УДК УДК 004.413 (045)
Рябоконь Ю.М[1] , Жигаревич О.К.[2], Шолом П.С.[2]
[1]Національний авіаційний університет,
[2]Луцький національний технічний університет

ДОСЛІДЖЕННЯ ІСНУЮЧИХ КОМПОНЕНТІВ РОЗРОБКИ КРОССПЛАТФОРМЕРНИХ ДОДАТКІВ

Рябоконь Ю.М., Жигаревич О.К., Шолом П.С. Дослідження існуючих компонентів розробки інтерактивних кроссплатформерних додатків. У даній статті автори проаналізували результати проектування системи розробки інтерактивних додатків. Був проведений аналіз ринку інтерактивних додатків та існуючих компонентів розробки. З'ясувалось, що чим складніша система, тим більше ресурсів вона використовує.
Ключові слова: веб-застосування, база даних, сервер, клієнт, Silverlight, ASP.NET.

Рябоконь Ю.М., Жигаревич О.К., Шолом П.С. Исследование существующих компонентов разработки интерактивных кроссплатформерних приложений. В данной статье авторы проанализировали результаты проектирования системы разработки интерактивных приложений. Был проведен анализ рынка интерактивных приложений и существующих компонентов разработки. Выяснилось , что чем сложнее система , тем больше ресурсов она использует.
Ключевые слова: веб-приложения, база данных, сервер, клиент, Silverlight, ASP.NET

Ryabokon Y.M., Zhyharevych O.K., Sholom P.S. Investigatin of existing components development of interactive cross platform applications. In this article, the authors analyzed the results of system design development of interactive applications. The market of interactive applications and existing development components were analyzed. It was clarified that the more complex the system, the more resources it uses.
Keywords : web application , database, server, client , Silverlight, ASP.NET

Постановка проблеми. На сьогодні ринок інтерактивних додатків набув великих масштабів та стрімко розвивається і має широку аудиторію користувачів. Тому даний напрямок є перспективним. Для розробки ігор дається мало часу і на створення кожної гри з початку не вистачає ресурсів тоді необхідно використовувати існуючі готові рішення, на основі яких і створюється більшість інтерактивних додатків.

Готові компоненти для розробки інтерактивних додатків значно зменшують бюджет та час на виконання роботи. На даний момент існує велика кількість операційних систем, на яких виконуються ігри, і якщо писати гру окремо для кожної операційної системи це займе дуже багато часу та забере велику кількості ресурсів. Саме для уникнення таких великих витрат і використовуються компоненти для розробки інтерактивних додатків.

Гральний рушій (англ. Game engine) — програмний рушій, центральна програмна частина будь-якої відеогри, яка відповідає за всю її технічну сторону, дозволяє полегшити розробку гри за рахунок уніфікації і систематизації її внутрішньої структури. Важливим значенням рушія є можливість створення багатоплатформенних ігор (сьогодні найчастіше одночасно для ПК, PS3 та Xbox 360).

Основну функціональність гри зазвичай забезпечує рушій гри, що включає рушій візуалізатору, фізичний рушій, звук, систему скриптів, анімацію, ігровий штучний інтелект, мережевий код, керування пам'яттю, граф сцени. Часто на процесі розробки можна заощадити за рахунок повторного використання одного рушія гри для створення декількох різних ігор.

Такі системи, як Infocom'івська Z-Machine і SCI компанії Sierra, можна вважати першими закінченими гральними рушіями. Проте, термін «гральний рушій» з'явився в середині 1990-х років, головним чином, у зв'язку з 3D-іграми, такими як шутери від першої особи. Ігри Doom і Quake від id Software виявилися настільки популярними, що інші розробники замість того, щоб працювати із чистого аркуша, ліцензували основні частини програмного забезпечення й створювали свою власну графіку, персонажів, зброю й рівні — «ігровий контент» або «ігрові ресурси». Рушій Quake був використаний у більш ніж десяти проектах і дав серйозний поштовх розвиткові middleware-індустрії.

Гральні рушії також використовуються в іграх, спочатку розроблених для гральних консолей; наприклад, рушій RenderWare використовується у франчайзах Grand Theft Auto III і Burnout.

Сучасні гральні рушії — одні з найскладніших у написанні застосунків, що найчастіше складаються з десятків різних компонентів, кожний з яких можна налаштовувати окремо під потреби гри. На сайті Future Game Coders є різні теми про підсистеми сучасних ігор.

