зручним є відсутність певної мови взагалі. По-третє, аналіз, у першу чергу, дозволяє розвинути навички, пов'язані з налагодженням і трасуванням, щоб виправити або гарантувати поведінку програми. По-четверте, це також дає можливість ознайомити учнів з логікою простих базових алгоритмів, таких як лінійний пошук і бульбашкове сортування. Раннє знайомство з аналізом коду може допомогти підготувати учнів до іспитів, які включають питання, пов'язані з відстеженням або пробним запуском коду. Але в залежності від віку та здібностей учнів викладачеві може бути складно розробити набір значущих і зрозумілих інструкцій псевдокоду. Наприклад, чи слід використовувати слова або символи?

Спочатку аналіз може бути проблематичним для незалежного навчання через відсутність негайного зворотного зв'язку. Крім того, існує ймовірність того, що розуміння псевдокоду може не привести до навичок, які можна

перенести на розуміння мови. Хоча підхід до аналізу коду доцільний не у всіх ситуаціях, він корисний для початківців програмістів. Його перевага полягає в тому, що він не вимагає певної мови програмування або комп'ютерного середовища. Він також сприяє розвитку навичок розв’язання проблем і логічного мислення, необхідних у багатьох галузях. Його використання може поліпшити здатність розуміти код, який становить частину базової основи, необхідної для полегшення написання програми.

2. Використання готових блоків програм («будівельних блоків»)

У підході до навчання програмування, заснованому на будівельних блоках, учні розвивають розуміння окремих частин, перш ніж об'єднувати їх для створення сенсу. Це вимагає набору інструментів розробки і певної мови програмування. Мовні конструкції вводяться і розуміються окремо, перед їх об'єднанням. Такий підхід не обмежується одномовними парадигмами і використовується в середовищах процедурних, функціональних і об'єктно-орієнтованих мов.

Підхід, заснований на будівельних блоках, має перевагу введення точно певного синтаксису мови і негайного зворотного зв'язку від інструменту перевірки синтаксису. На жаль, навчитися користуватися редактором коду, засобом перевірки синтаксису і, можливо, компілятором може бути складно для новачків. Крім того, той факт, що синтаксично правильний код часом не призводить до правильної логіки, викликає в учнів розчарування. Підхід, заснований на будівельних блоках, аналогічний навчанню говорити на окремих частинах мови перед їх об'єднанням спочатку в усній формі, а потім у письмовій. Учні зазвичай набувають усних мовних навичок раніше, ніж письмових. При спробі оволодіти мовою, окремі компоненти (іменники, дієслова, прикметники) деякою мірою розуміються, перш ніж з'єднати їх разом, щоб створити речення. Складний зміст будується на розумінні більш дрібних частин. Для цього підходу потрібен спеціально підібраний набір інструментів, що складається з мови програмування і середовища розробки. Передбачається, що вибрана мова і середовище відповідають можливостям учнів. Розуміння

поведінки конструкцій буде покращено за рахунок написання конструкцій у відповідному середовищі розробки, в якій інструмент може виділяти синтаксичні помилки. Крім того, виконання цих окремих конструкцій можливе в середовищі розробки, де учні в змозі зрозуміти, можливо, за допомогою візуалізації, поведінку кожної конструкції.

Той же підхід можна використовувати з об'єктно-орієнтованою парадигмою, як показав Саяніемі і Ху (2006). [9]. Вони вважають за краще представляти поведінку змінних і керуючих структур до об'єктів. Цей підхід включає в себе переваги раннього введення синтаксису мови і раннього впровадження інструментів, які допомагають при програмуванні. Різноманітні конструкції й поняття мови можна навчати ізольовано. Подальше вивчення різних конструкцій може бути виконано шляхом демонстрації того, як змінювати їх поведінку, змінюючи їх синтаксис. Цей підхід також забезпечує виконання розроблених алгоритмів, що складаються з декількох блоків, за умови їх синтаксичної правильності. Це, звичайно, не означає, що будь-яка конкретна комбінація блоків буде логічно правильною. Крім того, освоєння інструментів, необхідних для відображення і перевірки поведінки блоків, надає учням інструменти і стратегії налагодження, які будуть потрібні для розробки реальних рішень.

Помилки в реалізації синтаксису мови програмування призвели до розчарування Аль-Імамі, Алізаде і Нура (2006) [10]. Щоб подолати це, вони розробили середовище, в якому використовуються шаблони конструкцій, в яких учні заповнюють прогалини. Наприклад, шаблон умовного блоку може бути представлений з правильною структурою синтаксису. Учневі потрібно тільки заповнити необхідне поле.

Підхід, заснований на будівельних блоках, може створити проблеми для учнів, які тільки починають своє навчання. Інтерактивне навчання вимагає оволодіння деякими складними інструментами одночасно з вивченням поведінки блоків. Але окремі блокові конструкції можуть не виконуватися або не мати сенсу для виконання, навіть, якщо синтаксис правильний. Наприклад,

порожнє повторення або порожній умовний вираз синтаксично дозволені в деяких мовах, але не мають логічного сенсу. Освоєння поведінки окремих блоків може не перейти до конструктивної діяльності, необхідної для створення алгоритму, що виконує важливу задачу. Учень може пояснити і зрозуміти концепцію, але не зможе застосувати це розуміння при побудові алгоритму. Крім того, учням, можливо, буде важко зрозуміти відносини і взаємодії між блоками без контексту проблеми, яку потрібно вирішити. Проте, введення простого контексту проблеми може стати наступною дією.

Хоча використання підходу будівельних блоків часом є складним завданням як для учнів, так і для викладачів, воно передбачає раннє введення точно певного синтаксису мови і раннє впровадження набору інструментів для допомоги в написанні коду. Подання кожної мовної конструкції по черзі дозволяє природним чином досліджувати її, поки не стане зрозуміла її поведінка. Як тільки базова поведінка стає зрозумілою, можна проводити експерименти зі зміни синтаксису конструкцій для поглиблення навчання.

3. Використання простих конструкцій.

У підході до навчання програмування з використанням простих конструкцій, учні засвоюють стандартні фрази, використовуючи обмежений набір операцій та методів. Ці багаторазові блоки можуть зберігатися в наборі інструментів програміста для використання в більш складних рішеннях. Невеликі, вузькоспрямовані фрагменти можуть використовуватися для об'єднання конструкцій з метою створення одиниць, які корисні в більш великих рішеннях. Як і у випадку з будівельними блоками, підхід простих одиниць вимагає одночасного освоєння інструментів і конструкцій. Крім того, можна освоїти нові навички вирішення проблем.

У цьому підході конструкції об'єднуються в багаторазово використовувані блоки коду. Наприклад, набір конструкцій можна якимось чином об'єднати для сортування списку або знаходження максимального набору чисел. Невеликі, чітко визначені проблеми, такі як змусити візуальний об'єкт відскакувати, коли він торкається стіни лабіринту, можуть бути вирішені

з використанням невеликої кількості конструкцій. Самі по собі ці агрегати можуть здатися банальними. Однак вони стають більш потужними, коли об'єднуються разом для вирішення більш великих проблем. Як і в разі підходу з використанням будівельних блоків, учням обов'язково потрібно середовище розробки, і, на жаль, вони зіткнуться з усіма проблемами, пов'язаними з ним.

