УДК 536.24

# СТВОРЕННЯ ІГРОВИХ ДОДАТКІВ НА ОБ'ЄКТНО-ОРІЄНТОВАНІЙ МОВІ ПРОГРАМУВАННЯ С ++

**Дяченко Д. С.**
*Харківський національний автомобільно-дорожній університет*

Зміни, що відбуваються сьогодні в різних сферах професійного і суспільного життя, пов'язані перш за все з широким впровадженням засобів інформаційних та комунікаційних технологій. Вони вимагають від фахівця у галузі комп'ютерних наук вміння розробляти програмне забезпечення для розв'язування задач науки, техніки, економіки і управління; створювати і використовувати інформаційні моделі процесів і явищ; використовувати інформаційні технології у науковій, проектно-конструкторській, управлінській та фінансовій діяльності.

Одними з дисциплін, що формують теоретичну і практичну базу у професійній підготовці майбутніх інженерів, є програмування та подібні дисципліни: обчислювальна техніка, інформаційні технології та ін. На сучасному етапі ці дисципліни є невід'ємною частиною вищої освіти. Знання, вміння і навички, отримані студентами в курсі програмування та ін., необхідні студентам при засвоєнні учбового матеріалу більшості дисциплін, передбачених навчальними планами різних спеціальностей [1].

Програмування є однією з фундаментальних дисциплін, що формують теоретичну і практичну базу у професійній підготовці майбутніх спеціалістів ІТ галузей. Крім того, на сучасному етапі програмування є невід'ємною частиною освіти при підготовці математиків, фізиків, хіміків та ін., і тому відповідний курс є складовою навчальних планів цих спеціальностей. Знання, вміння і навички, отримані студентами в курсі програмування, необхідні їм при засвоєнні учбового матеріалу більшості дисциплін, передбачених навчальними планами різних спеціальностей [2].

Програмування – це перша професійно-орієнтована дисципліна, з якою зустрічаються майбутні програмісти під час навчання в

університеті. Від змісту курсу, від методики його навчання, від знань, умінь і навичок, які одержать студенти в цьому курсі, залежить їх подальше навчання, якість засвоєння дисциплін, заснованих на знаннях з програмування, успішність в майбутній професійній діяльності [2]. Тому якість викладання та якість засвоювання цієї дисципліни дуже важливі для майбутніх IT професіоналів.

Однією з провідних IT галузей є розробка та створювання комп'ютерних ігор. Світовий ринок комп'ютерних ігор у 2016 році перевищив позначку у 100 міліардів доларів США, а провідні фірми у цій галузі вже давно мають міліардні статки. Значна частина цих грошей є винагородженням для програмістів. Тому цей, здавалося б, легковажний бізнес насправді дає дуже велику кількість гарно оплачуваних робочих місць для програмістів, у тому числі і в Україні.

Програмування комп'ютерних ігор - це процес створення програмного коду з метою візуалізації ігрового "світу", взаємодії гравця з цим світом і пересування по ньому. Програмування ігор є глибоко командним процесом і підрозділяється на кілька спеціалізованих областей, кожна з яких відповідає за певну функціональну частину майбутньої гри.

Наприклад, провідний програміст займається зведенням всіх підрозділів ігрового движка в єдину працюючу систему, а програміст ігрової механіки - це людина, яка відповідає за програмну реалізацію всіх елементів комп'ютерних ігор. Програміст 3D-движка, програміст графічного движка - це фахівці, що відповідають за відображення ігрового світу на екрані, графічні ефекти і т. п. Програміст мережевого коду відповідає за взаємодію гри і гравця через мережу інтернет (або локальну мережу) з серверами оновлень, іншими гравцями і т.п.

Створення комп'ютерних ігор вимагає спеціальних знань, розуміння того, як працюють ігри та реалізуючи їх програми, великої кількості часу і бажання розробляти вихідний код. Навіть в безкоштовні готові рішення, доступні для використання, вкладені тисячі людино-годин програмування, не кажучи вже про комерційні движки, в які крім часу ще були вкладені і серйозні фінанси.

