Висновки. Алгоритми є невід'ємною частиною програмування і багатьох прикладних сфер діяльності, їх ефективна реалізація відіграє ключову роль у створенні надійного продукту. Практичні приклади та результати тестування показують, що один і той самий алгоритм може бути реалізований різними способами з різною ефективністю, що обумовлює необхідність вибору оптимального рішення для конкретних умов. Правильне використання алгоритмізації є основним шляхом підвищення ефективності програмного коду та його оптимізації. Часову ефективність реалізації алгоритму на програмному коді можна визначити за допомогою емпіричного тестування та вбудованих бібліотек.

Abstract. The article discusses the basics of algorithms and their importance and application in the field of programming. The article describes the key properties of algorithms that make a program code reliable and efficient. Different methods of describing algorithms and their widespread use in modern technologies are also investigated. Examples of algorithm implementation in the C# programming language are provided, which demonstrate the practical aspects of the effectiveness of various algorithmic approaches through empirical testing. The article emphasises the importance of choosing the right algorithmic structures to optimise software performance and reliability.

Keywords: algorithms, programming, data structures, tasks, resources.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Алгоритми та структури даних – від «десь чув» до «ефективно застосовую». *DOU.* 02.11.2022. URL: https://dou.ua/forums/topic/40645/

2. Що таке алгоритми: кроки, приклади, конструкції, мислення. *Dan-it education.* 21.10.2024. URL: https://dan-it.com.ua/uk/blog/shho-take-algorytmy-kroky-pryklady-konstrukcziyi-myslennya/

3. Романов В. Поняття алгоритму. Властивості алгоритмів. Форми подання алгоритму. Виконавець алгоритму. *Інформатика – шкільний курс.* URL: https://romanov.in.ua/11-3/

4. Алгоритм та його властивості. *КЗ «Піщанський ліцей № 1». Step by step.* 2021–2022. URL: https://step.org.ua/konspekt/algoritm/tema1

5. Чому важливо оцінювати складність алгоритму. *foxminded.ua. Курси програмування.* 23.02.2024. URL: https://foxminded.ua/skladnist-alhorytmu/

6. Емпіричні методи програмної інженерії: електронний конспект. Київ: НТУ «Київський Політехнічний Інститут», 2015. 119 с. URL: http://tc.kpi.ua/content/kurs/EMPI/EMPI_konspect.pdf

7. Sedgewick R., Wayne K. Algorithms. Addison-Wesley Professional, 2020. 956 p.

8. Korman T. H. Algorithms. Unlocked. The MIT Press, Massachusetts Institute of Technology, 2013. 207 p.

9. Креневич А. П. Алгоритми і структури даних: підручник. Київ: ВПЦ «Київський Університет», 2021. 200 с.

УДК 004.42: 006.73: 793.4

СТВОРЕННЯ 2D-ГРИ У СТИЛІ ПІКСЕЛЬ-АРТ

І. С. Діброва, О. В. Зелінська

Анотація. Дослідження фокусується на процесі створення піксельної арт-гри з використанням ігрового рушія Unity. У статті проаналізовано основні етапи розробки, включго з підготовкою графічних ресурсів, інтеграцією анімації, налаштуванням ігрової механіки та створенням користувацького інтерфейсу. Підкреслюється важливість детального вивчення кожного з цих етапів через їх вплив на загальний ігровий досвід. Також приділяється увага різним інструментам, доступним в Unity, та їх ролі у спрощенні процесу розробки, що дає змогу зосередитися на творчих аспектах.

Ключові слова: Aseprite, Unity, анімація, інтеграція, піксель-арт, скрипт, розробка ігор.

Комп'ютерна гра – це програмне забезпечення, призначене для забезпечення ігрового процесу. Ігри можуть спілкуватися з гравцями та співпрацювати з ними. В ігровому процесі використовуються різноманітні пристрої введення, зокрема комп'ютерні миші, клавіатури, камери та джойстики. Створення відеогри нічим не відрізняється від створення будь-якого іншого програмного продукту. До команди входять програмісти, менеджери, дизайнери, звукорежисери, дизайнери інтерфейсів.