Перевага цього підходу - можливість вводити невеликі, керовані і вузькоспрямовані контексти. Наприклад, при введенні конструкції повторення може бути використана концепція середнього значення. Побудова простого модуля, який знаходить середнє значення набору чисел, може бути легко записана з використанням конструкції повторення, але це також приведе до отримання корисного сегмента коду, який можна включити в більш складні програми. Об'єднання окремих конструкцій приводить до простих одиниць, комбінування простих одиниць приводить до більш складних одиниць. Як і в разі підходу з будівельними блоками, інтерактивне навчання вимагає оволодіння деякими складними інструментами одночасно з навчанням конструювання простих одиниць. Це часто є перешкодою для новачків. Більш того, оволодіння практичними навичками вирішення проблем, необхідних для розв’язання поставлених завдань, повинно відбуватися одночасно з вивченням інструментів і оволодінням поведінкою блоків. Також з використанням цього підходу розвивається корисна навичка декомпозиції - розбиття проблеми на невеликі розв'язні частини.

4. *Розв’язання комплексних завдань*

При повному системному підході до навчання програмування учні відразу ж занурюються у повне використання мовних конструкцій та інструментів. Цей підхід передбачає нетривіальне рішення проблеми. Концепції програмування і мовні конструкції вводяться в міру необхідності для вирішення проблеми. Хоча це може здатися нелогічним, деяких учнів мотивують реальні проблеми і їх повне рішення. Цей підхід, як і деякі з перелічених вище, вимагає одночасного володіння безліччю навичок та інструментів. Повноцінний системний підхід аналогічний вивченню мови

методом занурення. При повному системному підході учні розробляють або допомагають розробити рішення нетривіальної проблеми.

У концепції програмування мовні конструкції вводяться тільки тоді, коли рішення проблеми вимагає їх застосування. Наприклад, гра в хрестики-нулики для двох може служити вступним завданням. Для цього учням потрібно продемонструвати навички декомпозиції (обробки введення з клавіатури, відображення результатів у будь-якому форматі, відстеження ходів (змінних / присвоєнь), визначення доступності позиції та ідентифікації виграшного ходу). Конструкції мови будуть введені тільки тоді, коли необхідно вирішити певну частину проблеми.

Дослідження довели успішність використання повного системного підходу, що включає вирішення проблем до навчання програмування. Duke та ін. (2000) застосовують цей підхід у своїх класах для початківців з Java. Щоб приховати деякі об'єктно-орієнтовані складності, вони надають набір індивідуальних класів для своїх учнів. В одному завданні учнів попросили реалізувати гру «4 в ряд». Конструкції, які повинні були освоїти студенти, полягали у використанні повторення і логічних виразів для підтримки внутрішньої логіки системи. Студенти повідомили, що такий підхід дозволяє їм ефективно вивчати Java.

Метою Нуутіла та ін. (2008) [8] є придбання навичок проєктування систем. Своїм учням вони ставлять такі завдання, як програмування руху робота у лабіринті. Очікується, що учні розберуться з проблемою, спроєктують заняття і розроблять алгоритми для створення ефективного вирішення. Цікаво, що фактичне створення рішення не завжди є частиною завдання, хоча в курс також входить практика використання реальних інструментів програмування.

Розповідь історій з Алісою, світ тривимірного візуального програмування, - це контекст, обраний Саттар і Лоренценом (2009) [10] для ознайомлення своїх учнів з програмуванням з використанням повносистемного підходу. Студентам надається сценарій розкадровки, і очікується, що вони

розроблять код для анімації історії. Поняття (об'єкт, світ і сцена) і конструкції вводяться, коли це необхідно для просування анімації.

Інший приклад повносистемного підходу використовується Кемпбеллом і Болкером (2002) [12]. Вони використовують імерсивні техніки і просять своїх новачків прочитати і змінити симуляцію банкомату. Їх підхід відразу фокусується на інтерфейсах, архітектурі і дизайні. Синтаксис вибраної мови програмування розглядається тільки за необхідності. Вони визнають, що це непростий підхід, але він приділяє більше уваги дизайну і навичкам, ніж оволодінню синтаксисом.

Перевага полягає в тому, що учні можуть зіткнутися з реальними проблемами, для вирішення яких у них вже мають бути концептуальні моделі, наприклад торговий автомат або система телефонного білінгу. Крім того, вони можуть відчувати себе мотивованими і натхненними тим, що вчаться програмувати систему, яка представляє реальне життя. Наприклад, може знадобитися змінити код торгового автомата, щоб отримати правильну решту від монети 2 фунти стерлінгів. Якщо учень зможе внести цю зміну, то вся система стане більш функціональною й ефективною.

За визначенням, занурення або повний системний підхід вимагає одночасного оволодіння безліччю навичок. Інструменти середовища програмування, поведінка окремих блоків, взаємодія між блоками і налагодження повинні розглядатися разом. Це може деморалізувати деяких новачків. З точки зору вчителя, вибір завдань повинен бути добре продуманий, щоб виконати основні вимоги із засвоєння необхідних конструкцій і придбання навичок для роботи в середовищі програмування. Хоча це може здатися нелогічним, повний системний підхід може мотивувати деяких учнів. Розуміння того, як зміни в коді підвищують ефективність рішення, може швидко підштовхнути деяких учнів до подальшого дослідження. Реальні проблеми в уже відомому контексті спонукають розвинути більш досконалі навички вирішення проблем, такі як декомпозиція. Хоча, і за підходів з простими модулями і будівельними блоками необхідно освоїти певний

синтаксис мови та інструменти розробки, цей підхід має ту перевагу, що з самого початку працює над нетривіальними рішеннями.

5. *Комбінації підходів*

Комбінування різних підходів забезпечує прогрес у становленні ефективного програміста. Різні набори інструментів дозволяють використовувати різні шляхи в залежності від можливостей і потреб учня. Здатність писати код вимагає певних навичок у відстеженні і поясненні коду.

Ефективні програмісти демонструють навички аналізу коду (відстеження і пояснення), розуміння поведінки блоків (відстеження і пояснення), побудови простих одиниць (написання) і комбінування простих одиниць для створення повних систем (написання). Передбачається, що, комбінуючи підходи до навчання програмування, учні зможуть продемонструвати прогрес у становленні ефективних програмістів, незалежно від їх віку та здібностей.

Лінійний підхід від будівельних блоків до простих одиниць і до повних систем – це, мабуть, найбільш очевидна лінія розвитку, оскільки вона являє собою поетапний рух у складності. Однак зворотний шлях також є типом прогресу і може сприяти придбанню або поглибленню знань учнів. Порядок підходів залежать тільки від відправної точки. Остаточне рішення про використовувані підходи і порядок їх викладання повинні бути обрані залежно від вимог курсу, віку учнів, середовища або мови програмування.