Значна частина молоді, зокрема, студенти-програмісти є палкими прихильниками комп'ютерних ігор, аж до участі у змаганнях з цих

ігор (від місцевих до світових чемпіонатів). Дехто з них прагнуть самостійно створювати комп'ютерні ігри.

Тому у нашого викладача виникла ідея поєднання вивчення дисципліни "Програмування" при підготовці програміста з розробкою комп'ютерних ігор різного рівня складності (для бажаючих та встигаючих студентів). Цьому сприяло те, що згідно Програми навчальної дисципліни "Програмування" на 2-му курсі вивчається об'єктно-орієнтоване програмування на мові С++, а програмний код більшості комп'ютерних ігор створений на цій мові.

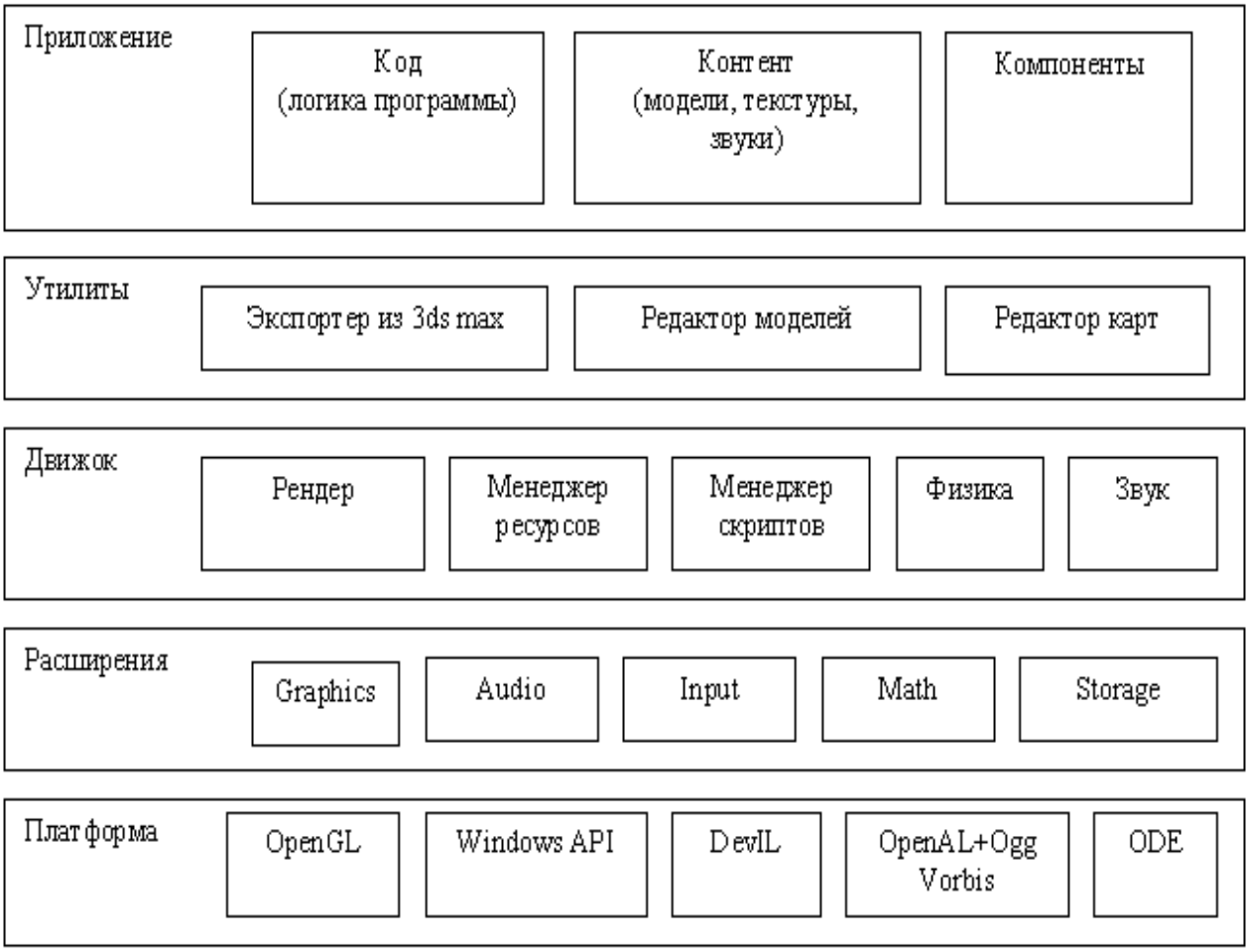


Рис. 1 – Схема створеного програмного проекту

Так як ми використовуємо частину готових бібліотек при розробці ігрових додатків, то деякі низькорівневі конструкції у нас вже є. Наш проект буде виглядати наступним чином (рис.1): на нижньому рівні будуть знаходяться бібліотеки OpenGL, Windows

API, DevIL, OpenAL + Ogg Vorbis, ODE, за ними прошарок з бібліотек і класів нашого движка Engine, який буде містити стандартні для кожного проекту класи і об'єкти, а далі йдуть допоміжні утиліти. На верхньому рівні буде створений додаток з розробленою ігровою логікою.

Наш додаток OpenGL, як і наступні додатки матиме наступну архітектуру: для кожної окремої операції ми будемо використовувати окрему функцію. Більшість з цих функцій будуть викликатися з WinMain, а деякі з MsgProc. У нашому прикладі ми будемо розглядати окремо кожну функцію и її вміст.

Кожна програма Windows повинна мати функцію WinMain. Ми будемо використовувати цю функцію з тією ж метою, що і завжди - як головну функцію програми. Розглянемо вихідний текст нашої функції WinMain:

