РЕАЛІЗАЦІЯ ІГРОВИХ ДОДАТКІВ МОВОЮ С++

Дяченко Д.С., Магістр
Науковий керівник – ***Біндюг С.А.,*** асистент
Харківський національний автомобільно-дорожній університет

Зміни, що відбуваються сьогодні в різних сферах професійного та суспільного життя, пов'язані переважно з поширенням інформаційно-комунікаційних технологій. Тому фахівці з комп'ютерних наук повинні вміти розробляти програмне забезпечення для розв'язування задач у галузі науки, техніки, економіки та управління; створювати та використовувати інформаційні моделі процесів і явищ; використовувати інформаційні технології в науковій, проектній, управлінській та фінансовій діяльності.

Однією з дисциплін, які формують теоретичну та практичну основу професійної підготовки майбутніх інженерів, є програмування та споріднені з ним дисципліни, такі як інформатика та інформаційні технології. На сучасному етапі ці дисципліни є невід'ємною частиною вищої освіти. Знання, вміння та навички, набуті студентами при вивченні таких дисциплін, як програмування, необхідні для засвоєння навчального матеріалу з більшості дисциплін, передбачених навчальними планами різних спеціалізацій [1].

Програмування є однією з фундаментальних дисциплін, які формують теоретичну та практичну основу професійної підготовки майбутніх ІТ-фахівців. Крім того, на сучасному етапі програмування є невід'ємною частиною освіти для підготовки математиків, фізиків, хіміків тощо, тому відповідні курси входять до навчальних планів цих спеціальностей. Знання, вміння та навички, набуті студентами на курсах програмування, необхідні для засвоєння навчального матеріалу з більшості дисциплін, зазначених у навчальних планах різних спеціалізацій [2].

Програмування є першою професійно-орієнтованою дисципліною, з якою майбутні програмісти стикаються під час навчання в університеті. Зміст курсу, методика викладання та знання, уміння і навички, набуті студентами при вивченні цієї дисципліни, визначають їх подальше навчання, якість засвоєння ними дисципліни, що базується на знаннях з програмування, та їх успіх у майбутній професійній діяльності [2]. Тому якість освіти та якість навчання в цій галузі має велике значення для майбутніх ІТ-фахівців.

Однією з провідних ІТ-галузей є розробка та створення комп'ютерних ігор: у 2016 році світовий ринок комп'ютерних ігор перевищив позначку в 100 мільярдів доларів США, а найбільші компанії цієї індустрії вже давно стали мільярдерами.

Значна частина цього фінансування припадає на винагороду програмістам. Таким чином, цей, на перший погляд, несерйозний бізнес насправді забезпечує програмістів великою кількістю високооплачуваних робочих місць, у тому числі і в Україні.

Програмування комп’ютерних ігор - це процес візуалізації ігрового «світу» та створення програмного коду для взаємодії з цим світом і переміщення по ньому. Програмування ігор є глибоко командним процесом і поділяється на кілька спеціалізацій, кожна з яких відповідає за певну функціональну частину майбутньої гри. Наприклад, провідний програміст відповідає за об’єднання всіх відділів ігрового рушія в одну робочу систему, тоді як програміст ігрової механіки відповідає за програмну реалізацію всіх елементів комп’ютерної гри.

Програмісти 3D-рушія та програмісти графічного рушія - це фахівці, які відповідають за екранне представлення та графічні ефекти ігрового світу. Програмісти мережевого коду відповідають за взаємодію гри та гравців із серверами оновлень, іншими гравцями тощо через Інтернет (або локальну мережу).

Створення комп’ютерних ігор вимагає спеціальних знань, розуміння того, як працюють ігри і як реалізуються програми, багато часу і бажання розробляти вихідний код.

Навіть готові рішення, доступні безкоштовно, вимагають тисяч людино-годин програмування, не кажучи вже про комерційні рушії, які потребують серйозних інвестицій і часу. Багато молодих людей, особливо студентів-програмістів, захоплюються комп’ютерними іграми і беруть участь в ігрових змаганнях (від регіональних до світових). Деякі з них навіть хочуть створювати власні комп’ютерні ігри. Тому у наших викладачів виникла ідея поєднати вивчення дисципліни «програмування» для отримання професії програміста з розробкою комп’ютерних ігор різного рівня складності (для тих, хто був вмотивований і міг встигати). Цьому сприяло те, що в програмі дисципліни «програмування» на другому курсі студенти вивчають об’єктно-орієнтоване програмування мовою C++ і програмний код більшості комп’ютерних ігор створюється саме цією мовою.

При розробці ігрового додатку використовується частина готових бібліотек, тому низькорівнева структура вже сформована. На нижньому рівні знаходяться бібліотеки OpenGL, Windows API, DevIL, OpenAL + Ogg Vorbis і ODE, далі йдуть бібліотеки рушія, класи і допоміжні утиліти, включаючи класи і об’єкти, які є стандартними для кожного проекту. На верхньому рівні створюються додатки з розвиненою 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.