РОЗДІЛ 2. РОЗРОБКА ПРАКТИЧНИХ ЗАВДАНЬ ДЛЯ НАВЧАННЯ ШКОЛЯРІВ ОСНОВ ПРОГРАМУВАННЯ МОВОЮ PYTHON В СЕРЕДОВИЩІ MINECRAFT

2.1. Специфіка середовища Minecraft для навчання програмування

Основи програмування - один з найскладніших розділів шкільного курсу інформатики. Одним із завдань навчання інформатики в основній школі є розвиток алгоритмічного мислення як засобу планування та організації діяльності. Тому основи алгоритмізації та програмування становлять одну із змістових ліній курсу інформатики, яка є наскрізною для всього курсу. Однією з вагомих проблем вивчення програмування інформатики є вибір середовища програмування. Доцільно розглядати процедурну об'єктно-орієнтовну парадигму програмування, прикладом якої мова Python. Ми вважаємо, що одним з цікавих шляхів вивчення мови Python – є залучення середовища Minecraft.

Перша версія вийшла у 2008 році. Гравці можуть створювати і руйнувати різні об'єкти в тривимірному навколишньому середовищі, використовуючи готові блоки. У 2013 році вийшла версія Minecraft для Raspberry Pi, яка була створена разом з бібліотекою Python. Вбудований в гру API дозволяє взаємодіяти з ігровим світом, тим самим навчаючи програмування. API складається з трьох стандартних окремих бібліотек, всі вони призначені для різної взаємодії з ігровим клієнтом. Початкова підтримка була надана лише для Python 2, але під час конференції на PyconUK 2014 бібліотека була перенесена на Python 3 і створено py3minepi [15]. Нажаль, в процесі було порушено багато зворотної сумісності.

Можливість використовувати Minecraft з Python була дуже популярною, і незабаром був створений плагін RaspberryJuice для Minecraft Java Edition. Цей плагін є ключем з'єднання гри з бібліотеками як py3minepi. RaspberryJuice також розширив оригінальний API, додавши додаткові функції. Мартіном Он'Хонелом було створено бібліотеку mcpi. Ця бібліотека підтримує Python 2 та Python 3 та Minecraft: Pi edition та плагін RaspberryJuice. Бібліотека вийшла в

реліз на сайті PyPl в травні 2018 року. На сьогодні це найсучасніша бібліотека для роботи з Minecraft. Ця бібліотека пропонує шість різних класів для взаємодії з клієнтом гри.

Клас Minecraft - це основний клас для взаємодії з грою, який містить чотири підкласи : Camera, Entity, Events, Player. Block.py та event.py – додаткові бібліотеки для опису типа блока зі зберіганням констант кожного блоку та перевірки подій що сталися з блоком

Оскільки мова програмування Python дуже гнучка, і гра Minecraft має відкрите з'єднання, було створено ще багато додаткових бібліотек [16] для спрощення деяких операція, або навпаки, для більш функціонального програмування.

Бібліотека minecraftstuff - стороння бібліотека для взаємодії з грою. Бібліотека забезпечує функції для малювання ліній, створення, переміщення та обертання фігур та модуль черепахи. У додатку А висвітлимо характерні оператори бібліотеки.

2.1. Розробка тренувальних завдань для навчання школярів основ програмування в середовищі Minecraft

Завдання №1. Перевірка уведених даних.

Користувачеві пропонується ввести числові значення за допомогою input() в программі Python, які присвоються відповідній координаті гравця Minecraft. Проаналізувати різницю між командами postToChat() та print(). Перевірити введені данних.

Приклад вікна із завданням зображений на рис.2.1.а та рис.2.1.б

Забезпечити:

1. Перевірка уведеного значення Y.

```
import mcpi.minecraft as minecraft
mc = minecraft.Minecraft.create()

x = int(input("Введіть значення X = "))

y = int(input("Введіть значення Y = "))
while y<0:
    print("Невірне значення Y")
    y = int(input("Введіть значення Y = "))

z = int(input("Введіть значення Z = "))

mc.player.setPos(x,y,z)
mc.postToChat("Teleported to "+str(x)+" "+str(y)+" "+str(z))
print("Гравця телепортовано на координати ",x,y,z)
```

```
Введіть значення X = 1000000
Введіть значення Y = 100
Введіть значення Z = 20000
Гравця телепортовано на координати  1000000 100 20000
```

Рисунок 2.1.а Приклад виконання завдання

Рисунок 2.1.б Приклад виконання завдання в Minecraft

Завдання №2. Знайти помилку у коді

Дано фрагмент кода (рис 2.2.а), в якому потрібно знайти помилку, яка заважає роботі програми. Наприкінці потрібно отримати результат, зображений на рис.2.2. Забезпечити:

1. Оцінювання правильності виконаного завдання таким чином, щоб при запуску правильного коду виводилося повідомлення з текстом «Все працює!»;

```
import mcpi.minecraft as minecraft
mc = minecraft.Minecraft.create()

x = 100
y = 200
z = "30"

mc.pleyr.setPos(str(x),z,c)
print("Все працює!")
```

```
import mcpi.minecraft as minecraft
mc = minecraft.Minecraft.create()

x = 100
y = 200
z = 30

mc.player.setPos(x,y,z)
print("Все працює!")
```

Рисунок 2.2 – Початкова форма завдання і результат

Завдання №3. Змінити позицію гравця на випадкове значення

Створити програму, яка змінює позицію гравця на випадкове значення 3 рази з інтервалом 10 секунд. Приклад виконання завдання зображений на рис.2.3

Рекомендації:

Для виконання завдання доцільно скористатися:

Модуль random, в якій знаходиться функція randint(a,b);

Модуль time, в якій знаходиться функція sleep(m);

Завдання №4. Вивести координати гравця

У наданому коді (рис. 2.4.а) потрібно додати команди для виведення координат гравця, отриманих за допомогою getTilePos(). Приклад виконаного завдання зображений на рис 2.4 б.

```
import mcpi.minecraft as minecraft
from random import *
from time import *
mc = minecraft.Minecraft.create()

x = randint(-100000,100000)
y = randint(70,200)
z = randint(-100000,100000)
mc.player.setPos(x,y,z)
sleep(10)
x = randint(-100000,100000)
y = randint(70,200)
z = randint(-100000,100000)
mc.player.setPos(x,y,z)
sleep(10)
x = randint(-100000,100000)
y = randint(70,200)
z = randint(-100000,100000)
mc.player.setPos(x,y,z)
sleep(10)
```

Рисунок 2.3 – Приклад програми до завдання 3

Забезпечити:

Оцінювання правильності виконаного завдання таким чином, щоб при запуску правильного коду виводилося 3 повідомлення в грі з координатами гравця.

