УДК 004.4.273

В. В. ВОЙТКО, Д. І. КАТЄЛЬНІКОВ, І. Р. АРСЕНЮК, Г. Л. ЛУЦИШИН

УНІВЕРСАЛЬНИЙ РЕДАКТОР ГРАФІЧНИХ ІГРОВИХ СЦЕН

Вінницький національний технічний університет,
Хмельницьке шосе 95, 21001, м. Вінниця, Україна
E-mail: Hennadiy@bk.ru

Анотація. У статті запропоновано принципи реалізації універсального редактора графічних ігрових сцен, орієнтованого на використання довільних ігрових каркасів та розширення функціональних можливостей за рахунок реалізації компонентної архітектури. Розроблено моделі редактора та структури конфігураційних файлів базових компонентів.

Аннотация. В статье предложено принципы реализации универсального редактора графических игровых сцен, ориентированного на использование любых игровых каркасов и расширение функциональных возможностей за счет реализации компонентной архитектуры. Разработано модели редактора и структуры конфигурационных файлов базовых компонентов.

Abstract. In article it is offered principles of realisation of the universal editor of the graphic game scenes focused on use of any game skeletons and expansion of functionality at the expense of realisation of componental architecture. It is developed models of the editor and structure of configuration files of base components.

Ключові слова. Редактор ігрових сцен, ігрові каркаси, автоматизація розробки ігор, ігрові сутності, компонентна архітектура, XML-скрипт, .NET Framework.

ВСТУП

Сьогодні стрімкий розвиток комп'ютерних технологій обумовлює появу широкого кола мультимедійних продуктів, спрямованих на досягнення наукової, навчальної, просвітницької, а також розважальної та комерційної мети. Сучасні стратегічні ігрові програми ефективно використовуються як емулятори та засоби моделювання проблемних ситуацій у процесі ведення досліджень в різних галузях діяльності людини.

Розробка графічних комп'ютерних ігор передбачає попереднє створення редактора ігрових сцен, що вимагає додаткових часових затрат у межах прогнозованого терміну розробки кінцевого програмного продукту. Сучасні редактори ігрових сцен характеризуються обмеженими й жорстко визначеними функціональними можливостями або орієнтовані під конкретне програмне забезпечення [1]. Тому використання існуючих графічних редакторів для написання нових ігрових програм є обмеженим, а часто й неможливим. Це обумовлює актуальність розробки принципів створення редакторів графічних ігрових сцен, спеціалізовані характеристики яких носили б універсальний характер, що зумовить перспективність їх широкого використання в засобах мультимедіа.

Отже, метою роботи є автоматизація процесу створення графічних ігрових сцен засобами універсального редактора. Під об'єктом дослідження розуміємо принципи та методи побудови широкопрофільних редакторів графічних ігрових сцен. Предметом дослідження постають засоби створення ігрових сцен та методи реалізації в них графічних об'єктів. Головними задачами роботи вважаємо розробку редактора графічних ігрових сцен і забезпечення його універсальних характеристик.

АНАЛІЗ ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ УНІВЕРСАЛЬНОГО РЕДАКТОРА

Під універсальністю розроблюваного редактора розуміємо розширення його функціональних можливостей, незалежність від жанрів та сюжетів ігор, можливість налаштування під часткові ситуації спеціалізованого характеру, незалежність від ігрових каркасів. Редактор орієнтований на реалізацію .NET-технології. Підтримка розширених функціональних можливостей забезпечується набором взаємодіючих об'єктно-орієнтованих класів, що реалізують логічні компоненти, у бібліотеках динамічної компоновки [2, 3]. Таким чином, будь-який компонент може бути легко замінений іншим, більш спеціалізованим чи модернізованим, з метою налаштування редактора під визначені потреби. Додані

компоненти заносяться в архів бібліотеки для повторного використання. Незалежність редактора від жанрів та сюжетів ігор забезпечується шляхом створення нових ігрових об'єктів, реалізацією функцій налаштування їх властивостей та логіки поведінки відповідно до вимог обраних ігрових сцен. Можливість налаштування редактора під вимоги часткових ситуацій з ефективним приведенням універсальних можливостей до спеціалізованих вважаємо головною характеристикою редактора.