Перше, що потрібно зробити, це створити головне вікно програми. Ми просто зробили виклик функції InitWindow і створили вікно.

Наступна операція - це ініціалізація пристрою: Init Compatible Context. У цій функції ми зробили попередню ініціалізацію OpenGL. Потім ми провели остаточну ініціалізацію в процедурі InitContext. Контекст в OpenGL – це кореневої об'єкт, який потрібно створити, щоб працювати з OpenGL, в цій функції вже включиться наш відеоприскорювач.

Кожна програма Windows має цикл обробки повідомлень. Обробка повідомлень потрібна для того, щоб відстежувати надходячи в вікно повідомлення від системи - клавіші, згортання і розгортання вікна і т. п.

Ініціалізація OpenGL дуже відрізняється від будь-яких інших комплектів 3d систем і має свою специфіку. Вся система ділиться на OpenGL-специфічні кореневі функції і платформ-специфічні функції. Таким чином, система працює тільки з графікою. Але щоб ініціалізувати OpenGL, необхідно виконати платформ-специфічні функції.

Будь-який додаток OpenGL має виконувати наступну послідовність дій:

**1. Ініціалізація.** На цьому етапі просто підключені всі необхідні динамічні бібліотеки і встановлюються константи. Також при

ініціалізації ми створили основне вікно програми. Ця платформ-специфічна операція, вона не відноситься до OpenGL, а тільки до нашого додатком.

**2. Створення OpenGL контекста.** OpenGL контекст вдає із себе віртуальний об'єкт, до якого прив'язана наш додаток. Таким чином, контекст є, по-перше, кореневих об'єктом OpenGL, і по-друге, сполучною ланкою між нашим додатком і системою OpenGL. В якійсь мірі, контекст OpenGL також є і віртуальним екраном, на якому здійснюється рендер різними функціями OpenGL.