Рекомендації:

Скористатися командами команд getTilePos() та postToChat() :

Використовувати зміну pos для створення списку координат гравця;

```
import mcpi.minecraft as minecraft
mc = minecraft.Minecraft.create()

pos= mc.player.getTilePos()
x = pos.x
y = pos.y
z = pos.z
```

```
import mcpi.minecraft as minecraft
mc = minecraft.Minecraft.create()

pos= mc.player.getTilePos()
x = pos.x
y = pos.y
z = pos.z

mc.postToChat(x)
mc.postToChat(y)
mc.postToChat(z)
```

Рисунок 2.4.а – Приклад коду до завдання 4

Завдання №5. Відстежити відстань, яку проходить гравець за 20 секунд

Створити програму, за допомогою якої можна відстежити відстань, яку проходить гравець за 20 секунд вільної гри в Minecraft. Приклад виконаного завдання зображений на рис. 2.5.

Виконати завдання таким чином:

1. Створити зміну pos1 з координатами гравця при запуску програми.
2. Записати окремо кожну координату в зміну(X1,Y1,Z1).
3. Використати модуль time з функцією sleep().
4. Створити зміну pos2 з координатами гравця.
5. Записати окремо кожну координату в зміну(X2,Y2,Z2).
6. Створити нові зміні – різниці координат, від координати з pos1 відняти координату з pos2. Приклад : *distX=x2-x1;*
7. Вивести в гру кожну різницю координат.

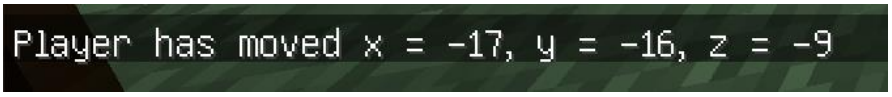

Рисунок 2.5 – Приклад виконаного завдання 5 в Minecraft

Завдання №6 Змінити координати гравця

Користувачеві пропонується ввести, яку координату потрібно змінити та значення, на яке змінити. Це значення присвоється відповідній координаті гравця Minecraft.

исвітлення зміни координати гравця відносно свого місцезнаходження та різниці здійснюється за допомогою команд getTilePos() та getPos(). Приклад вікна із завданням зображений на рис.2.6

```
import mcpi.minecraft as minecraft
mc = minecraft.Minecraft.create()

pos= mc.player.getPos()
x = pos.x
y = pos.y
z = pos.z


a = input()
if a=='x':
    mc.player.setPos(x+int(input()),y,z)
elif a=='y':
    mc.player.setPos(x,y+int(input()),z)
else:
    mc.player.setPos(x,y,z+int(input()))
```

Рисунок 2.6 – Приклад завдання 6

Завдання №7. Створення нового блоку

Користувачеві пропонується ввести ідентифікатор блоку в программі Python. Після цього, в світі Minecraft, поруч з гравцем, створиться блок згідно цього ідентифікатору. Переглянути параметри команди Приклад виконаного завдання зображений на рис.2.7.а та 2.7.б

Рекомендації:

Використати список ID блоків.

```
import mcpi.minecraft as minecraft
mc = minecraft.Minecraft.create()

pos= mc.player.getPos()
x = pos.x
y = pos.y
z = pos.z


ID = int(input())
mc.setBlock(x+1,y,z,ID,0)
```

Рисунок 2.7.а – Приклад коду до завдання 7

Завдання №8.Створення барєру

Створити програму, яка створює навколо гравця бар’єр висотою 1 блок з дерева. Приклад виконаного завдання зображений на рис. 2.8.

Виконати завдання таким чином:

1. Створити зміну pos з координатами гравця.

2. Записати окремо кожну координату в зміну(X,Y,Z).

3. Додати 8 команд setBlock(x,y,z,5,0), де x,y,z – відповідно будуть змінюватись згідно позиції поруч з гравцем.

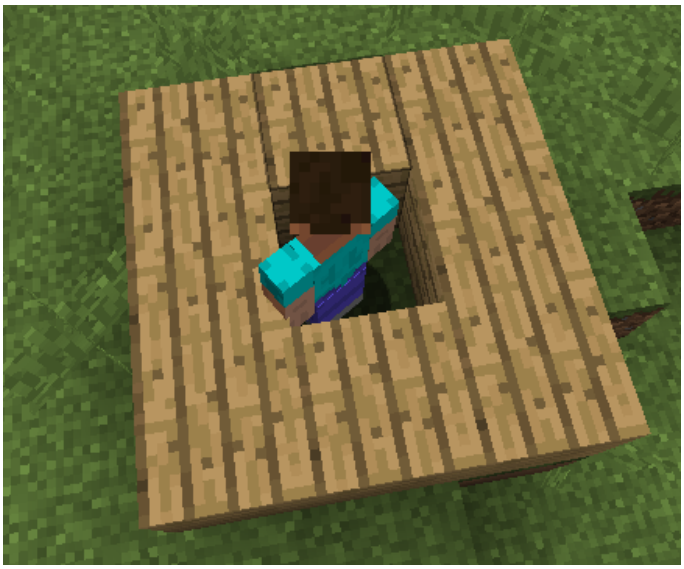

Рисунок 2.8 – Приклад виконаного завдання 8 в Minecraft

Завдання №9. Побудова вежі з каменя

Створити програму, яка створює поруч з гравцем вежу з камня висотою 10 блоків. Приклад виконаного завдання зображений на рис. 2.9.а та 2.9.б.

Виконати завдання таким чином:

1. Створити зміну pos з координатами гравця.

2. Визначити цикл for, що виконує 10 ітерацій

3. Додати в цикл команду створення нового блока, в якій до координати Y слід додати змінну циклу

```
import mcpi.minecraft as minecraft
mc = minecraft.Minecraft.create()

pos= mc.player.getPos()

for i in range(10):
    mc.setBlock(pos.x,pos.y+i,pos.z,1,0)
```

Рисунок 2.9.а – Приклад коду до завдання 7

Рисунок 2.9.б – Приклад виконаного завдання 9 в Minecraft

Завдання №10. Побудова вежі з випадкових блоків

Створити програму, яка створює поруч з гравцем вежу висотою 30 випадкових блоків. Приклад виконаного завдання зображений на рис. 2.10.

Рекомендації: Для виконання завдання доцільно скористатися циклом for.

Додаткове завдання: Додати в цикл for команду sleep(1). Відстежити зміну значення змінних виконання програми в грі.

Рисунок 2.10 – Приклад виконаного завдання 10 в Minecraft

Завдання №11 Побудова куба

Користувачеві пропонується ввести розмір куба та ідентифікатор блока в программі Python. Ці значення використаються в команді setBlocks().

Висвітлення використання команди setBlocks() та команди map(). Приклад вікна із завданням зображений на рис.2.11.а та 2.11.б

Рисунок 2.11 – Приклад виконаного завдання 11 в Minecraft

2.3. Розробка ігрових ситуацій для школярів в середовищі Minecraft

Завдання №1 «Неймовірні пригоди»

Стів трохи занудьгував. Він не може знайти жодного цікавого містечка по всьому світу Minecraft. Допоможіть йому. Напишіть код, в якому потрібно телепортувати Стіва в цікаві місця кожну хвилину. Координати цих точок спочатку потрібно вам самим відшукати (рис. 2.12), тому зупинимось на п’яти цікавих місцевостях. Програму можна зробити більш хаотично якщо використовувати бібліотеку random замість заданих координат.