На сьогодні існує багато ігрових та графічних каркасів, які відрізняються принципами функціонування та спеціалізованими можливостями, серед яких популярними є: Nebula Device, OGRE, XNA Game Studio, NeoAxis Engine, Unreal Tournament Engine [4, 5]. Ігровий каркас – це центральний програмний компонент інтерактивних комп'ютерних додатків, один з компонентів якого обробляє графіку в режимі реального часу. Графічні ефекти, які підтримує один каркас, можуть частково чи повністю не підтримуватися іншими. Якщо графічний редактор працюватиме на довільному ігровому каркасі, то цей же каркас обмежуватиме його функціональність. Тому важливо, щоб редактор не був за замовчуванням орієнтованим під конкретний тип каркасу. Тоді розробник на початку процесу проектування програмного продукту створює компонент, який ідентифікує каркас у середовищі редактора засобами самого каркасу. При чому такий компонент може стати спільним у багаторазовому використанні не лише для конкретного розробника, а для всіх користувачів, які працюватимуть з обраним ігровим каркасом. Слід зауважити, що ігрові каркаси є високорівневими програмними засобами, тому створення такого компонента не вважають складною задачею програміста.

Графічний редактор призначений для реалізації визначеного набору сутностей. Під сутністю розуміємо об'єкт ігрової логіки на сцені, а саме ігрову дійову особу, джерело світла, ландшафт і т. п., тобто все те, що може бути розміщеним на сцені чи мати до неї певне відношення. Група сутностей визначається компонентом, який, у свою чергу, може містити дочірні групи та сутності. Тобто множина сутностей сцени впорядковується ієрархічно.

Визначимо базові функціональні можливості редактора, які є спільними для всіх редакторів ігрових сцен, а, отже, забезпечують універсальність його характеристик. До них належить відображення списку сутностей, які вже додані до ігрової сцени, та списку сутностей, які у перспективі можуть бути доданими до сцени; забезпечення можливості групування сутностей в обох списках; підтримка процесів виділення, видалення, додавання сутностей та їх груп в обох списках; підтримка процесів переміщення сутностей та їх груп у межах батьківської групи; відображення та програмна підтримка можливості оновлення списку властивостей сутностей та їх груп; підтримка процесів відображення, перейменування та приховування сутностей та їх груп у доданому списку; супроводження кожної з розглянутих операцій повідомленнями типу "Before" та "After" з метою реалізації можливості відміни команди на виконання операції.

Усі процеси повинні забезпечуватися як шляхом програмної реалізації, так і шляхом підтримки інтерфейсної взаємодії користувача. Окремо слід забезпечити підтримку головного меню, яке може бути розширене опціями сторонніх компонентів. Редактор має підтримувати процеси створення, відкривання, закривання та збереження ігрових сцен. У базовий набір компонентів потрібно також включити компонент читання/запису вмісту сцени у спеціалізований файл. Стандарт редактора повинен забезпечувати можливість реєстрації активних компонентів у своєму середовищі з метою реалізації їх подальшої взаємодії. Крім того, важливою є умова перспективного розширення редактора власними сутностями та їх групами.