На додаток до багаторазово використовуваних програмних компонентів, ігральні рушії надають набір візуальних інструментів для розробки. Ці інструменти зазвичай складають інтегроване середовище розробки для спрощеної, швидкої розробки ігор на зразок потокового виробництва. Ці ігральні рушії іноді називають «гральним підпрограмним забезпеченням» (скор. ППЗ; англ. middleware), тому що, з погляду бізнесу, вони надають гнучку й багаторазово використовувану програмну платформу з усією необхідною функціональністю для розробки грального застосунка, скорочуючи витрати, складність і час розробки — усі критичні фактори в сильно конкуруючій індустрії відеоігор.

Як й інші ППЗ рішення, ігральні рушії зазвичай платформо-незалежні й дозволяють деякій грі запускатися на різних платформах, включаючи ігральні консолі й персональні комп'ютери, з деякими внесеними у сирцевий код змінами (або взагалі без них). Часто ігральне ППЗ має компонентну архітектуру, що дозволяє заміняти або розширювати деякі системи рушія більш спеціалізованими (і часто дорожчими) ППЗ компонентами, наприклад, Havok — для фізики, FMOD — для звуку або SpeedTree — для рендеринга. Деякі ігральні рушії, такі як RenderWare, проектують як набір слабо зв'язаних ППЗ компонентів, які можуть вибірково комбінуватися для створення власного рушія, замість традиційнішого підходу розширення або налаштовування гнучкого інтегрованого рішення. Проте розширюваність досягнута й залишається високопріоритетною в гральних рушіях через широкі можливості їхнього застосування. Незважаючи на специфічність назви, ігральні рушії часто використовуються в інших типах інтерактивних застосунків, що вимагають графіку в реальному часі, таких як рекламні демо-ролики, архітектурні візуалізації, що навчальні симулятори й середовища моделювання.

Деякі ігральні рушії надають тільки можливості 3D рендеринга в реальному часі замість усієї функціональності, необхідної іграм. Ці рушії довіряють розроблювачеві гри реалізацію іншої функціональності або її складання на основі інших гральних ППЗ компонентів. Такі типи рушіїв зазвичай відносять до «графічних рушіїв», «рушіїв рендеринга» або «3D рушіям» замість змістовнішого терміна «ігральний рушій». Однак ця термінологія використовується суперечливо: так, багато повнофункціональних гральних 3D рушіїв згадані просто як «3D рушії». Деякі приклади графічних рушіїв: RealmForge, Ogre 3D, Power Render, Crystal Space і Genesis3D. Сучасні ігральні або графічні рушії завичай надають граф сцени — об'єктно-орієнтовані представлення 3D світу гри, що часто спрощує ігровий дизайн і може використовуватися для ефективнішого рендеринга величезних віртуальних світів.

Графічний рушій (англ. graphics engine; іноді «рендерер» або «візуалізатор») — підпрограмне забезпечення (англ. middleware), програмний рушій, основним завданням якого є візуалізація (рендеринг) двовимірної або тривимірної комп'ютерної графіки. Може існувати як окремий продукт або в складі ігрового рушія. Може використовуватися для візуалізації окремих зображень або комп'ютерного відео. Графічні рушії, що використовуються в програмах, що працюють із комп'ютерною графікою (такі, як 3ds MAX, Maya, Cinema 4D, Zbrush, Blender), зазвичай називаються «рендерерами», «рисувальниками» або «візуалізаторами». Сама назва «графічний рушій» використовується, як правило, у відеоіграх.

Основна й найважливіша відмінність «ігрових» графічних рушіїв від програмних рендерерів полягає в тому, що перші повинні обов'язково працювати в режимі реального часу, тоді як другі можуть витрачати по кілька десятків годин на виведення одного зображення. Другою істотною відмінністю є те, що, починаючи приблизно з 1995—1997 років, графічні рушії роблять рендеринг за допомогою графічних процесорів (англ. GPU), які встановлені на окремих платах — відеокартах. Програмні рендерери використовують тільки центральні процесори (англ. CPU).

На етапі становлення відеоігор графічний рушій був найголовнішою частиною ігрового рушія. Властиво, приблизно 90-95 % ігрового рушія складав саме графічний рушій (іншу частину займали такі незначні підсистеми, як «система введення» і деякі інші). Однак із середини 90-х років внаслідок стрімкого розвитку відеоігор розробники ігор почали додавати у свої продукти й інші підсистеми, такі як звуковий рушій, робота з мережею. У сучасних відеоіграх графічний рушій — один із багатьох компонентів ігрового рушія (хоча й найголовніший), куди входять фізичний рушій, звуковий рушій, система анімації (кістякова й лицева анімація), система з роботи з мережею, ігровий штучний інтелект.