Рисунок 2.12 – Приклад цікавої локації до завдання «Неймовірні пригоди»

Завдання №2 «Супер сила»

Всі мріять стати супергероєм, Алекс також не виняток. Але все що вона може робити, це створити випадково пожежу в своєму лісі. Давайте зробимо їй сюрприз. Створіть програму, яка підкинить її на 100 блоків вгору, по 10 блоків кожну секунду. Після цього програма поставить під нею блок скла, щоб вона змогла побачити всі простори під собою(рис 2.13). Нехай відчує себе справжнім супергероєм!

Рисунок 2.13 – Приклад фінальної стадії завдання «Супер сила»

Завдання №3 «Блоки за секунду»

Всі знають швидкість вагонетки, коня, свині але ніхто не знає швидкість Стіва в Minecraft! Виправте ситуацію, напишіть код, який буде виводити результат підрахунків - скільки блоків пройшов Стів по прямій за одну хвилину. Результат зображенний на рис 2.14.

Рисунок 2.14 – Приклад виконанання завдання «Блоки за секунду»

Завдання №4 «Безпека понад усе»

Інколи трапляються ситуації, коли дещо пішло не по плану. Наприклад, у процесі шахтарства, Стів провалився в лаву, або під час подорожі, Алекс забула, що не вміє плавати та почала тонути в воді. Треба цього уникнути! Напишіть код, який буде завжди перевіряти, де знаходиться гравець. Якщо він впав на один блок в лаву, то піднімить його на два блоки верх, та поставте під ним блок, бажано непалаючий. Якщо ж гравець занурився в воду на три блоки, то підніміть його на чотири блоки, і також розмістіть, як в ситуації з лавою, під ним блок, можна будь-який. Результат зображенний на рис 2.15.

Рисунок 2.15 – Приклад виконаного завдання «Безпека понад усе»

Завдання №6 «Копач»

Стів втомився шукати золоту руду для своєї нової кирки. Щоб зберегти час від шахтарства, напишіть код, який дізнається, чи є в землі під гравцем будь-яка руда(залізо, золото, алмаз). Програма отримує поточні координати гравця і по одному перебирає блоки під ногами, перевіряючи, чи не є вони рудою.

Рисунок 2.16 – Приклад виконаного завдання «Копач»

Якщо знайдеться задовільний блок або блоки, повідомте Стіва де він знаходиться. Результат зображенний на рис 2.16.

Завдання №6 «Шахтар-експерс»

Всі гравці полюбляють мандрувати по світу Minecraft, незалежно де, в болотах, в шахтах, в глубинах океану. Але є одна прикра річ, відстань, інколи доводиться проходити значний шлях. Спробуймо це подолати! Створіть код, який буде будувати автоматичну колію для вагонеток. Так як, всі світи дуже привабливі і Стів хоче всюди побувати, вам потрібно поміркувати як зробити колію придатну для підземних та підводних пригод. Приклад виконання зображенно на рис. 2.17.

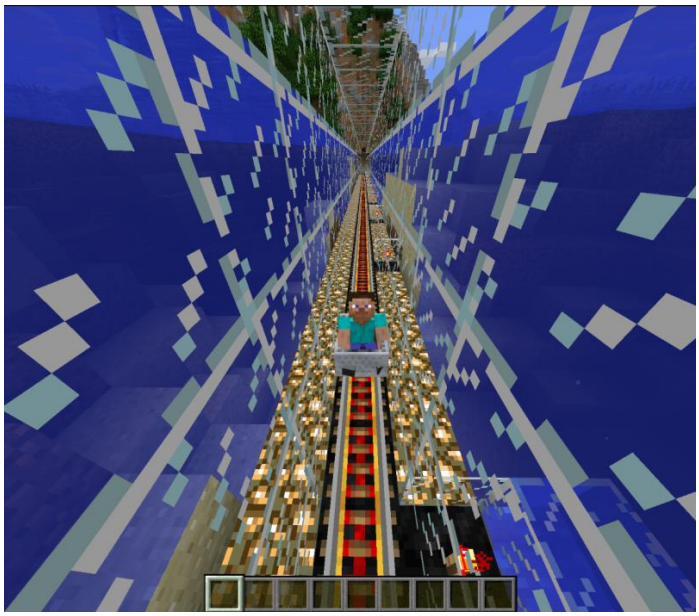

Рисунок 2.17 – Приклад виконаного завдання «Шахтар-експерс»

Завдання №7 «Неприступна фортеця»

Незабаром настане темрява! Стів та Алекс загубили свою домівку, а блоків для побудови нового будинку в них зовсім немає. Починають з’являтися зомбі, і без нашої допомоги вони не впораються. Створіть код, який побудує захисну оселю для Стіва та Алекс. Параметри оселі: будинок має бути розміром 5 х 5 х 5 блоків, матеріал стіни – обсидіан, мати вхід з дверима та принаймні 1 віконце для того, щоб дізнатись чи є небезпека зовні. Приклад виконання зображенно на рис. 2.18.

Рисунок 2.18 – Приклад виконаного завдання «Неприступна фортеця»

ВИСНОВКИ

Навчання будь-якого предмета в школі повинно бути організоване таким чином, щоб учням було цікаво на уроках, щоб вони самі прагнули отримувати нові знання і вчителю не доводилося б змушувати їх засвоювати навчальний матеріал. Предмет «Інформатика», з одного боку, знаходиться в більш вигідному становищі, ніж інші шкільні предмети, тому що використання на уроках комп’ютера саме по собі вже привабливе для учнів. Але, з іншого боку, багато учнів пов'язують комп'ютер виключно з можливістю пограти, а дітей потрібно навчити не тільки грати в ігри, але й вміти їх створювати.

В результаті аналізу літератури з теми дослідження виявлено великий спектр можливостей ігрових елементів, які можна застосувати в освітньому процесі. Продемонстровані перспективність і актуальність використання різноманітних платформ, ігор та рушіїв гри на уроках інформатики, виявлено особливості шкільних занять з використанням ігрових елементів, виділені переваги їх використання. У ході роботи було розроблено комплект тренувальних завдань та ігрових ситуацій для навчання школярів 7-9 класів основ об'єктно-орієнтованого програмування мовою Python у середовищі Minecraft. Запропоновано й детально висвітлено 11 тренувальних завдань та 6 ігрових ситуацій. Всі завдання є авторськими і апробованими у практичній роботі з індивідуальної підготовки школярів.