РОЗРОБКА УЗАГАЛЬНЕНОЇ МОДЕЛІ УНІВЕРСАЛЬНОГО РЕДАКТОРА

Узагальнена модель реалізації універсального редактора графічних ігрових сцен містить ідентифікований набір компонентів, необхідних для забезпечення визначених функціональних можливостей та характеристик. Загальну модель графічного редактора можна подати у вигляді кортежу (1):

$$M = \{E, FR, FW, SM, ES, GS, IM, EI, GI, A\}, \quad (1)$$

де E – головний компонент редактора (Editor);
FR – об'єкт читання файлу сцени (FileReader);
FW – об'єкт збереження файлу сцени (FileWriter);
SM – менеджер (SampleManager) зразків сутностей;
ES – зразки сутностей (EntitySample), що потенційно можуть бути додані до сцени;
GS – зразки груп сутностей (GroupSample), що потенційно можуть бути додані до сцени;
IM – менеджер (InstanceManager) екземплярів сутностей;
EI – екземпляри сутностей (EntityInstance), що вже додані до сцени;

GI – екземпляри груп сутностей (GroupInstance), що вже додані до сцени;
A – інші додаткові компоненти редактора (компонент ігрового каркасу і т. п.).

Загальну модель редактора у вигляді UML-діаграми класів наведено на рисунку 1.