Як правило, графічні рушії не поширюються окремо від ігрових. Єдиного графічного рушія без додаткових компонентів й інструментарію недостатньо для створення гри, тому розроблювачі

рушіїв продають лише ігрові рушії з повним набором інструментів і компонентів. Однак це правило не відноситься до вільного програмного забезпечення. Ентузіасти створюють вільні графічні рушії й вільно їх розповсюджують. Згодом розробники ігор можуть об'єднати вільний графічний рушій із фізичним, звуковим та іншими компонентами й створити на їхній основі повноцінний ігровий рушій.

Починаючи з 2009 року, у зв'язку з розвитком графічних процесорів, а саме у зв'язку зі збільшенням їхньої багатофункціональності й гнучкості, почали розроблятися й виходити графічні рушії реального часу, які використовують потужності GPU для розрахунків. Як правило, такі рушії реалізують освітлення через метод трасування променів, а геометрія іноді представлена вокселями, а не полігонами. Дані рушії призначаються для роботи як у відеоіграх, так і в інших інтерактивних і неінтерактивних додатках, включаючи наукові розрахунки, розділення 1600*1200 пікселів, OpenGL-рендерер, 16-кратне повноекранне згладжування (FSAA), 32-бітний колір. У кадрі присутні 23653 трикутники.

На рисунку 1 зображений «Мармуровий чихуахуа» — зображення, створене з використанням вбудованого рендерера Blender.

Рис.1. Графічний мармуровий чихуахуа

OptiX — графічний рушій реального часу, розроблений nVidia, використовує CUDA, і працює винятково на графічних процесорах виробництва nVidia і призначений для різноманітних обчислень, досліджень і моделювань. «OptiX» є гібридним рендерером — основним є використання трасування променів, але є присутнім і растеризація.

Octane Render — графічний рушій реального часу, розроблений компанією Refractive Software LTD, що використовує CUDA і працює на всіх графічних процесорах nVidia, починаючи з 8X00. Використовує трасування променів.

id Tech 6 — графічний рушій, що входить до складу ігрового рушія id Tech 6, буде використовувати трасування променів і вокселі.

Фізичний рушій (англ. physics engine) — програмний рушій, що робить симуляцію фізичних законів реального світу у світі віртуальному з тим або іншим ступенем апроксимації. Найчастіше фізичні рушії використовуються не як окремі самостійні програмні продукти, а як складені компоненти (підпрограми) інших програм. Усі фізичні рушії умовно діляться на два типи: ігрові й наукові. Перший тип використовується в комп'ютерних іграх як компонент ігрового рушія. У цьому випадку він повинен працювати в режимі реального часу, тобто відтворювати фізичні процеси в грі з тою ж швидкістю, з якою вони відбуваються в реальному світі. Разом із тим, від ігрового фізичного рушія не потрібно точності обчислень. Головна вимога — візуальна реалістичність, — і для її досягнення не обов'язково проводити точну симуляцію. Тому в іграх використовуються дуже приблизні апроксимації, наближені моделі й інші програмні «трюки». Наукові фізичні рушії використовуються в науково-дослідних розрахунках і симуляціях, де вкрай важлива саме фізична точність обчислень. Разом із тим швидкість обчислень не грає істотної ролі.

Сучасні фізичні рушії симулюють не всі фізичні закони реального світу, а лише деякі, причому із часом і прогресом у галузі інформаційних технологій і обчислювальної техніки список «підтримуваних» законів збільшується. На початок 2010 року фізичні рушії можуть симулювати такі фізичні явища й стани:

- динаміка абсолютно твердого тіла;
- динаміка деформованого тіла;
- динаміка рідин;
- динаміка газів;
- поведінка тканин;
- поведінка мотузок (тросів, канатів тощо).

У серпні 2009 року англомовний журнал Game Developer (англ.), присвячений розробці комп'ютерних ігор, опублікував статтю про сучасні ігрові рушії та їхнє використання. Згідно з даними журналу, найпопулярнішим серед розробників є рушій nVidia PhysX, що займає 26,8% ринку. На другому місці перебуває Havok, що займає 22,7% ринку. Третє місце належить рушію Bullet Physics Library (10,3%), а четверте — Open Dynamics Engine (4,1%).

Фізичний рушій дозволяє створити деякий віртуальний простір, який можна наповнити тілами (віртуальними статичними й динамічними об'єктами), і вказати для нього якісь загальні закони взаємодії тіл і середовища, тією, чи іншою мірою наближені до фізичних, задаючи при цьому характер і ступінь взаємодій (імпульси, сили тощо). Властиво розрахунок взаємодії тіл рушій і бере на себе. Коли простого набору об'єктів, що взаємодіють за певними законами у віртуальному просторі, недостатньо в силу неповного наближення фізичної моделі до реального світу, можливо додавати (до тіл) зв'язки. Розраховуючи взаємодію тіл між собою й із середовищем, фізичний рушій наближає фізичну модель одержуваної системи до реального світу, передаючи уточнені геометричні дані засобові відображення (рендереру).