Використання сучасних технологій у навчальному процесі дозволяє підтримувати високий рівень мотивації учнів, запропонувати учням велику кількість готових, старанно відібраних, відповідним чином організованих знань, розвивати інтелектуальні, творчі здібності учнів і сприяти розвитку комунікативних аспектів, навичок самостійного створення програм.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Al-Imamy S., Alizadeh J, Nour M. On the Development of a Programming Teaching Tool: The Effect of Teaching by Templates on the Learning Process. URL: https://www.researchgate.net/publication/220590581_On_the_Development_of_a_Programming_Teaching_Tool_The_Effect_of_Teaching_by_Templates_on_the_Learning_Process

2. Campbell W., Bolker E.. Teaching programming by immersion, reading and writing. URL: http://fie2012.fie-conference.org/sites/fie2012.fie-conference.org/history/fie2002/papers/1637.pdf

3. Kolling M., Rosenberg J. Guidelines for Teaching Object Orientation with Java. URL: https://kar.kent.ac.uk/13607/1/guidelines_for_teaching_object_kolling.pdf

4. Linxiao Ma. Investigating and Improving Novice Programmers' Mental Models of Programming Concepts/ c.2. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.721.8479&rep=rep1&type=pdf

5. Lui A., Kwan R., Poon M., Cheung Y. Saving weak programming students: applying constructivism in a first programming course. URL: https://dl.acm.org/doi/10.1145/1024338.1024376

6. Miliszewska I. Befriending Computer Programming: A Proposed Approach to Teaching Introductory Programming URL: https://www.researchgate.net/publication/240948260_Befriending_Computer_Programming_A_Proposed_Approach_to_Teaching_Introductory_Programming

7. Minecraft. Education Edition. URL: https://education.minecraft.net/how-it-works/why-minecraft

8. Nuutila E., Törmä S., Kinnunen P., Malmi L. Learning Programming with the PBL Method — Experiences on PBL Cases and Tutoring. URL:

https://www.researchgate.net/publication/226134435_Learning_Programming_with_the_PBL_Method_-_Experiences_on_PBL_Cases_and_Tutoring

9. Sajaniemi J. and other Roles of variables in three programming paradigms. URL: https://www.researchgate.net/publication/228623601_Roles_of_variables_in_three_programming_paradigms

10. Sattar A., Lorenzen T. Mathematics and computer science faculty publications. URL: https://vc.bridgew.edu/math_compsci_fac/16/

11. Scott T. Leutenegger, Edgington J. A games first approach to teaching introductory programming. URL: https://www.researchgate.net/publication/221538278_A_games_first_approach_to_teaching_introductory_programming

12. UAE schools need to offer the 'language of the future' in regular lessons. URL: https://www.thenationalnews.com/uae/uae-schools-need-to-offer-the-language-of-the-future-in-regular-lessons-1.790391

13. Офіційний сайт міністерства освіти і науки України, Навчальні програми для учнів 5-9 класів. URL: https://mon.gov.ua/ua/osvita/zagalna-serednya-osvita/navchalni-programi/navchalni-programi-5-9-klas

14. Чайка В. М. Основи дидактики, розділ Класифікації за системним підходом. URL: https://textbook.com.ua/pedagogika/1473451780/s-14

15. Glass K., Hickman G. Official GitHub py3minepi project URL: https://github.com/py3minepi/py3minepi

16. O'Hanlon M. Minecraft - Stuff Library URL: https://minecraft-stuff.readthedocs.io/en/latest/

Додаток А

ОПЕРАТОРИ БІБЛІОТЕК MINECRAFT ДЛЯ НАВЧАННЯ ШКОЛЯРІВ ПРОГРАМУВАННЮ МОВОЮ PYTHON

1) MinecraftDrawing API - можна використовувати для створення ліній, кіл, сфер та граней.

Class minecraftstuff.MinecraftDrawing(mc)

MinecraftDrawing - клас корисних функцій малювання

Параметри:	mc (mcpi.minecraft.Minecraft) – Об'єкт Minecraft, який пов'язаний зі світом.

drawCircle(x0, y0, z, radius, blockType, blockData=0) - малює коло в площині Y (тобто вертикально)

Параметри:	x0 (int) – X положення центру кола. y0 (int) – У положення центру кола. z0 (int) – Z положення центру кола. radius (int) – Радіус кола. blockType (int) – Ідентифікатор блоку. blockData (int) – Значення даних блоку за замовчуванням дорівнює 0.

drawFace(vertices, filled, blockType, blockData=0) - малює грань, коли передається колекція вершин, що складають багатогранник

Параметри:	vertices (list) – Список точок, переданих як будь-який minecraftstuff.Points об'єкт або як список mcpi.minecraft.Vec3 об'єктів. filled (boolean) – Якщо True наповнює грань блоками.

blockType (int) – Ідентифікатор блоку, далі “ID”.
blockData (int) – Значення даних блоку, за замовчуванням дорівнює 0, далі “Значення даних”.

drawHollowSphere(x1, y1, z1, radius, blockType, blockData=0) - малює порожню сферу навколо точки до радіуса

Параметри:
x1 (int) – X положення центру сфери.
y1 (int) – У положення центру сфери.
z1 (int) – Z положення центру сфери.
radius (int) – Радіус сфери.
blockType (int) – ID.
blockData (int) – Значення даних.

drawHorizontalCircle(x0, y, z0, radius, blockType, blockData=0) - малює коло в площині X (тобто горизонтально)

Параметри:
x0 (int) – X положення центру кола.
y0 (int) – Y положення центру кола.
z0 (int) – Z положення центру кола.
radius (int) – Радіус кола.
blockType (int) – ID.
blockData (int) – Значення даних.

drawLine(x1, y1, z1, x2, y2, z2, blockType, blockData=0) - проводить лінію між 2 точками

Параметри:
x1 (int) – X положення першої точки.
y1 (int) – Y положення першої точки.
z1 (int) – Z положення першої точки.
x2 (int) – X положення лругої точки.
y2 (int) – Y положення лругої точки.

z2 (int) – Z положення лругої точки.
blockType (int) – ID.
blockData (int) – Значення даних.

drawPoint3d(x, y, z, blockType, blockData=0) - малює одну точку в Minecraft, тобто 1 блок

Параметри:
x (int) – Позиція X.
y (int) – Позиція Y.
z (int) – Позиція Z.
blockType (int) – ID.
blockData (int) – Значення даних.

drawSphere(x1, y1, z1, radius, blockType, blockData=0) - малює сферу навколо точки до радіуса

Параметри:
x1 (int) – X положення центру сфери.
y1 (int) – Y положення центру сфери.
z1 (int) – Z положення центру сфери.
radius (int) – Радіус сфери.
blockType (int) – ID.
blockData (int) – Значення даних.

drawVertices(vertices, blockType, blockData=0) - малює всі точки в колекції вершин з блоком

Параметри:
vertices (list) – Список mcpi.minecraft.Vec3 об’єктів.
blockType (int) – ID.
blockData (int) – Значення даних.

getLine(x1, y1, z1, x2, y2, z2) - Повертає всі точки, які складають лінію між 2 точками як список. 3D реалізація алгоритму лінії Брезенхема

Параметри:
x1 (int) – X положення першої точки.

y1 (int) – Y положення першої точки.

z1 (int) – Z положення першої точки.

x2 (int) – X положення лругої точки.

y2 (int) – Y положення лругої точки.

z2 (int) – Z положення лругої точки.

