УДК 004.453.2

В. В. Войтко, к. т. н., доц; А. В. Денисюк; Г. Л. Луцишин

УНІВЕРСАЛЬНИЙ ГРАФІЧНИЙ КОМПОНЕНТНИЙ РЕДАКТОР ІГРОВИХ КАРТ

У статті запропоновано розробку графічного редактора ігрових карт, реалізованого за архітектурними принципами керування об'єктами з використанням компонентної технології для розширення своїх функціональних можливостей в системах відкритого типу. Редактор характеризується незалежністю від сюжетів і жанрів ігор та від конкретних ігрових каркасів, що забезпечує можливість його універсального використання не лише в галузі створення ігрових технологій, а і як засобу моделювання систем об'єктів у дизайнерських та інженерних розробках.

Ключові слова: *ігрові каркаси, автоматизація розробки ігор, ігрові сутності, компонентна архітектура, універсальні характеристики, структура графічних ігор.*

Вступ

Сьогодні, в епоху стрімкого розвитку комп'ютерних технологій, мультимедійні продукти, спрямовані на досягнення науково-дослідної, науково-просвітницької, навчальної, популяризаторської, розважальної і комерційної мети, набувають широкого використання. Розробці сучасних комп'ютерних ігор передує створення редактора ігрових карт, що вимагає додаткового часу у межах прогнозованого терміну розробки кінцевого програмного продукту [1]. Розроблені фірмами-виробниками ігрові карти реалізуються у вигляді окремих файлів, формати яких не є уніфікованими і не розголошуються. Редактор ігрових карт може входити до базової ігрової поставки для забезпечення можливості створення і використання користувачами власних карт у рамках окремого мультимедійного продукту з метою реалізації авторських ідей у графічному середовищі (така можливість забезпечується в іграх Termins, Lode Runner, Serious Sam та ін.).

Проте сучасні редактори ігрових карт характеризуються вузькоспеціалізованими характеристиками, обмеженими й жорстко визначеними функціональними можливостями та орієнтацією під конкретне програмне забезпечення для створення комп'ютерного додатку, що обмежує перспективи використання існуючих графічних редакторів навіть для написання нових версій ігрових мультимедійних продуктів. Використання наявних редакторів ігрових карт іншими розробниками взагалі вбачається неможливим через орієнтування редакторів на різні ігрові каркаси. Тому актуальною сьогодні є розробка графічних редакторів ігрових карт, спеціалізовані характеристики яких носили б універсальний характер, що зумовить перспективність їхнього використання в засобах створення комп'ютерних ігор.

Отже, метою роботи є автоматизація процесу створення ігрових карт засобами універсального графічного редактора. Об'єкт дослідження – методи побудови графічних редакторів. Предметом дослідження постають принципи створення ігрових карт і методи реалізації графічних ефектів у комп'ютерних іграх. Головними задачами роботи вважаємо розробку графічного редактора ігрових карт і забезпечення його універсальних характеристик.

Аналіз сучасних графічних редакторів ігрових карт

Розглянемо можливості та обмеження деяких існуючих редакторів. Гра Termins – двовимірна графічна стратегія реального часу, яка поставляється з вбудованим редактором карт. Він дозволяє редагувати та створювати власні карти, які використовуватимуться потім у грі. Редактор містить стандартний набір ігрових об'єктів, орієнтованих на забезпечення сюжету гри. Цей набір не підлягає доповненню власними об'єктами і є кінцевим. Крім того,

існують певні правила щодо розміщення об'єктів на карті. Отже , за призначенням редактор носить вузькоспеціалізований характер, має прихований формат карти, не забезпечений можливостями видозміни чи доповнення бібліотеки об'єктів, що унеможливлює його використання іншими розробниками у процесі створення нових ігор.

Розглянемо більш потужний редактор, який поставляється разом з грою Heroes 3. У редакторі дозволяється будувати карту гри з наперед визначених наборів об'єктів – набору доріг, перешкод, дерев, трави тощо. Об'єкти кожного набору мають фіксовані властивості й чітко визначене призначення. Створення власних об'єктів не передбачено. Редактор підтримує лише кілька фіксованих розмірів карт. Формат файлу карти є прихованим. Тому застосування редактора для створення власних ігор також є неможливим.