Із розвитком ігрової індустрії з'являлись і нові жанри: шутери, пригоди, головоломки, навчальні, симулятори, платформери, RPG, стелс та багато інших.

Піксельне мистецтво – це форма цифрового мистецтва, створена на рівні пікселів. Здебільшого асоціюється з графікою відеоігор 1980-х та 1990-х років. У той час художникам доводи-

лося враховувати обмеження пам’яті та низьку роздільну здатність. Сьогодні піксельне мистецтво популярне в іграх і як загальний художній стиль, незважаючи на можливість створення реалістичної 3D-графіки [1].

Для створення анімації та графіки у стилі піксель-арту існує безліч застосунків, проте найпопулярніший із них – **Aseprite** (рис. 1).

Рис. 1. Додаток Aseprite

Перед створенням анімованої растрової моделі необхідно визначити розмір полотна. Розмір полотна визначає кількість пікселів, що використовуються для малювання моделі, і впливає на деталізацію та якість кінцевого результату. Найпоширеніші розміри полотна для піксельного мистецтва – 16×16, 32×32, 64×64 і 128×128 пікселів. Менші розміри, як-от 16×16, зазвичай використовуються для спрайтів з невеликою деталізацією, як-от персонажі та іконки зі старих 8-бітних ігор. Більші розміри (64×64 і вище) підходять для відображення більш деталізованих об’єктів і складних сцен у сучасних піксельних іграх з високим рівнем деталізації [2].

Aseprite дає змогу створювати або завантажувати власні палітри. Оскільки піксельне мистецтво має обмежену кількість кольорів, вибір палітри є важливим кроком. Палітри можуть бути дуже обмеженими, типовими для ретро-ігор (наприклад, до 16 кольорів), або більш гнучкими для сучасного піксельного мистецтва. Aseprite має вбудовані палітри, а також можливість імпортувати готові палітри, створені спільнотою.

Програма надає корисні інструменти для малювання пікселів (табл. 1).

Таблиця 1

Інструменти в програмі Aseprite

№	Інструмент	Опис
1	Олівець	малювання окремих пікселів або ліній
2	Пензель	малювання товстих ліній або заповнення великих областей кольором
3	Заливка	заповнює вибрану область або весь шар одним кольором
4	Гумка	видаляє пікселі або робить їх прозорими
5	Чарівна паличка	автоматично виділяє суміжні пікселі одного кольору
6	Шар	створює різні елементи зображення на окремих шарах
7	Кадри	створює окремі кадри для анімації, що дає змогу переглядати і редагувати послідовність рухів
8	Піпетка	вибирає колір з будь-якої точки полотна для подальшого використання
9	Часова шкала	редагування те регулювання кожного кадру анімації
10	Градієнт	створює плавний перехід між двома або більше кольорами
11	Обрізання	видаляє зайві частини зображення
12	Збільшення	змінює масштаб зображення
13	Поворот	повертає зображення або виділену область під певним кутом
14	Клонування	дублює вибрану область

Окремі інструменти дають змогу створювати елементи анімації, наприклад, окремо малювати тіло і рухомі частини (як-от руки або зброю), що значно полегшує роботу над деталями.

До того ж Aseprite має кадри, які допомагають створювати послідовності зображень для плавної анімації. Функція зациклення дає змогу спостерігати анімацію в реальному часі, що особливо корисно для періодичних анімацій, як-от рухи персонажів під час ходьби або бігу.

Програма може експортувати анімацію у форматі GIF, PNG або окремими кадрами, що допомагає легко інтегрувати анімаційну модель в ігри та інші проєкти.

Unity – багатоплатформовий інструмент для розроблення відеоігор та застосунків, і рушій, на якому вони працюють. Інтеграція анімації в ігровий рушій Unity – складний, багатоетапний процес, що вимагає детального налаштування графічних ресурсів та параметрів анімації для забезпечення коректної взаємодії з ігровим середовищем. Першим кроком є підготовка анімації у відповідному форматі: для 2D-ігор найчастіше використовуються формати PNG для окремих спрайтів або спрайт-листів, а також GIF для готових анімацій. Unity підтримує як окремі кадри анімації, так і спрайт-листи, тому можна створювати анімаційні послідовності. Після імпорту спрайтів їх можна інтегрувати в анімаційну систему Unity, яка використовує компоненти **Animator** та **Animation** для керування послідовністю кадрів [3].