2)Points - колекція Minecraft позицій або Vec3. Використовується для малювання граней MinecraftDrawing.drawFace().

Class minecraftstuff.Points

add(x,y,z) - додати одну позицію до списку точок.

Параметри:
x (int) – Позиція X.
y (int) – Позиція Y.
z (int) – Позиція Z.

getVec3s() - повертає список позицій Vec3

3)MinecraftShape API - дозволяє створювати та обробляти фігури (колекції блоків) у Minecraft.Фігури можуть бути малими або великими, і API допомагає, ефективно рухаючи їх і змінюючи лише ті блоки, які насправді змінилися.

Classminecraftstuff.MinecraftShape(mc, position, shapeBlocks=None, visible=True)

MinecraftShape - реалізація "фігури" в Minecraft. Кожна фігура складається з одного або багатьох блоків з положенням відносно один одного. Фігури можуть бути перетворені за допомогою руху та обертання. Коли фігура змінюється і перемальовується в Minecraft, оновлюються лише ті блоки, які змінились.

Параметри:
mc (mcpi.minecraft.Minecraft) – Об'єкт Minecraft, який пов'язаний зі світом.

position (mcpi.minecraft.Vec3) – Позиція, де слід створювати фігуру.

shapeBlocks (list) – Список ShapeBlocks, з яких складається фігура. За замовчуванням значення None.

visible (bool) – Де фігура повинна бути видно. За замовчуванням значення True.

clear() - очищає фігуру в Minecraft.

draw() - малює фігуру в Minecraft, беручи до уваги, де вона була намальована востаннє, лише оновлюючи змінені блоки.

getShapeBlock(x, y, z) - повертає ShapeBlock для "фактичного положення".

Параметри: x (int) – Позиція X.
y (int) – Позиція Y.
z (int) – Позиція Z.

move(x, y, z) - переміщує положення фігури до x, y, z.

Параметри: x (int) – Позиція X.
y (int) – Позиція Y.
z (int) – Позиція Z.

moveBy(x, y, z) - переміщує положення фігури на x, y, z.

Параметри: x (int) – Кількість блоків для переміщення в X.
y (int) – Кількість блоків для переміщення Y.
z (int) – Кількість блоків для переміщення Z.

redraw() - перемальовує фігуру в Minecraft, очищаючи всі блоки та перемальовуючи їх.

reset() - скидає фігуру у початкове положення.

rotate(yaw, pitch, roll) - встановлює обертання фігури за допомогою повороту навколо вертикальної осі, повороту та обертання

Параметри: yaw (float) – Обертання щодо вертикальної осі в градусах.

pitch (float) – Кут обертання в градусах.

roll (float) – Обертання осі в градусах.

rotateBy(yaw, pitch, roll) – збільшує the rotation of a shape by yaw, pitch and roll

setBlock(x, y, z, blockType, blockData=0, tag='') - встановлює один блок у фігурі і перемальовує, щоб він намалював одну точку в Minecraft, тобто 1 блок

Параметри:

x (int) – Позиція X.

y (int) – Позиція Y.

z (int) – Позиція Z.

blockType (int) – ID.

blockData (int) – Значення даних.

tag (string) – Тег для блоку, це корисно для групування блоків та їх відстеження, оскільки положення блоків може змінюватися, за замовчуванням "".

setBlocks(x1, y1, z1, x2, y2, z2, blockType, blockData=0, tag='') - створює кубоїд із блоків у формі і перемальовує його

Параметри:

x1 (int) – X положення першої точки.

y1 (int) – Y положення першої точки.

z1 (int) – Z положення першої точки.

x2 (int) – X положення лругої точки.

y2 (int) – Y положення лругої точки.

z2 (int) – Z положення лругої точки.

blockType (int) – ID.

blockData (int) – Значення даних.

tag (string) – Тег для блоку, це корисно для групування блоків та їх відстеження, оскільки положення блоків може змінюватися, за замовчуванням "".

4)ShapeBlock

classminecraftstuff.ShapeBlock(x, y, z, blockType, blockData=0, tag='')

ShapeBlock - клас, щоб утримувати один блок у фігурі

Параметри:
x (int) – Позиція X.
y (int) – Позиція Y.
z (int) – Позиція Z.
blockType (int) – ID.
blockData (int) – Значення даних.
tag (string) – Тег для блоку, це корисно для групування блоків та їх відстеження, оскільки положення блоків може змінюватися, за замовчуванням "".

resetRelativePos() - скидає відносне положення блоку назад до початкового положення

5) MinecraftTurtle

Minecraft Turtle - це відтворення класичної графічної черепахи для Minecraft. Ключова відмінність полягає в тому, що ви можете малювати в 3 вимірах, а не лише в 2.

Classminecraftstuff.MinecraftTurtle(mc, position=<MagicMock name='mock()' id='139984989718456'>)

MinecraftTurle - графічна черепаха, яку можна використовувати для створення "малюнків" у Minecraft, контролюючи її положення, кути та напрямок

Параметри:
mc (mcpi.minecraft.Minecraft) – Об'єкт Minecraft, який пов'язаний зі світом.
position (mcpi.minecraft.Vec3) – Позиція, де слід створювати фігуру.

*backward(*distance*)* - рухати черепаху назад

Параметри: distance (int) – кількість блоків для переміщення.

*down(*angle*)* - повернути черепаху вниз

Параметри: angle (float) – кут в градусах для повороту.

*forward(*distance*)* – рухати черепаху вперед

Параметри: distance (int) – кількість блоків для переміщення.

home() - скинути положення черепахи.

isdown() - повертає True, якщо ручка опущена.

*left(*angle*)* - поверніть черепаху вліво

Параметри: angle (float) – кут в градусах для повороту. angl

*penblock(*blockId, blockData=0*)* - встановлює блок, який черепаха використовує як перо

Параметри: blockType (int) – ID.
blockData (int) – Значення даних.

pendown() - відкладіть черепаху, покажіть, що вона намалює.

penup() - підніміть ручку черепах, покажіть, що вона не намалює.

*right(*angle*)* - повернути черепаху вправо

Параметри: angle (float) – кут в градусах для повороту.

*setposition(*x, y, z*)* - встановити положення черепахи

Параметри: x (int) – Позиція X.
y (int) – Позиція Y.
z (int) – Позиція Z.

*speed(*turtlespeed*)* - встановіть швидкість черепахи

Параметри: turtlespeed(int) – 1 - 10, 1 - найповільніший, 10 – найшвидший. Якщо встановлено значення 0, черепаха миттєво малює.

*up(*angle*)* - повернути черепаху вгору

Параметри: angle (float) – кут в градусах для повороту.