Ігровий штучний інтелект (англ. Game artificial intelligence) — набір програмних методик, які використовуються у відеоіграх для створення ілюзії інтелекту в поведінці персонажів, керованих комп'ютером. Ігровий ШІ, крім методів традиційного штучного інтелекту, включає також алгоритми теорії керування, робототехніки, комп'ютерної графіки та інформатики у цілому.

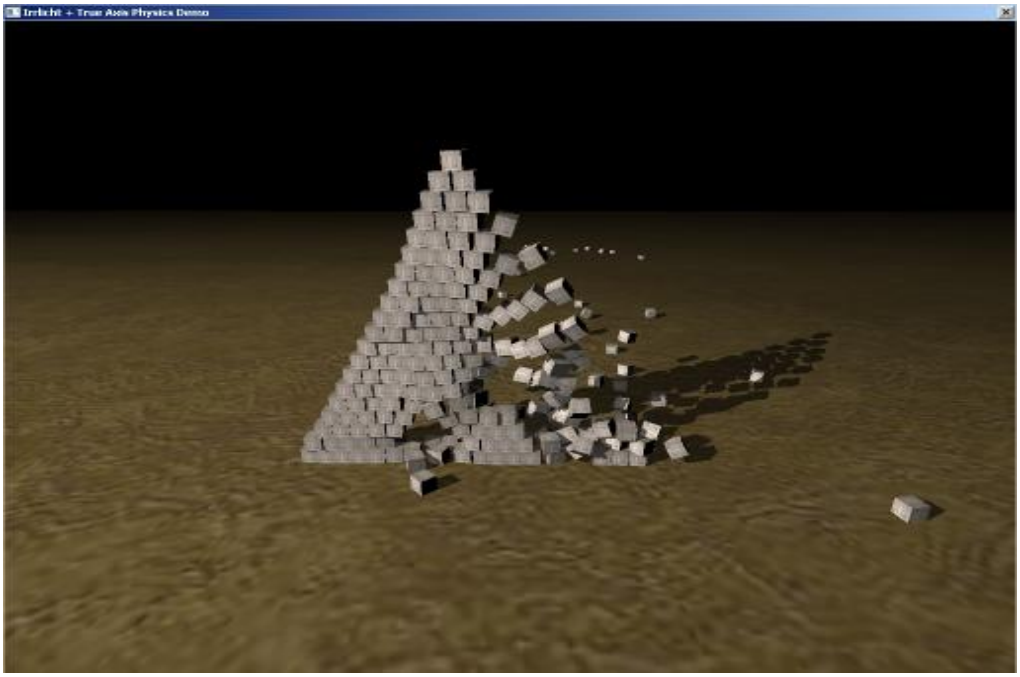

Рис. 2. Приклад роботи фізичного рушія

Реалізація ШІ сильно впливає на геймплей, системні вимоги і бюджет гри, і розробники балансують між цими вимогами, намагаючись зробити цікавий і невимогливий до ресурсів ШІ малою ціною. Тому підхід до ігрового ШІ серйозно відрізняється від підходу до традиційного ШІ — широко застосовуються різного роду спрощення, обмани й емуляції. Наприклад: з одного боку, в шутерах від першої особи безпомилковий рух і миттєве прицілювання, властиве ботам, не

залишає жодного шансу людині, так що ці властивості штучно знижуються. З іншого боку — боти повинні робити засідки, діяти командою й т.д., для цього застосовуються «костилі» у вигляді контрольних точок, розставлених на рівні.

Деякі ігрові програмісти розглядають будь-яку методику, що використовується для створення ілюзії інтелекту, як частину ігрового ШІ. Однак цей погляд є спірним, тому що він включає методики, які широко використовуються поза рушієм ігрового ШІ. Наприклад, інформація про потенційні майбутні зіткнення є важливою інформацією, що вводиться в алгоритми, які допомагають створювати ботів, що будуть досить розумними для уникнення зіткнень з об'єктами. Але ті ж самі методики виявлення зіткнень є необхідним і одним із найважливіших компонентів фізичного рушія. Точно так само результати іспитового напрямку погляду бота звичайно є важливими даними, що вводяться в систему прицілювання бота; разом із тим ці дані широко використовуються при рендерингу в графічному рушії. Фінальним прикладом є скриптинг, що може бути зручним інструментом для всіх аспектів ігрової розробки, однак часто сильно асоціюється з контролюванням поведінки неігрових персонажів.