Система анімації Unity базується на контролері Animator, який визначає стани та переходи між ними. Для кожної анімації створюється окремий анімаційний кліп, що містить послідовність кадрів. Контролер анімації керує її поведінкою в реальному часі та може встановлювати умови перемикання між різними станами, наприклад, у разі зміни швидкості руху персонажа або інших параметрів.

Важливим етапом інтеграції є налаштування параметрів анімації, як-от швидкість відтворення, плавність переходів між станами та синхронізація з іншими ігровими об’єктами. Значущою особливістю є можливість зациклювати анімацію для повторюваних дій, як-от ходьба, біг, атака тощо. Unity дає змогу регулювати тривалість кожного кадру анімації для досягнення необхідної плавності руху.

Важливим кроком є організація ігрових елементів за допомогою системи тегів та шарів. Теги допомагають швидко ідентифікувати об’єкти та взаємодіяти з ними за допомогою скриптів. Натомість шари допомагають структурувати об’єкти на різних рівнях і контролювати їх взаємодію [4]. Це важливо для налаштування колайдерних систем і фізичних взаємодій, а також для зручної роботи з анімацією. Наприклад, шари можна використовувати для розділення персонажів, снарядів і декоративних елементів, що полегшує їх незалежне редагування та взаємодію (рис. 2).

Рис. 2. Об’єкти та елементи гри

Щоб об’єкти коректно взаємодіяли, важливо додавати колайдери. Колайдери – це спеціальні компоненти, які визначають зону зіткнення між об’єктами. Колайдери мають форму коробок, сфер або капсул і використовуються для того, щоб забезпечити правильну реакцію об’єктів під час зіткнення. Фізичні властивості, як-от гравітація і тертя, також можуть бути використані для налаштування гри для досягнення бажаного рівня реалізму.

Налаштування руху персонажа вимагає реалізації різних типів рухів – ходьби, бігу, стрибків і т. ін. Unity використовує фізичні компоненти, як-от **Rigidbody,** що відповідає за симуляцію фізики, і скрипти, які контролюють введення даних гравцем (рис. 3).

```
// x-axis movement
if (Input.GetKey(KeyCode.A))
{
    transform.position += transform.right * -speed * Time.deltaTime;
}
if (Input.GetKey(KeyCode.D))
{
    transform.position -= transform.right * -speed * Time.deltaTime;
}
```

Рис. 3. Скрипт для руху персонажа за натисканням відповідних клавіш

Аналогічно можна налаштувати інші кнопки для руху персонажа, також для руху можна застосовувати відповідну анімацію, стрибок, рух, присідання тощо (рис. 4).