ДОДАТОК Б

КОД ДО ПРАКТИЧНИХ ЗАВДАНЬ

1. Завдання 1.

```
mc = minecraft.Minecraft.create()

x = int(input("Введіть значення X = "))

y = int(input("Введіть значення Y = "))
while y<0:
    print("Невірне значення Y")
    y = int(input("Введіть значення Y = "))

z = int(input("Введіть значення Z = "))

mc.player.setPos(x,y,z)
mc.postToChat("Teleported to "+str(x)+" "+str(y)+" "+str(z))
print("Гравця телепортовано на координати ",x,y,z)
```

2. Завдання 2.

```
import mcpi.minecraft as minecraft
mc = minecraft.Minecraft.create()

x = 100
y = 200
z = "30"

mc.pleyr.setPos(str(x),z,c)
print("Все працює!")
```

3. Завдання 3.

```
import mcpi.minecraft as minecraft
from random import *
from time import *
mc = minecraft.Minecraft.create()

x = randint(-100000,100000)
y = randint(70,200)
z = randint(-100000,100000)
mc.player.setPos(x,y,z)
```

```
sleep(10)
x = randint(-100000,100000)
y = randint(70,200)
z = randint(-100000,100000)
mc.player.setPos(x,y,z)
sleep(10)
x = randint(-100000,100000)
y = randint(70,200)
z = randint(-100000,100000)
mc.player.setPos(x,y,z)
sleep(10)
```

4. Завдання 4.

```
import mcpi.minecraft as minecraft
mc = minecraft.Minecraft.create()

pos= mc.player.getTilePos()
x = pos.x
y = pos.y
z = pos.z

mc.postToChat(x)
mc.postToChat(y)
mc.postToChat(z)
```

5. Завдання 5.

```
import mcpi.minecraft as minecraft
import time
mc = minecraft.Minecraft.create()

pos= mc.player.getTilePos()
x = pos.x
y = pos.y
z = pos.z

time.sleep(10)

pos= mc.player.getTilePos()
x1 = pos.x
y1 = pos.y
z1 = pos.z

mc.postToChat("Player has moved x = " + str(x1-x)+", y = "
```

```
        + str(y1-y) + ", z = "+ str(z1-z))
```

6. Завдання 6.

```
import mcpi.minecraft as minecraft
mc = minecraft.Minecraft.create()

pos= mc.player.getPos()
x = pos.x
y = pos.y
z = pos.z



a = input()
if a=='x':
   mc.player.setPos(x+int(input()),y,z)
elif a=='y':
   mc.player.setPos(x,y+int(input()),z)
else:
   mc.player.setPos(x,y,z+int(input()))
```

7. Завдання 7.

```
import mcpi.minecraft as minecraft
mc = minecraft.Minecraft.create()

pos= mc.player.getPos()
x = pos.x
y = pos.y
z = pos.z



ID = int(input())
mc.setBlock(x+1,y,z,ID,0)
```

8. Завдання 8.

```
import mcpi.minecraft as minecraft
mc = minecraft.Minecraft.create()

pos= mc.player.getPos()
x = pos.x
y = pos.y
z = pos.z
```

```
mc.setBlock(x+1,y,z,5,0)
mc.setBlock(x-1,y,z,5,0)
mc.setBlock(x,y,z-1,5,0)
mc.setBlock(x,y,z+1,5,0)
mc.setBlock(x+1,y,z+1,5,0)
mc.setBlock(x+1,y,z-1,5,0)
mc.setBlock(x-1,y,z+1,5,0)
mc.setBlock(x-1,y,z-1,5,0)
```

9. Завдання 9.

```
import mcpi.minecraft as minecraft
mc = minecraft.Minecraft.create()

pos= mc.player.getPos()

for i in range(10):
   mc.setBlock(pos.x,pos.y+i,pos.z,1,0)
```

10. Завдання 10.

```
import mcpi.minecraft as minecraft
from random import *
mc = minecraft.Minecraft.create()

pos= mc.player.getPos()

for i in range(30):
   mc.setBlock(pos.x,pos.y+i,pos.z,randint(0,100),0)
```

11. Завдання 11.

```
import mcpi.minecraft as minecraft
mc = minecraft.Minecraft.create()

pos= mc.player.getPos()
x = pos.x + 5
y = pos.y
z = pos.z

size,ID = map(int,input().split())

mc.setBlocks(x,y,z,x+size,y+size,z+size,ID,0)
```

АНОТАЦІЯ

НАВЧАННЯ ШКОЛЯРІВ ОСНОВ ПРОГРАМУВАННЯ ЗА ДОПОМОГОЮ СЕРЕДОВИЩА MINECRAFT

На даний час вважливо розвивати інтерес школярів до програмування. Це можна робити за допомогою впровадження в освітній процес різних ігрових елементів. З цією метою у нагоді стане середовищеMineCraft, яке на даний час має потужну освітню компоненту і надає різні можливості для впровадження у навчальний процес. Завдяки своїй гнучкості гра легко підлаштовується під різні дисципліни. Гармонійне поєднання ігрових та навчальних технологій дає змогу учням сприймати складний навчальний матеріал у ігровій формі, що більш цікаво та комфортно для них. Але наразі бракує методичних розробок з навчання програмуванню, які би враховували інтереси школярів та були доступними для сприйняття. Отже, на наш погляд, розробка практичних завдань з навчання мови Python у середовищі MineCraft є цілком доречною і актуальною на даний час.

Мета дослідження: визначити способи навчання учнів об’єктно-орієнтованому програмуванню, а також розробити практичні завдання з основ об'єктно-орієнтованого програмування для навчання учнів 7-9 класів мови програмування Python у середовищі MineCraft

Відповідно до мети, поставлено такі **завдання**:

4. Розкрити роль ООП у системі інформатичної підготовки учнів 7-9 класів і розглянути методичні аспекти викладання інформатики.

5. Схарактеризувати основні підходи до навчання програмування й описати ООП як сучасну технологію в розробці програмного забезпечення.

6. Розробити практичні завдання з основ об'єктно-орієнтованого мовою Python у середовищі MineCraft.

Методи дослідження: для вирішення поставлених завдань та досягнення мети було використано такі методи дослідження: *теоретичні*: вивчення

навчальних програм, підручників, методичних посібників, а також аналіз статей, наукових досліджень, публікацій та метод *проектування*.

В результаті аналізу літератури з теми дослідження виявлено великий спектр можливостей ігрових елементів, які можна застосувати в освітньому процесі. Продемонстровані перспективність і актуальність використання різноманітних платформ, ігор та рушіїв гри на уроках інформатики, виявлено особливості шкільних занять з використанням ігрових елементів, виділені переваги їх використання. У ході роботи було розроблено комплект тренувальних завдань та ігрових ситуацій для навчання школярів 7-9 класів основ об'єктно-орієнтованого програмування мовою Python у середовищі Minecraft. Запропоновано й детально висвітлено 11 тренувальних завдань та 6 ігрових ситуацій. Всі завдання є авторськими і апробованими у практичній роботі з індивідуальної підготовки школярів.