Слід окремо звернути увагу на те, що сьогодні з'явилися системи автоматизованої розробки ігор. До них належить Microsoft XNA Game Studio, призначена для розробки ігор під консолі XBox та ОС Windows. Система має розвинутий редактор рівнів. Унікальною у своєму роді є система SkyStudio фірми Skyfallen Entertainment, що являє собою універсальне інтегроване середовище розробки ігор, яке не залежить від платформи й ігрового каркасу; генерує програмний код і має широкий спектр можливостей. Проте подібні сучасні середовища ще не набули популярності й поширення, а переважна більшість ігор пишеться традиційно, з використанням самостійно розроблених редакторів рівнів. Автоматизовані системи мають власні мови скриптів, методи програмування фізичних явищ, логіки та графіки однак мають і певні обмеження на розширення й удосконалення бібліотеки типових об'єктів. Отже, актуальною вбачаємо розробку графічного редактора ігрових карт, модель якого носитиме відкритий характер, що забезпечить можливість розширення набору об'єктів власними COM-компонентами і забезпечить універсальність характеристик редактора.

Постановка завдання

Під універсальністю розроблюваного редактора розуміємо: розширення функціональних можливостей; незалежність від жанрів та сюжетів ігор; можливість налаштування під конкретні окремі ситуації спеціалізованого характеру; незалежність від ігрових каркасів. Прихованість формату вихідного файлу карти реалізуємо з метою захисту авторських прав, проте редактор має бути забезпеченим засобами зчитування файлу карти.

Підтримка розширення функціональних можливостей редактора полягає в тому, що редактор є набором COM-компонентів [2], які взаємодіють між собою через документовані інтерфейси. Отже, будь-який компонент може бути легко замінений на інший, більш модернізований чи спеціалізований. Крім того, забезпечується можливість додавання нових компонентів, створених розробниками ігор з метою налаштування редактора під свої потреби. Додані компоненти заносяться в актив бібліотеки для повторного використання. Тому розроблюваний редактор повинен мати повністю компонентну архітектуру.

Усі сучасні ігри поєднує їх спільна структура [3]. Гра – це набір об'єктів, які взаємодіють між собою. Кожен з об'єктів має визначений набір властивостей і певну логіку поведінки. Незалежність розроблюваного редактора від жанрів та сюжетів ігор полягає в можливості засобами редактора створювати нові ігрові об'єкти, задавати їм певні властивості та логіку поведінки й використовувати у процесі складання ігрових карт (навіть об'єкт «трава» на карті може мати властивості: колір, небезпечність, не кажучи про більш складні об'єкти) [3].

Можливість налаштування редактора на описання ситуацій з подальшим перетворенням універсальних можливостей у спеціалізовані вважаємо головною характеристикою редактора. З одного боку редактор забезпечує можливість створення різних ігрових об'єктів, але це дозволяється лише на етапі розробки моделей гри. Кінцевий варіант редактора повинен поставлятися в повністю спеціалізованому вигляді, щоб забезпечити коректність режимів роботи гри. Ця проблема може вирішуватися двома шляхами: забезпеченням можливості заміни компонентів (такий підхід передбачає створення двох копій компонента:

універсальної – для етапу розробки гри та спеціалізованої – для кінцевого програмного продукту), що, у свою чергу, вимагає надлишкових системних ресурсів; створенням компонентів, які через файли конфігурації можна налаштовувати на різні режими роботи, причому розробник гри може у визначати синтаксис подібних конфігураційних файлів.

Сьогодні існує багато ігрових каркасів, які відрізняються принципами функціонування, використання та спеціалізованими можливостями [4]. Графічні ефекти, які підтримує один каркас, можуть взагалі не підтримуватися іншими чи виглядати в них по-іншому. Якщо графічний редактор працюватиме на певному ігровому каркасі, то цей же каркас його й обмежуватиме. Якщо розробник використає інший ігровий каркас, то він буде змушений створювати графічні ефекти двічі на різних каркасах: для редактора і для самої гри. Тому було вирішено, що редактор не повинен бути орієнтованим за замовчуванням на конкретний каркас. Розробник один раз створює для редактора компонент ініціалізації каркасу засобами власного каркасу. Причому такий компонент стане єдиним не лише для конкретного розробника, а й для всіх розробників, що використовують цей ігровий каркас. Отже, можна стверджувати, що розроблений редактор являтиме собою надбудову над певним ігровим каркасом. Варто зазначити, що ігрові каркаси є високорівневими засобами розробки ігор, тому створення компоненту ініціалізації не є складною задачею.