Пуристи вважають, що вираз «штучний інтелект» у терміні «ігровий штучний інтелект» є перебільшенням, оскільки ігровий ШІ описує не інтелект і використовує тільки деякі з напрямків академічної науки «Штучний інтелект». Беручи до уваги, що «реальний» ШІ звертається до галузей систем, що самонавчаються, і прийняття рішень, які базуються на довільному уведенні даних, і навіть до остаточної мети «сильного» ШІ, що може мислити, ігровий ШІ часто складається з декількох емпіричних правил і евристики, яких досить, щоб надати гравцеві гарний геймплей, відчуття й враження від гри.

Покращення розуміння академічного ШІ розробниками ігор і зростаючий інтерес академічного співтовариства до комп'ютерних ігор викликає питання, наскільки й у якій мірі ігровий ШІ відрізняється від класичного. Однак, істотні розходження між різними прикладними галузями штучного інтелекту означають, що ігровий ШІ все ще може бути розглянутий як окрема підгалузь ШІ. Зокрема, здатність «законним» чином вирішити деякі проблеми ШІ в іграх через обман утворює важливе розходження. Наприклад, виведення позиції невидимого об'єкта з минулих спостережень може бути важкою проблемою, коли ШІ застосований до робототехніки, але в комп'ютерних іграх неігровий персонаж може просто шукати позицію в ігровому графі. Такий обман може призвести до нереалістичної поведінки й тому не завжди бажаний. Але його здатність служити для розрізнення ігрового ШІ веде до нових проблем, таких як: коли і як використовувати обман.

Евристичні алгоритми ігрового штучного інтелекту використовуються в широкій розмаїтості в багатьох галузях усередині гри. Найочевидніше застосування ігрового ШІ проявляється в контролюванні неігрових персонажів, хоча скриптинг теж є дуже розповсюдженим способом контролю. Пошук шляху є іншим широко розповсюдженим застосуванням ігрового ШІ, — він особливо проявляється в стратегіях реального часу. Пошук шляху є методом для визначення того, як неігровому персонажеві перейти з однієї точки на мапі до іншої: потрібно враховувати ландшафт, перешкоди й, можливо, «туман війни». Ігровий ШІ також пов'язаний із динамічним ігровим балансуванням.

Концепція непередбачуваного (англ. emergent) ШІ була недавно досліджена в таких іграх як Creatures, Black & White і Nintendogs і в таких іграшках, як тамагочі. «Свійські тварини» у цих іграх мають здатність «навчатися» на діях, вчинених гравцем, і їхня поведінка змінюється відповідно. У той час, як ці рішення взяті з обмеженої множини можливих рішень, це дійсно часто дає бажану ілюзію інтелекту по іншу сторону екрана.

При розробці ігор звуковий супровід завжди перебувало кілька на другому плані. Розробники вважають за краще витратити час на введення новомодних ефектів для 3D -графіки , тоді як реалізація звуку пускається на самоплив. Людей складно переконати витратити час і кошти на якісний звук в грі. Разом з тим, більшість користувачів також більш охоче витратять гроші на новітній 3D-акселератор, ніж на нову звукову карту.

Однак, ситуація змінюється: останнім часом звуку стало приділятися набагато більше уваги і з боку користувачів, і з боку розробників. У сучасних проектах звуку відводиться до 40 відсотків бюджету і тимчасово - людських ресурсів.

Виробники звукових чипів і розробники технологій 3D - звуку також доклали чимало зусиль , щоб переконати користувачів і розробників додатків в тому, що хороший 3D - звук є невід'ємною частиною сучасного комп'ютера.

Звук з стерео перетворився на тривимірний, потім з'явилися і багатоканальні рішення: 4 - канальні, 5.1 - звук, а останнім часом і 7.1.

Саме поняття «тривимірний звук» має на увазі, що джерела звуку розташовуються в тривимірному просторі навколо слухача. При цьому кожен джерело являє собою в широкому сенсі будь-який об'єкт у віртуальному ігровому світі, здатний виробляти звуки.

Джерела звуку (Sound Sources). Причому, деякі з джерел звуку - звичайні стерео, такі як фонова музика (в даній конкретній грі це main ambient: вітер або трек «звуки джунглів»), 8 джерел створюються ігровими персонажами - монстрами, 1 джерело створює безпосередньо гравець - постріл, звуки кроків і 3 випадкових джерела - для посилення ефекту простору (ambient sounds) - в даному випадку «джунглі»: звуки комах, птахів і т.п.