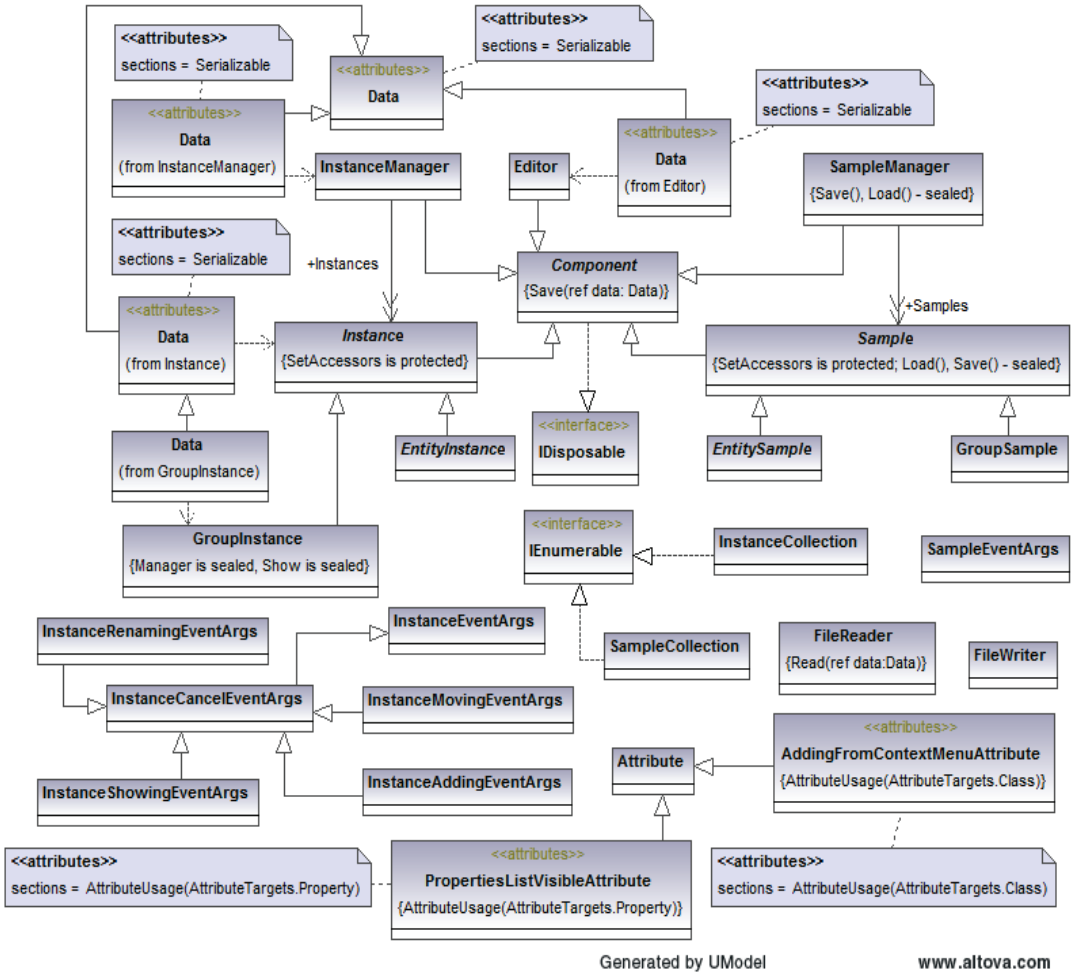

Рис. 1. UML-діаграма класів загальної моделі редактора

У редакторі розроблено власний формат компонента, який репрезентується базовим абстрактним класом Component. Клас Component декларує метод ініціалізації компонента в редакторі, методи збереження компонентів у файлі сцени; подію (event) Disposed, призначену для виконання цих процесів Editor викликає функції для всіх компонентів, які зберігаються в його списку активних компонентів. Аналогічно працює й процес ініціалізації.

Іншим стандартним компонентом редактора є SampleManager, основна задача якого полягає у підтримці списку зразків сутностей, які можуть бути вибрані та додані до сцени. Цей список може бути згрупованим, тобто компонент SampleManager керує також зразками груп. Під групою розуміємо спеціалізований структурний компонент, що зберігає список вкладених компонентів-груп та компонентів-сутностей. Таким чином, редактор підтримує ієрархічне групування сутностей, що є головним призначенням компонента-групи.

Компонент-сутність, на відміну від компонента групи, не є контейнерним, а отже презентує кінцевий вузол ієрархії. Компоненти-групи та компоненти-сутності наслідують абстрактний клас Sample, що забезпечує поліморфність менеджерного управління. Компонент InstanceManager додатково забезпечує можливості додавання, видалення, перейменування, приховування/відображення компонентів менеджера. Редактор визначає базові класи EntityInstance та GroupInstance для полегшення процесу створення власних користувацьких компонентів. Поліморфність менеджерного управління обраними типами компонентів забезпечується функціональними можливостями базового класу Instance. Крім того, передбачається компонентне розширення класів EntityInstance, GroupInstance, EntitySample та GroupSample продукцією сторонніх розробників. Слід зауважити, що компонент Editor дозволяє

підключення обраного масиву компонентів на етапі ініціалізації даних, що забезпечується складанням списку необхідних компонентів у конфігураційному XML-файлі. Стандартна функціональність редактора надає можливість створення ієрархічного списку груп та сутностей у менеджері SampleManager та списку груп у менеджері InstanceManager на етапі ініціалізації менеджерів.

РОЗРОБКА СТРУКТУРИ КОНФІГУРАЦІЙНИХ ФАЙЛІВ КОМПОНЕНТІВ

Розглянемо структуру конфігураційного XML-файлу компонента SampleManager, наведену на рисунку 2.