Формат вихідного файлу карти має бути прихованим. Це дозволить дизайнерам створювати ігрові карти для існуючих ігор без порушення авторських прав. У редакторі передбачено два обов'язкових системних компоненти, один з яких зберігає карту у файлі, а інший зчитує інформацію з нього. Звісно, як і будь-який інший компонент редактора, ці компоненти можна додатково удосконалити чи замінити власними. Компонент зчитування файлу карти редактора використовуватиметься безпосередньо у грі. Аналогічно можливе також використання будь-яких інших компонентів редактора. Отже, розробник гри лише один раз повинен реалізувати необхідні графічні та інші можливості карти засобами свого каркасу, програмна підтримка їх використання забезпечується засобами автоматизації графічного редактора.

Розробка структури редактора

Базовим компонентом графічного редактора є реєстратор інших компонентів UCWERegistrar. Цей компонент містить динамічний список компонентів, які на час роботи редактора перебувають у активному режимі. Це забезпечує принцип: «кожен компонент може зареєструватися в списку і зв'язатися з кожним активним компонентом бібліотеки». Завдяки такому принципу, можливості редактора можна розширювати в будь-якому напрямку. Інтерфейси компонента UCWERegistrar наведені в табл. 1.

Таблиця 1

Інтерфейси компонента UCWERegistrar

Назва інтерфейсу	Назва функції	Призначення функції
ICUWERegistrar	Registering	Викликається компонентом для реєстрації себе в списку компонентів
	UnRegistering	Викликається компонентом для видалення себе зі списку зареєстрованих компонентів
	GetComponent	Зареєстрований компонент завжди містить посилання на інтерфейс ICUWERegistrar. Через цю функцію він може запросити інтерфейс IUnknown будь-якого іншого зареєстрованого компонента.
	GetComponentsList	Повертає список усіх активних компонентів редактора.

Головним компонентом редактора є компонент UCWEEditor, який створює вікно

головного меню програми й відповідає за завантаження інших компонентів, список яких вказаний у конфігураційному файлі. Одним із інтерфейсів цього компонента є інтерфейс IUCWEComponent. Це головний інтерфейс, який мають реалізувати всі компоненти, щоб підтримуватися редактором. Інтерфейс IUCWEComponent містить основні функції керування компонентом.

Ще одним підтримуваним інтерфейсом є IUCWEMenu. Його наявність засвідчує про підтримку компонентом головного меню, а інші компоненти можуть додавати в це меню власні опції, використовуючи інтерфейс IUCWEMenuItem. Інтерфейс IUCEMenu зберігає список інтерфейсів IUCWEMenuItem компонентів редактора й забезпечує виклики їх функцій відповідно до обраного елемента меню. Список інтерфейсів, що підтримуються компонентом UCWEEditor наведено в табл. 2.

Таблиця 2

Список інтерфейсів компонента UCWEEditor

Назва інтерфейсу	Назва функції	Призначення функції
IUCWEComponent (функції цього компонента є однаковими для кожного з компонентів редактора)	Registering	Будь-який компонент-завантажувач повинен викликати цю функцію завантажуваного компонента, щоб завантажуваний компонент міг зареєструватися, викликавши функцію ICUWERegistrar.Registering.
	Init	Після виклику Registering компонент-завантажувач повинен викликати цю функцію завантажуваного компонента, щоб останній міг ініціалізуватися.
	Do	Ця функція викликається компонентами-завантажувачами, щоб завантажені ними компоненти могли виконувати свої функції.
	UnInit	Функція деініціалізації. Викликається компонентом-завантажувачем.
	UnRegistering	Функція дереєстрації. Викликається компонентом-завантажувачем, щоб завантажуваний компонент міг звільнити раніше зайнятий компонент UCWERegistrar.
	GetData	Функція викликається компонентом для збереження оновлених даних карти на диску.
	SetData	Функція викликається, коли завантажувач карти зчитав дані відповідного компонента і передає останньому його збережені на диску за допомогою функції GetData дані карти.
IUCWEMenu	AddItem	Будь-який компонент, маючи покажчик інтерфейсу IUCWEMenu іншого компонента, може додати в меню останнього власні опції, передавши в функцію AddItem реалізований інтерфейс IUCWEMenuItem.
	DeleteItem	Компонент, який додав опцію в меню іншого компонента, може видалити свою опцію, викликавши цю функцію.