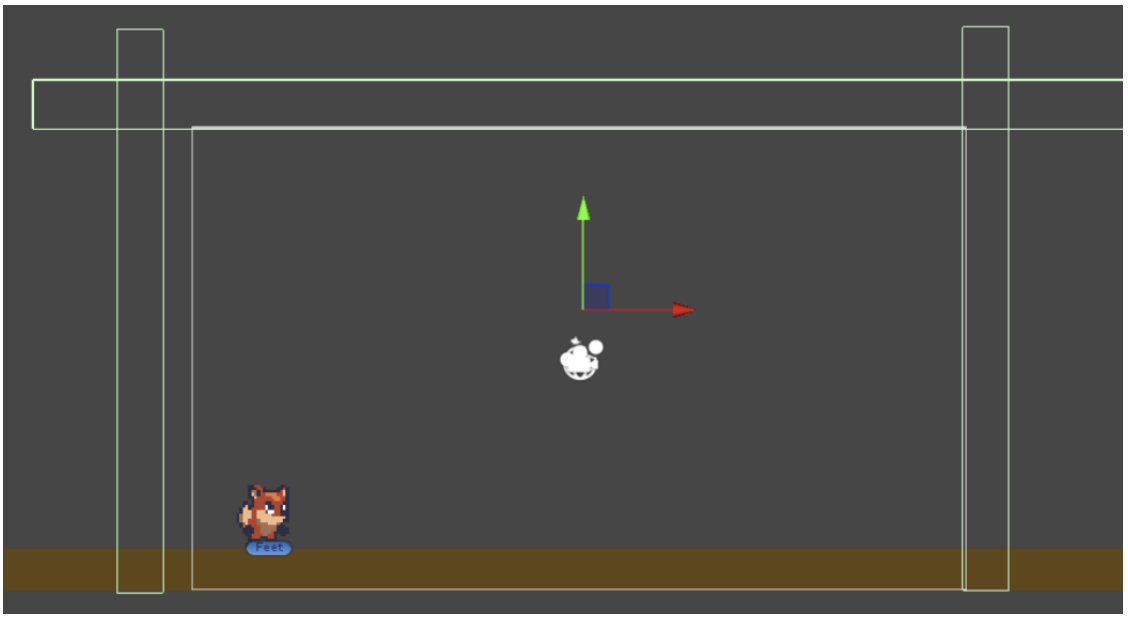

Рис. 4. Рух персонажа по x, y

Механіка руху часто використовує анімацію для забезпечення плавних переходів між різними станами руху персонажа. Спеціальні алгоритми та методи можуть використовуватися для досягнення природності та реалістичності у взаємодії персонажа з навколишнім середовищем, враховуючи складні фізичні взаємодії – тертя, гравітацію та зіткнення з об'єктами [5].

Генеруючи перешкоди та об'єкти в ігровому просторі, можна створювати різні сценарії взаємодії гравця з навколишнім середовищем. У Unity цей процес полягає у створенні спеціальних скриптів. Ці скрипти відповідають за появу об'єктів у певний час і в певних місцях на ігровому полі.

Бали використовуються для відображення досягнень та прогресу гравця. Цей компонент візуально інформує гравців про накопичені бали та допомагає мотивувати і залучати їх до гри. Реалізація **Score UI** включає кастомізацію елементів інтерфейсу, як-от текстові поля для відображення балів та механізми для оновлення інформації в режимі реального часу.

Так можна отримати повністю функціонуючу гру з рухом персонажа, проте для більшого функціоналу можна додати скрипт для системи для управління здоров'ям персонажа.

AudioManager – управління звуковими ефектами та музикою. Дуже важливим є **GameManager** – загальне управління ігровим процесом (збереження прогресу, перезапуск гри). **UIManager** може слугувати за управління інтерфейсом користувача. Керування всіма анімаціями може зберігати **AnimationController,** а для більшого реалізму можна створити штучний інтелект для ворогів та механіку збирання предметів [6].

Тому створення піксельних ігор на платформі Unity є чудовою можливістю для розробників, які хочуть поєднати класичну естетику з сучасними технологіями. Піксельне мистецтво, як стиль, вирізняється своєю простотою та емоційністю, що дає змогу створювати унікальні візуальні світи, які сприймаються гравцем з ностальгією. Unity завдяки своїй гнучкості та потужному інструментарію робить розробку піксельних ігор доступним.

Abstract. The research focuses on the process of creating a pixel art game using the Unity game engine. The article analyzes the main stages of development, including the preparation of graphic resources, animation integration, game mechanics configuration, and user interface creation. It emphasizes the importance of thoroughly studying each of these stages due to their impact on the overall gaming experience. Attention is also given to the various tools available in Unity and their role in simplifying the development process, allowing for a greater focus on creative aspects.