3D - звук потрібен для більш глибокого занурення користувача в віртуальний світ гри, за рахунок посилення реалізму відбувається на ігровій сцені. Для цього використовуються різні технології, адаптовані або гіперболізує поведінку звуку в реальному світі. Наприклад, реверберації, відбиті звуки, оклюзії (звук, що пройшов через перешкоду), обструкції (звук не пройшов через перешкоду), дистанційне моделювання (вводиться параметр віддаленості джерела звуку від слухача) і маса інших ефектів.

Звичайно ж, сприйняття звуку строго індивідуально у кожної людини (воно залежить від форми вуха, віку та психологічного настрою в конкретний момент). Таким чином, ніколи не буде однозначної думки про звучання тієї чи іншої звукової карти або ефективності тієї чи іншої технології 3D-звуку. Відтворення ж звуку сильно залежить від звукової карти і колонок, а також від конкретної реалізації звукового движка в грі.

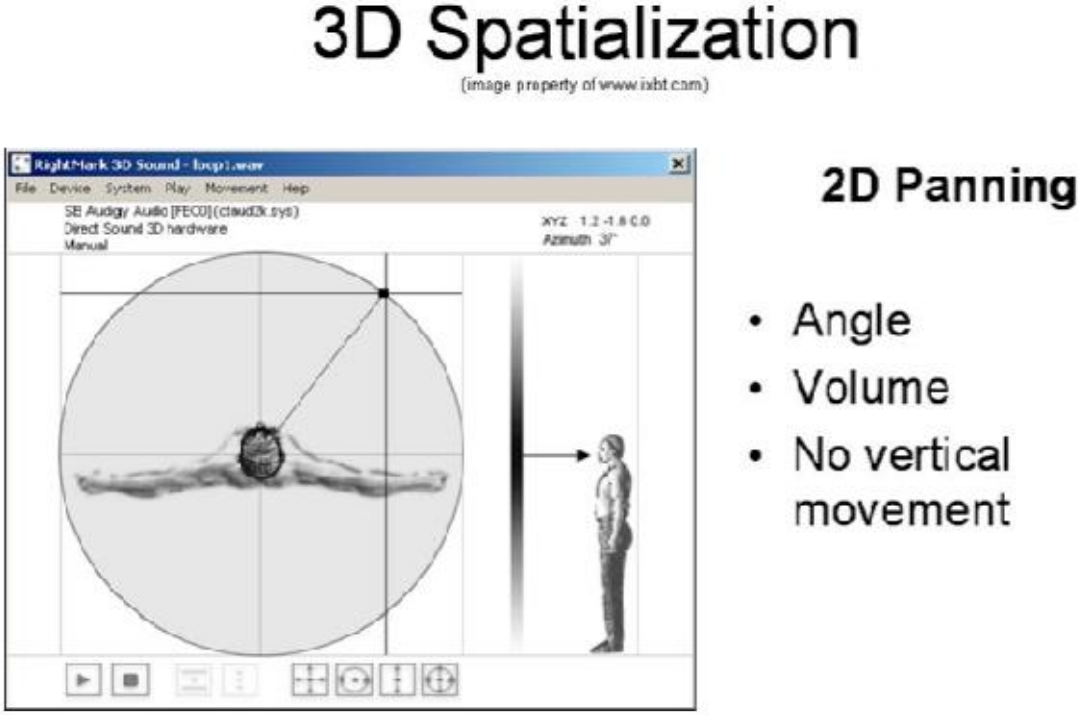

Рис.3. Представлення сприйняття звуку людиною

Розглянемо, як же створюється ефект 3D - звуку. На рисунку 3 представлено сприйняття звуку людиною. Почнемо з найпростішої технології 2D - панорамування. Ця технологія застосовувалася ще в DOOM від відомих id Software - все просто: кожне джерело звуку грається як стерео, а позиціонування створюється за рахунок зміни гучності лівого або правого каналів. У такій системі немає вертикального позиціонування, але можлива реалізація ефектів, пов'язаних, скажімо, з невеликою зміною (фільтрацією високих частот) звуку, коли він знаходиться ззаду слухача, так як в реальності ми чуємо злегка заглушений звук, якщо він знаходиться за головою слухача.

Тепер ми підійшли до апаратної реалізації. На двох колонках або в навушниках звукова карта відтворює положення джерела звуку за допомогою HRTF (Head Related Transfer Function). Фільтрація та інші перетворення відтворюють поведінку слухової системи людини.