Компонент UCWEMap додає опції відкриття та створення карти в меню компонента UCWEEditor, реалізує ці опції та може завантажувати інші компоненти. UCWEMap на початку своєї роботи завантажує три компоненти: UCWEEngine, UCWESaver та UCWEOpener. UCWEEngine – це компонент ініціалізації ігрового каркасу та обробки введення даних, можливості якого користувач використовуватиме в редакторі у процесі створення карт. UCWESaver забезпечує збереження ігрової карти у файлі на диску, а UCWEOpener відповідає за коректне завантаження. UCWEOpener використовується безпосередньо в грі для завантаження карт. UCWEMap підтримує інтерфейс IUCWEComponent, як і компонент UCWEEditor, і має аналогічне призначення функцій

інтерфейсу.

Під час вибору в меню компонента UCWEEditor опції створення чи відкриття карти UCWEMap завантажує компоненти UCWEObjectsManager та UCWELayersManager. UCWEObjectsManager призначений для створення, зберігання та видалення ігрових об'єктів. Компонент зберігає список екземплярів ігрових об'єктів, які потім користувач може вибирати й розташовувати на полі карти. Рис. 1 ілюструє можливості діалогового вікна компонента UCWEObjectsManager.

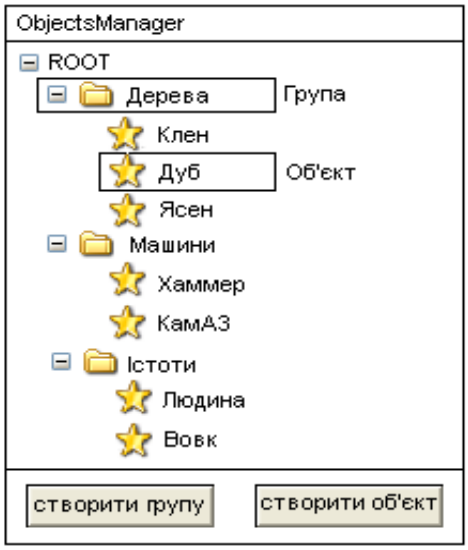

Рис. 1. Діалогове вікно компонента UCWEObjectsManager

Усі об'єкти в списку структуровано поділено на групи. Група не несе кінцевої інформації й не є компонентом, а служить для зручного візуального впорядкування об'єктів у списку. Інтерфейси, які підтримує компонент UCWEObjectsManager, наведено в табл. 3.

Таблиця 3

Інтерфейси компонента UCWEObjectsManager

Назва інтерфейсу	Назва функції	Призначення функції
IUCWEComponent	Функції інтерфейсу описані в табл. 2	
IUCWEObjectsManager	CopyCurrentObject	Повертає копію виділеного об'єкта в списку екземплярів об'єктів
	GetCurrentObject	Повертає вказівник на виділений об'єкт у списку екземплярів об'єктів
	GetObjectsList	Повертає список екземплярів об'єктів

Кожен об'єкт ініціалізується в редакторі компонентом UCWEObject, тобто UCWEObjectsManager зберігає список компонентів UCWEObject. Користувач у процесі створення ігрової карти обирає необхідний об'єкт і розташовує його на карті. Структура карти складається з шарів. У певний момент часу існує лише один активний шар, тому будь-який доданий об'єкт обов'язково належатиме одному конкретному шару. Шари слугують засобом групування доданих до карти об'єктів з метою організації зручного керування ними. Шар у редакторі подано компонентом UCWELayer. За редагування шарів відповідає компонент UCWELayersManager, який зберігає список усіх створених шарів. Діалогове вікно компонента UCWELayersManager зображено на рис. 2.

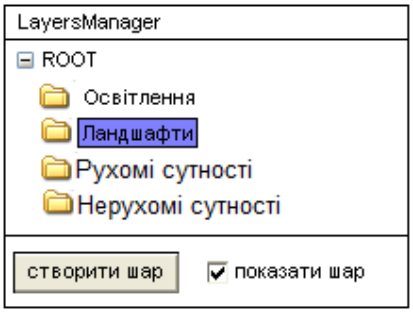

Рис. 2. Діалогове вікно компонента UCWELayersManager

Інтерфейси, які підтримує компонент UCWELayersManager, подано в табл. 4.