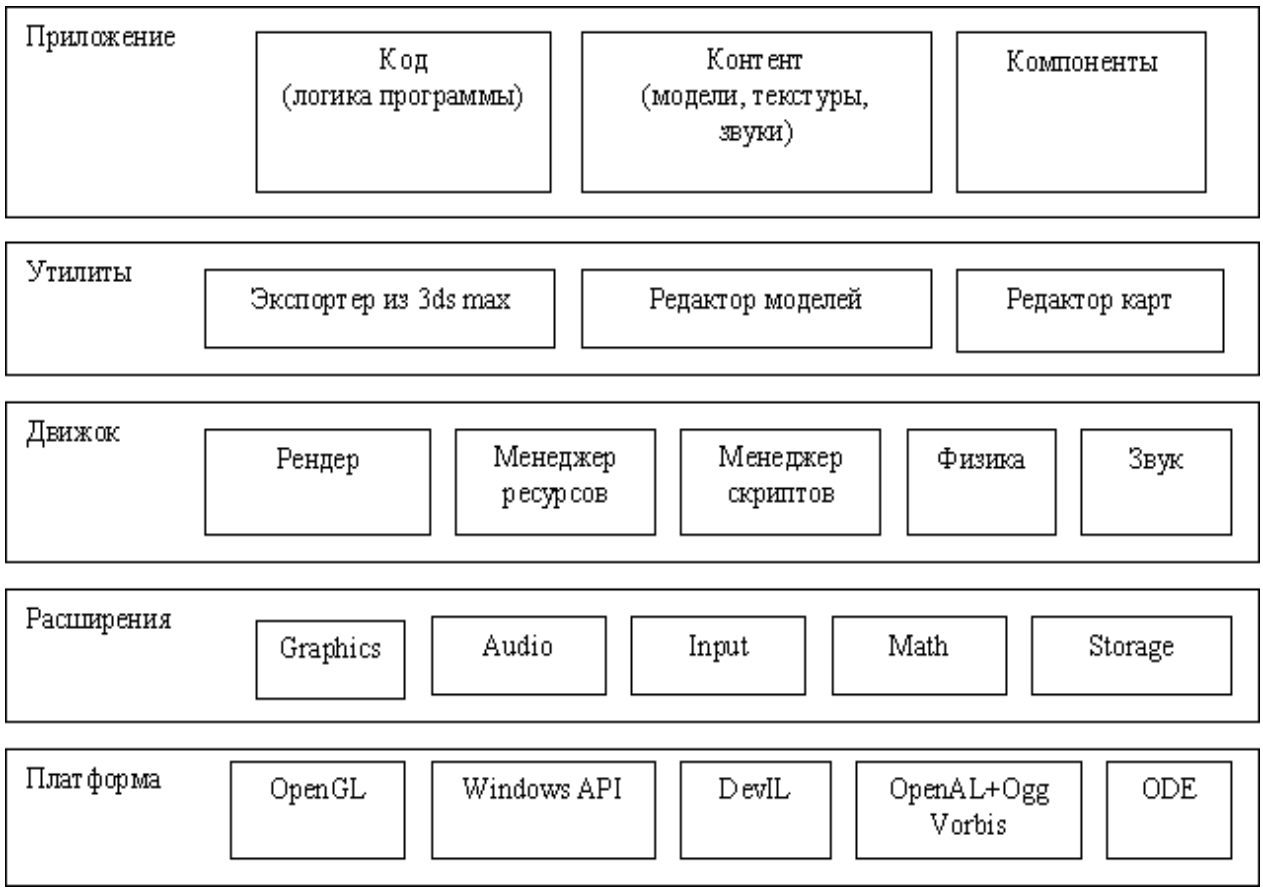

Рисунок 1. Схема створеного програмного проекту

Наступна операція - це ініціалізація пристрою: InitCompatibleContext. 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:

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.

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.

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 недо- статньо зробити просто #inlude і потім написати ім’я бібліотеки з функ- ціями. Крім #include, потрібно ще і виконати команду для підключення функцій. У нашій програмі ми використали бібліотеку GLEW, яка також назива- ється як OpenGL Wrangler Library. Таким чином, підключивши бібліотеку

GLEW командами #include, ми підключили всі необхідні функції OpenGL.

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.

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 wglCreateContext повинна слідувати функція wglMakeCurrent.

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 wglDeleteContext, 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.

ПІДВИЩЕННЯ ПАЛИВНОЇ ЕКОНОМІЧНОСТІ АВТОМОБІЛЯ ЗА РАХУНОК ВДОСКОНАЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ БОРТОВОГО КОМП’ЮТЕРА

Муляренко М. Ю., Магістр
Науковий керівник – ***Біндюг С.А.,*** асистент
Харківський національний автомобільно-дорожній університет

Чип-тюнінг – це коригування програмного забезпечення електронного блоку управління автомобілем з метою зміни режиму його роботи. Чип-тюнінг в основному стосується модифікації програми блоку управління двигуном автомобіля з метою збільшення потужності та оптимізації інших параметрів.

Останнім часом чип-тюнінг все частіше використовується для зменшення споживання палива через зростання цін на нього. Програми блоку управління