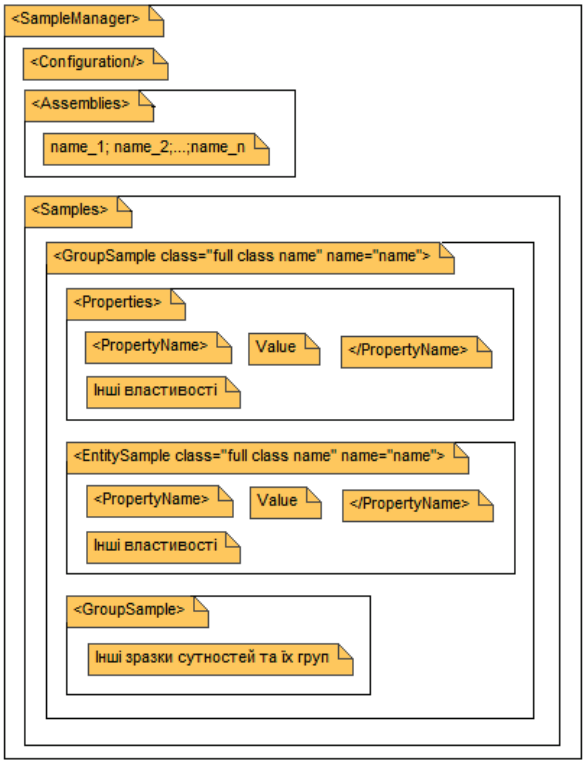

Рис. 2. Структура конфігураційного файлу компонента SampleManager

Компоненти SampleManager та InstanceManager у процесі обробки події створення чи завантаження сцени, у свою чергу, можуть створювати ієрархічні списки сутностей та груп сутностей. Потрібні ієрархічні списки вказуються в конфігураційних файлах відповідних компонентів.

У вузлі “Configuration” встановлюються значення конфігурації самого компонента SampleManager. У вузлі “Assemblies” вказуються збірки класів груп та сутностей, перерахованих у вузлі “Samples”. Ієрархічний список екземплярів груп та сутностей описується у вузлі “Samples”. Для екземпляру групи вказується ім’я класу та ім’я об’єкту. У дочірньому вузлі “Properties” задаються початкові значення public-властивостей об’єкту. Крім того, група може містити в собі як вкладені сутності, так і інші групи (рис. 2). Структура конфігураційного файлу компонента InstanceManager відрізняється відсутністю опису сутностей. Додавання екземпляру сутності до сцени вважається більш складним процессом.

Конфігурація компонента Editor задається у вузлі “Configuration”. Компоненти, які активуються в редакторі, вказуються у вузлі “Components”, де ідентифікується ім’я збірки та перелік імен класів компонентів, що входять до обраної збірки.

РОЗРОБКА МОДЕЛІ РОЗГОРТУВАННЯ РЕДАКТОРА

Діаграму моделі розгортування наведено на рисунку 3. Єдиними обмеженнями розгортування модулів редактора є обов’язкова умова збереження конфігураційних файлів в одному каталозі зі збірками відповідних компонентів. Каталоги всіх збірок (керуючих компонентів та сутностей) повинні бути вказані в конфігураційному файлі.