*Створення контексту є платформ-специфічною операцією, функція створення контексту опосередковано вбудована в OpenGL, і для створення контексту існує певна методика.*

***3. Отримання функцій.*** *Сама система OpenGL являє собою набір функцій, але щоб використовувати функції, до них потрібно підключитися. Підключення функцій - це динамічне компонування функцій. У OpenGL недостатньо зробити просто #inlude і потім написати ім'я бібліотеки з функціями. Крім #include, потрібно ще і виконати команду для підключення функцій.*

*У нашій програмі ми використали бібліотеку GLEW, яка також називається як OpenGL Wrangler Library. Таким чином, підключивши бібліотеку GLEW командами #include, ми підключили всі необхідні функції OpenGL.*

**4. Використання функцій.** Після того, як функції були підключені, ми скористалися OpenGL і OpenGL-специфічними функціями. Ці функції мають характерний синтаксис і будь-яку функцію OpenGL можна легко ідентифікувати.

**5. Ініціалізація вікна InitWindow.** Так як ми користуємося функціями Windows, то необхідно на початку програми ввести відповідний #include. Цей рядок підключить необхідні API функції. Усередині функції InitWindow існує два основні виклики - це реєстрація класу вікна.

Зверніть увагу, що при створенні вікна можна вказати різні параметри, в тому числі його розмір.

Тепер здійснимо по порядку всі дії, які потрібні для ініціалізації OpenGL. Операція ініціалізації OpenGL проходить в два кроки. Розберемо перший з них, це операція створення сумісного контексту.

Для цієї операції ми створимо функцію InitCompatibleContext. Ця функція створить контекст таким чином, що контекст створюється на будь-яких пристроях OpenGL.

Установка сумісного OpenGL контексту відбувається в кілька кроків. Перед створенням контексту завжди встановлюється його формат.

Таким чином, функцією SetPixelFormat встановлюється формат кольору пікселів, формат z-буфера, а також інші параметри. Після цього функцією wglCreateContext створюється контекст. Вхідним параметром функції є hDC, тобто дескриптор вікна.

У нашому додатку ми створили вікно попередньо функцією InitWindow, так що інформація про дескрипторі вікна є. Завжди після функції wgl Create Context повинна слідувати функція wglMake Current.

Тепер настав ключовий момент ініціалізації, коли ми повинні прілінковати всі функції OpenGL до нашого додатком. Це робиться дуже просто командою glewInit.

Тепер доступ до функцій OpenGL отримано, і сумісний контекст більше не потрібний, так що ми звільнимо його за допомогою функції wglMakeCurrent с параметром NULL, а також функції wglDelete Context, і перейдемо до наступного розділу, в якому ми створимо основний контекст, тобто такий контекст, який буде працювати з останніми версіями OpenGL.

Спочатку ми створили сумісний контекст OpenGL. Другою дією створили основний контекст. Для цього в нашому додатку є функція InitContext. Можна було продовжувати користуватися сумісним контекстом, але для нашого додатку в цьому немає необхідності. Після створення сумісного контексту, для того щоб отримати доступ до функцій версій OpenGL 4.x був створити основний контекст.

Тепер залишається тільки скомпілювати і запустити додаток. Був розроблений вихідний код цього додатка, і проведена його компіляція.

В операційній системі повинен бути встановлений OpenGL версії не нижче 4.0. У поточних версіях Windows, наприклад Windows 7, а також Windows 8 все вже встановлено, і наш додаток успішно запускається і функціонує на них.

## Література

1. Симбірский Г. Д. Деякі аспекти викладання дисципліни "Програмування" та подібних в транспортних ВНЗ / Г.Д. Симбірский // Інформаційні технології і мехатроніка: міжнар. наук.-практ. конф.: тез. докладів. – Х.: ХНАДУ. - 2016. - С. 126-127.

2. Гришко Л.В. Концептуальні підходи до навчання основ програмування у вищій школі / Л.В. Гришко / Комп'ютерно-орієнтовані системи навчання: Зб. наук. праць / Редкол. – К.: НПУ ім. М.П. Драгоманова. – Випуск 8. – 2004. – С. 134-148.