На рисунку 3 зображені три HRTF (позиція джерела звуку азимут 135 градусів і 36 градусів) трьох різних людей для лівого і правого вуха відповідно. Можна помітити певні закономірності на всіх трьох функціях. Звідки ж вони беруться? У більшості своїй вони записуються за допомогою спеціальних методик і спеціальних стереомікрофоном, які вставляються у вуха людини або спеціального манекена (KEMAR). Фірма Sensaura, наприклад, використовує синтетичні HRTF, використовуючи ті самі закономірності, які ми можемо спостерігати на ілюстрації. Наприклад, пік в районі 2500 Гц і спад близько 5000 Гц для даної точки в просторі. Інші фірми використовують усереднені HRTF .

По суті, вся система - це два FIR - фільтра (Finite Impulse Response), передавальна характеристика яких і є HRTF. Так як HRTF дискретні, і зберігати мегабайти HRTF для часток градуса важко - реальне положення джерела прораховується інтерполяцією HRTF.

OGRE (англ. Object-Oriented Graphics Rendering Engine, об'єктно орієнтований графічний рушій) — гнучкий, орієнтований на сцену та кросплатформений графічний рушій (на відміну від рушія гри) написаний на C++ та спроектований так, щоб зробити простішим та інтуїтивним процес розробки програм, що використовують тривимірну графіку. Поширюється на правах MIT ліцензії. Бібліотека класів спроектована таким чином, що її можна однаково використовувати з OpenGL та Direct3D не змінюючи програмного коду прикладної програми.

Серед комерційних ігор, які використовують OGRE можна відмітити: Ankh, Torchlight та Garshasp.

OGRE сам по собі не є ігровим рушієм і за заявою автора ніколи таким не буде. OGRE був, є і буде графічним рушієм для рендеринга тривимірної графіки. Велику популярність рушій отримав за рахунок своєї гнучкості, що дозволяє «схрещувати» його з багатьма іншими бібліотеками (фізика — ODE, Newton, PhysX, Bullet; звук, мережа, графічний інтерфейс тощо). На рисунку 2.7 представлений приклад роботи OGRE.

Для реалізації графічного інтерфейсу користувача (англ. GUI — Graphic User Interface) можуть застосовуватися як стандартні (недостатньо добре реалізовані, і, за словами розробників, в майбутньому, можливо, будуть виключені) функції графічного інтерфейсу OGRE, так і імпортуватися сторонні бібліотеки (OpenGUI, MyGUI (дуже популярний серед учасників російського співтовариства OGRE), CEGUI).

OGRE є вільним програмним забезпеченням, поширюваним під ліцензіями LGPL/MIT і має дуже активне співтовариство.

Можливості:

- підтримка платформ Windows, Linux та Mac OS X;
- скриптова система управління матеріалами (мультитекстурування, мультипрохідне змішування);
- завантаження текстур у форматі PNG, JPEG, TGA, BMP або DDS, підтримка стислих текстур (DXT/S3TC);
- експортери для основних комерційних та вільних пакетів 3D моделювання;
- система управління ресурсами;
- підтримка DirectX, OpenGL;
- підтримка шейдерів, написаних на асемблері або мовах високого рівня: Cg, DirectX HLSL або GLSL;
- складна скелетна анімація (анімація тіла), анімація гнучких форм, морфінг (анімація особи), анімація шляху (камера, переміщення).

Irrlicht (Irrlicht Engine) - тривимірний графічний движок, який є безкоштовним вільним програмним продуктом і поширюється на умовах ліцензії zlib.

Irrlicht використовує можливості OpenGL, DirectX і декількох власних рендерерів. Користувачеві надаються різні функціональні можливості по завантаженню та управління тривимірними (3D) об'єктами (сцени, моделі тощо), небагатьма спецефектами і графічним інтерфейсом користувача. Рекомендується для ознайомлення з процесом розробки і створення нескладних ігор і демосцен (Irrlicht підтримує формати популярних ігор і движків , зокрема моделі quake 2, quake 3, карти рівнів та ін). Не вимагає підключення сторонніх модулів для

реалізації високорівневих функцій (є найпростіша фізика, GUI (графічний інтерфейс користувача) і т. д.). Існує три офіційні доповнення для Irrlicht: IrrKlang (аудіобібліотеку), IrrXML (завантаження і обробка XML - файлів), IrrEdit (редактор сцен). Для використання розширених функцій фізики існує фізичний движок ChronoEngine (з причини того, що в Irrlicht вбудована примітивна фізична система).

Одна з важливих особливостей Irrlicht його кроссплатформенність - тобто здатність працювати на різних платформах. Платформо - незалежний прошарок забезпечує легке перенесення на різні, які не підтримує офіційно платформи, зокрема існують порти під android, iPhone і інші.