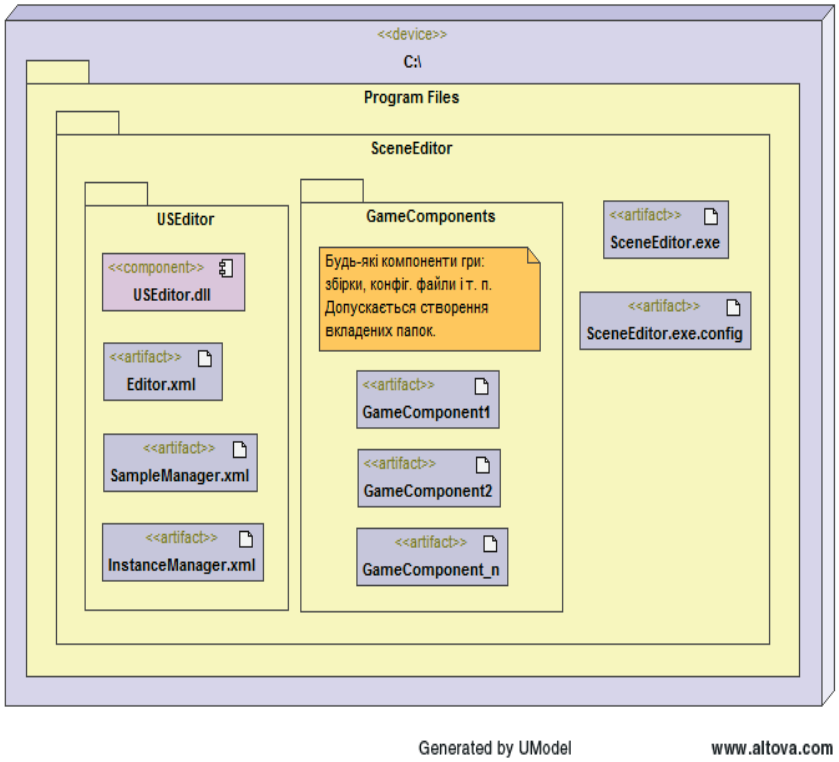

Рис. 3. Діаграма розгортування компонентів редактора

USEditor.dll – бібліотека динамічної компоновки, в якій знаходяться усі стандартні компоненти Editor, SampleManager, InstanceManager, класи читання/запису даних у файл сцени й інші допоміжні класи (колекції, атрибути, класи для зберігання аргументів подій тощо.). До базової поставки редактора також входять об'єкти з каталогів USEditor та GameComponent, де розміщуються спеціалізовані компоненти гри, в тому числі компонент візуалізації сцени. Каталоги USEditor та GameComponents повинні бути вказані у файлі SceneEditor.exe.config як місця пошуку збірок.

Базовий пакет редактора розгортається методом XCopy. Робочий файл SceneEditor.exe створюється користувачем редактора – стороннім розробником. Компоненти SampleManager та InstanceManager завантажуються в редактор компонентом Editor, якщо вони зазначені у конфігураційному файлі Editor.xml.

ВИСНОВКИ

Сьогодні ігрова комп'ютерна індустрія – це напрямок стрімкого динамічного розвитку. Ігрові програмні продукти повністю увійшли в повсякденне життя суспільства. Їх комп'ютерна реалізація передбачає поєднання систем обробки графіки, звуку, фізичних та математичних розрахунків, штучного інтелекту. Розробка універсального каркасонезалежного графічного редактора є значним кроком у розвитку засобів автоматизації створення інтерактивних мультимедійних програмних продуктів.

Орієнтацію універсального редактора на платформу .NET вважаємо перевагою розробки, оскільки сьогодні спостерігається активний розвиток ігрових каркасів, спрямованих на .NET-технологію. Крім того, вибрана платформа надає широкі можливості компонентного розширення програмних систем.

Розроблений універсальний редактор узагальнив можливості існуючих редакторів, забезпечивши повну незалежність від сюжетів та жанрів ігор. Було вирішено проблему незалежності редактора від конкретних ігрових каркасів. Завдяки застосуванню компонентної технології редактор забезпечує розширення своїх функціональних можливостей у системі відкритого типу. Редактор може бути використаний для моделювання об'єктів у дизайнерських та інженерних роботах різного спрямування.

ЛІТЕРАТУРА

4. Strategy game programming with DirectX 9.0 / Todd B. // Wordware Publishing Inc. – 2003.
5. C# и платформа .NET. Библиотека программиста / Э. Троелсен — СПб.: Питер. – 2004. – 796 С.
6. C# 2005 и платформа .NET 3.0 для профессионалов / Кристиан Нейгел, Билл Ивьен, Джей Глинн,

Карли Уотсон, Морган Скиннер – М.: “Диалектика”. – 2008. – 1790 С.
7. Pro OGRE 3D Programming / Gregory Junker.: Apress. – 2006.
8. Professional XNA Game Programming for Xbox360 and Windows / Benjamin Nizchke.: Wrox Press. – 2007.

Надійшла до редакції 15.04.2010 р.

ВОЙТКО В. В. – к.т.н., доцент кафедри програмного забезпечення, Вінницький національний технічний університет, Вінниця, Україна.

КАТЄЛЬНІКОВ Д. І. – к.т.н., доцент кафедри програмного забезпечення, Вінницький національний технічний університет, Вінниця, Україна.

АРСЕНЮК І. Р. – к.т.н., доцент кафедри комп’ютерних наук, Вінницький національний технічний університет, Вінниця, Україна.

ЛУЦИШИН Г. Л. – студент 5-го курсу факультету комп’ютерного інтелекту інституту інформаційних технологій та комп’ютерної інженерії, Вінницький національний технічний університет, Вінниця, Україна.