Keywords: Aseprite, Unity, animation, integration, pixel art, script, game development.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Pina M. Essential Techniques for Creating Pixel Art Games. *Gamasutra.* 2021. URL: https://www.gamasutra.com/blogs/MauricioPina/20210615/392307/Essential_Techniques_for_Creating_Pixel_Art_Games.php
2. Smith A. The Art of Pixel Animation: A Comprehensive Guide to Aseprite. *Pixel Art Journal.* 2022. Vol. 1, № 2. P. 50–67.
3. Д. Хоккінг – Unity в дії. Мультиплатформенна розробка на C#. 2023. С. 21–31. https://freeprog.org.ua
4. Бондаренко С. М. Розробка 2D-ігор з використанням Unity: посібник для початківців. Київ: Вид. дім «Освіта», 2021. 210 с.
5. Grey T. Designing Engaging Gameplay with Aseprite and Unity. *Game Studies Journal.* 2023. Vol. 15, № 1. P. 20–30.
6. Кузьменко В. С. Інтерфейс користувача в Unity: сучасні підходи. *Науковий вісник Національного університету «Львівська політехніка».* 2024. № 9. С. 90–98. URL: https://skid.lpnu.ua/wp-content/uploads/2024/05/ICS2024_Proceedings.pdf

УДК 004.415:005.591.6

ВИКЛИКИ ТА ОБМЕЖЕННЯ ПІД ЧАС ВПРОВАДЖЕННЯ AGILE В ІТ-ПРОЄКТАХ

Ю. О. Круць, В. Ю. Василенко

Анотація. У поданій статті розглядаються виклики та обмеження впровадження методології Agile в управлінні ІТ-проєктами. Виділяється важливість гнучких підходів для покращення адаптивності організацій, підвищення якості продуктів та ефективної взаємодії з клієнтами. Автори аналізують основні труднощі, зокрема супротив змінам на рівні організації, нечіткість вимог, складнощі з прогнозуванням ресурсів та вартості, а також проблеми у вимірюванні прогресу. Також приділяється увага методам подолання цих перешкод, як-от навчання персоналу, залучення замовників на ранніх етапах та впровадження інструментів візуалізації прогресу. Загальний висновок полягає в тому, що Agile може суттєво підвищити ефективність управління ІТ-проєктами, але потребує ретельної адаптації до реальних умов бізнесу та належної підготовки команд.

Ключові слова: Agile, інформаційні технології, впровадження, виклики, обмеження.

Вступ. У сучасному світі зростає необхідність у стандартизації підходів до управління проєктами. Управління ІТ-проєктами здійснюється як за допомогою традиційних методів, так і за допомогою підходів, що орієнтовані на роботу в умовах високої невизначеності. Сьогодні доступний широкий спектр методологій гнучкого управління проєктами, що включає різноманітні інструменти для досягнення цілей проєкту. Особливо варто відзначити сучасні гнучкі методології Agile, до яких належать ітеративно-інкрементальні методи управління – Scrum, Lean, Kanban, Crystal, eXtreme Programming (XP) та інші, що допомагають стандартизувати знання щодо використання гнучких підходів [1].

Agile методологія передбачає ітераційний підхід до розробки, акцентуючи на постійному вдосконаленні та тісній співпраці з клієнтом. Однак, впроваджуючи цю методологію, на практиці можна зіткнутися з певними труднощами та обмеженнями. Багато компаній стикаються з проблемами організаційної культури, адаптації до нового формату роботи та зовнішніми обмеженнями [2]. Ця стаття містить аналіз викликів та обмежень під час впровадження Agile в ІТ-проєктах, а також перспективи та можливості подолання цих проблем.

Мета поданої роботи – аналіз ключових викликів та обмежень, які можуть виникати під час впровадження методології Agile в ІТ-проєктах, а також розгляд можливих шляхів їх подолання.

Основний розділ. Agile-підхід до управління проєктами є однією з найпоширеніших та найефективніших методологій у різних галузях сьогодення, оскільки він дає змогу організаціям оперативніше адаптуватися до змін, підвищувати якість кінцевого продукту та краще взаємодіяти з клієнтами. Втім, попри численні переваги, процес його впровадження не позбавлений викликів. Саме тому важливо ретельно проаналізувати основні ***труднощі та обмеження, з якими можуть зіштовхуватися компанії під час переходу на Agile.***

Насамперед варто виділити ***супротив змінам на рівні організації,*** що є одним із найбільших викликів під час впровадження Agile. Багато працівників звикли до традиційних моделей