Таблиця 4

Інтерфейси компонента UCWELayersManager

Назва інтерфейсу	Назва функції	Призначення функції
IUCWEComponent	Функції інтерфейсу описані в табл. 2	
IUCWELayersManager	GetCurrentLayer	Повертає покажчик на інтерфейс IUCWELayer компонента UCWELayer, що ініціалізує поточний робочий шар карти

Компонент UCWELayersManager керує списком компонентів UCWELayer. Компонент UCWELayer ініціалізує конкретний шар карти. Він містить список компонентів UCWEObject – копій об'єктів зі списку в UCWEObjectsManager, доданих до карти. Дія користувача на карті перехоплюється компонентом UCWEEngine, який звертається до відповідної функції UCWELayer, а та, у свою чергу, додає до свого списку новий об'єкт, який передається їй як аргумент. Коли користувач натискає клавішу Delete, то компонент UCWEEngine викликає потрібну функцію компонента UCWELayer, яка видаляє з активного списку конкретний (виділений) об'єкт. Аналогічно відбуваються усі дії над об'єктами карти. Компонент UCWEEngine відслідковує введення даних користувачем і, викликаючи функції відповідних компонентів, оперує об'єктами на карті. Список інтерфейсів компонента UCWELayer подано в табл. 5.

Таблиця 5

Список інтерфейсів компонента UCWELayer

Назва інтерфейсу	Назва функції	Призначення функції
IUCWEComponent	Функції інтерфейсу описані в табл. 2	
IUCWELayer	AddObject	Додати новий об'єкт до шару
	DeleteObject	Видалити об'єкт зі списку
	SetCurrentObject	Зробити об'єкт на карті поточним
	GetCurrentObject	Повернути покажчик на виділений об'єкт

Компонент UCWEObject є ігровим об'єктом карти. Кожен об'єкт – це сутність, яка акумулює певні властивості. Властивість ініціалізується в редакторі компонентами UCWEProperty. Отже, компонент UCWEObject містить набір компонентів UCWEProperty. Компоненти UCWEProperty самостійно створює користувач редактора відповідно до сюжету, яка розробляється гри засобами обраного ігрового каркасу. Ініціалізовані компоненти автоматично використовуються у грі. Інтерфейси компонента UCWEObject подано в табл. 6.

Таблиця 6

Список інтерфейсів компонента UCWEObject

Назва інтерфейсу	Назва функції	Призначення функції
IUCWEComponent	Функції інтерфейсу описані в табл. 2	
IUCWEObject	GetPropertiesList	Повертає список компонентів UCWEProperty як властивостей об'єкта

Для кожної конкретної гри компоненти UCWEProperty можуть бути своїми. Одним із компонентів UCWEProperty можна реалізувати зовнішній вигляд об'єкта, іншим – числове значення параметра і т. п. Компонент обов'язково підтримує стандартний інтерфейс IUCWEComponent. Слід зауважити, що кожен із компонентів редактора може пітримувати інші власні інтерфейси. Особливо цей принцип стосується компонентів UCWEProperty у процесі реалізації тісної взаємодії з компонентом UCWEEngine, оскільки вони створюються засобами одного ігрового каркаса. Загальна модель графічного компонентного редактора зображена на рис. 3.