Marmalade SDK - кроссплатформне SDK від Ideaworks3D Limited. Являє собою набір бібліотек, зразків, інструментів і документацій необхідних для розробки , тестування та розгортання додатків для мобільних пристроїв. Основоположною концепцією Marmalade SDK є одноразове написання програми і компілювання її на всі підтримувані платформи, без необхідності програмування на різних мовах програмування і використання різних API для кожної платформи.

Для того щоб використовувати Marmalade SDK необхідно придбати ліцензію. Є чотири види ліцензій, які надають доступ до різних наборів платформ, розгортання та рівня технічної підтримки. Ліцензія потрібна для кожного комп'ютера де встановлений Marmalade SDK.

Основа Marmalade SDK складається з двох основних шарів. Низькорівневий з API називається Marmalade System забезпечує рівень абстракцій, що дозволяє отримати програмісту доступ до функціональності пристрою, таких як управління пам'яттю, доступ до файлів, мережі, даними введення (таким як , акселерометр, клавіатура, сенсорний екран), звуку.
Marmalade Studio C ++ API , який забезпечує високорівневу функціональність, в основному спрямований на підтримку 2D (наприклад, обробка растрового зображення і шрифтів) і 3D-рендрингу графіки.

Nebula Device - ігровий движок, розроблений німецькою компанією Radon Labs і вперше використаний в комп'ютерній грі 2002 Project Nomads. Nebula Device вільним програмним продуктом і поширюється на умови ліцензії MIT. При розробці Nebula Device основний напрямок робився на оптимальну роботу з великими відкритими просторами, ефекти візуалізації неба і велику дистанцію промальовування. Зараз Nebula Device не тільки використовується в усіх іграх компанії Radon Labs, але і в численних сторонніх розробках.

Движок написаний на мові програмування C++ і підтримує кілька скриптових мов, таких як Tcl, Lua, Python, Ruby, Java і . NET Framework. Є можливість підключити і другі скриптові мови, за допомогою під'єднуваних плагінів. Рендеринг движка функціонує у двох режимах (DirectX і OpenGL), завдяки чому забезпечується кроссплатформенність .

Підтримуються операційні системи Linux, Mac OSX, IRIX і Microsoft Windows а також ігрова приставка Xbox. Для текстур підтримуються графічні формати DDS, BMP, JPEG, GIF, TIFF, PNG і деякі інші. Відкритість графічних форматів дає деяку творчу свободу ентузіастам , які розробляють модифікацій для ігор на движку Nebula Device.

Підтримувані формати тривимірних моделей - NVX, N3D і OBJ. Nebula Device дозволяє також використовувати шейдерні ефекти, скелетну анімацію, системи частинок, динамічні тіні та пост- ефекти. До складу SDK входять також додаткові утиліти, такі як програма для контролювання джерел світла «Light Control Tool»; архів з вихідним кодом також можна завантажити окремо. До вільної завантаження доступні три покоління ігрового движка.

Висновки. У статті було здійснено дослідження системи компонентів розробки інтерактивних додатків, яка включає в себе велику кількість зв'язаних модулів. Модулі графічного відображення, фізичного моделювання, моделювання звуку, штучний інтелект.

Була розглянута достатня кількість існуючих альтернативних компонентів, кожний з яких має свої переваги та недоліки. Система, що розробляється буде брати основні принципи архітектури з даних компонентів та додавати свої рішення, які будуть сприяти зменшенню входження для використання компонентів.

1. Бабенко Л.П., Лавріщева К.М. Основи програмної інженерії: Навч. посіб. – К.:Т-во «Знання», 2001. – 269 с.

2. М. Мак-Дональд, М.Шпушта Microsoft ASP.NET 3.5 с примерами на C# 2008 и Silverlight 2 для профессионалов = Pro ASP.NET 3.5 in C# 2008: Includes Silverlight 2. — 3-е издание. — М.: «Вильямс», 2009. — С. 1408.
3. Мэтью Мак-Дональд. Silverlight 3 с примерами на C# для профессионалов = Pro Silverlight 3 in C#. — 3-е изд. — М.: Вильямс, 2010. — 656 с.
4. Мэтью Мак-Дональд Microsoft ASP.NET 2.0 с примерами на C# 2005 для профессионалов.: Пер. с англ. – М.:ООО «И.Д. Вильямс», 2006. –1408с.
5. Роб Камерон, Дэйл Михалк ASP.NET 3.5, компоненты AJAX и серверные элементы управления для профессионалов = Pro ASP.NET 3.5 Server Controls with AJAX Components. — М.: «Вильямс», 2009. — С. 608.
6. Дейт К. Дж. Введение в системы баз данных. — 8-е изд. — М.: Вильямс, 2006. — 1328 с.