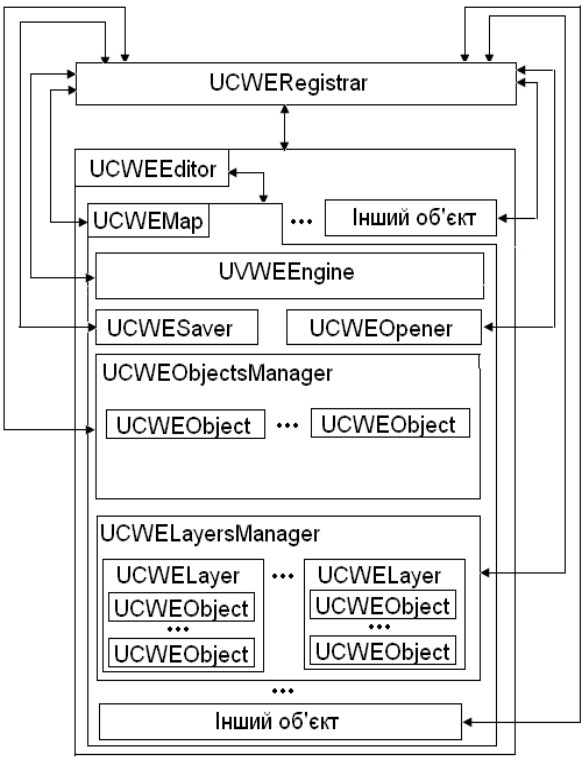

Рис. 3. Загальна модель графічного компонентного редактора

Зауважимо, що під вкладеністю одного компонента в інший розуміємо те, що перший компонент створює (завантажує) останній. Отже, час життя завантажуваного компонента є вкладеним у час життя компонента-завантажувача. Компоненти взаємодіють між собою за технологією COM, реалізуючи архітектурний принцип керування об'єктами. Так, запуск до виконання кінцевого файлу забезпечує завантаження компонентів UCWERegistrar та UCWEEditor. UCWEEditor завантажує безпосередньо компонент UCWEMap та інші компоненти, що перераховані в конфігураційному файлі компонента UCWEEditor. UCWEMap, у свою чергу, завантажує відповідні компоненти UCWESaver, UCWEOpener, UCWEEngine. Робочий файл UCWE.exe викликає функцію IUCWEComponent::Registering компонента UCWEEditor, який викликає ці ж функції в усіх своїх завантажених компонентах, спонукаючи їх до виконання аналогічних дій аж до найнижчого рівня

вкладеності. Аналогічно файл UZWE.exe викликає функцію IUCWEComponent::Init. Компонент UCWEEngine, створивши власними засобами вікно рендерингу, циклічно викликає функцію IUCWEComponent::Do компонента UCWEEditor. Останній, у свою чергу, забезпечує виклик цих же функцій в усіх активних компонентах, рухаючись послідовно до останнього рівня вкладеності. Якщо користувач обирає опцію меню створення карти, то UCWEMap додатково довантажує компоненти UCWEObjectsManager та UCWELayersManager, і циклічні виклики функцій Do повторюються знову. Коли управління дійде до функцій Do компонентів UCWEPtorerty, то, залежно від свого призначення, один з них прорисує сам об'єкт, інший надасть йому відтінку чи графічно реалізує інший визначений ефект. Так покроково відбувається створення об'єктів, які утворюють карту. Коли користувач закриває редактор, то спочатку викликаться функції UnInit, а потім функції UnRegistering усіх компонентів редактора у зворотній до завантаження послідовності. Якщо користувач вибирає опцію збереження файла, то компонент UCWEMap викликає функції збереження компонента UCWESaver, забезпечуючи ланцюжковий виклик функцій GetData всіх вкладених компонентів редактора, і у файлі зберігаються всі дані, які будуть надані цими функціями. За аналогічною схемою працює й алгоритм відкриття файлу карти з викликом компонента UCWEOpener.

Компоненти редактора можуть створювати робочі потоки, тому всі компоненти повинні створюватися з урахуванням багатопотоковості. Стандартною для розробленого редактора є модель вільних потоків COM (Free threaded model).

Використання ланцюжкової схеми реалізації архітектурного принципу керування об'єктами забезпечує надійну роботу графічного редактора й дозволяє уникати створення колізій та конфліктних ситуацій у процесі опрацювання даних.

Висновок

Розроблено універсальний графічний компонентний редактор ігрових карт, який узагальнив можливості існуючих редакторів, забезпечивши таким чином незалежність від сюжетів та жанрів ігор. Вирішено проблему незалежності редактора від конкретних ігрових каркасів. Завдяки застосуванню компонентної технології, редактор забезпечує розширення своїх функціональних можливостей у системі відкритого типу. Наявні засоби захисту файла карти дозволяють дизайнерам реалізувати авторські ідеї. Редактор може бути використаний для моделювання систем об'єктів у дизайнерських та інженерних розробках різних напрямків.

СПИСОК ЛІТЕРАТУРИ

1. Todd Barron. Strategy game programming with DirectX 9.0. – Wordware Publishing Inc.: 2003. – 350 с.

2. Дональд Бокс. Сущность технологии COM. Библиотека программиста. – СПб.: Питер, 2001. – 400 с.: ил.

3. Ламот Андре. Программирование игр для Windows. Советы профессионала, 2-е изд. : Пер. с англ. – М. : Издательский дом "Вильямс", 2004. – 880 с. : ил. – Парал. тит. англ.

4. Марк Зальцман. Компьютерные игры: как это делается.: Издательский Дом "Логрус", 2000. – 540 с. ил.

Войтко Вікторія Володимирівна - к. т. н., доцент кафедри програмного забезпечення, тел. +38(0432) 598 483.

Денисюк Алла Василівна – асистент кафедри програмного забезпечення, тел. +380961104084.

Луцишин Геннадій Леонідович – студент 4-го курсу факультету комп'ютерного інтелекту інституту інформаційних технологій та комп'ютерної інженерії, тел. +380969721736, e-mail: Hennadiy@bk.ru.

Вінницький національний технічний університет.