СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки
Кафедра програмування та математики

ПОЯСНЮВАЛЬНА ЗАПИСКА
до кваліфікаційної випускної роботи

освітній ступінь бакалавр
спеціальність 121 „Інженерія програмного забезпечення”
(шифр і назва спеціальності)
спеціалізація „Інженерія програмного забезпечення”

на тему „ Програмування додатку, реалізованого в комп’ютерній грі ”

Виконав: студент групи ІПЗ-16д _____ _В.С. Бухаров_____
(підпис) (ініціали і прізвище)

Керівник _____ _В.Г. Іванов_____
(підпис) (ініціали і прізвище)

Завідувач кафедри _____ _В.О. Лифар_____
(підпис) (ініціали і прізвище)

Рецензент_____

Сєвєродонецьк – 2020

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки
Кафедра програмування та математики

Освітній ступінь бакалавр
спеціальність 121 „Інженерія програмного забезпечення”
(шифр і назва спеціальності)
спеціалізація „Інженерія програмного забезпечення”
(назва спеціалізації)

ЗАТВЕРДЖУЮ
Завідувач кафедри ПМ,
д.т.н., доцент
_____ Лифар В.О.
“____” _____2020 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ ВИПУСКНУ РОБОТУ СТУДЕНТУ

_____Бухаров Володимир Сергійович _____
(прізвище, ім’я, по батькові)

1.Тема роботи Програмування додатку, реалізованого в комп’ютерній грі _____
_____

Керівник роботи___доцент, к.т.н. Іванов Віталій Геннадійович_____,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджений наказом університету від “___”_____20__року №___

2. Строк подання студентом роботи_20 травня 2020 р._____ _____
3. Вихідні дані до роботи___Об'єктом даної роботи є процес розробки ігрового додатку._____
_____
_____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Вступ. Аналітичний огляд, Аналіз популярних представників, розбиття засобів на класи. Постановка задач для тесових проектів ігор-прототипів, у розрізі спрощених вимоги до сучасних ігрових проектів. Висновки. Перелік використаних
джерел._____
_____
_____

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслеників)
_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання__30 березеня 2020 року._____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання кваліфікаційної випускної роботи	Строк виконання етапів	Примітка
1	Одержання завдання на виконання роботи	30.03.20	
2	Укладання і погодження з керівником плану і етапів виконання роботи	06.04.20	
3	Узагальнення даних літературних джерел, укладання розділу «Аналіз предметної галузі»	13.04.20	
4	Аналіз шляхів виконання завдання. Вибір і погодження керівником оптимального шляху	20.04.20	
5	Укладання та тестування програмного продукту	27.04.20	
6	Укладання, оформлення та погодження пояснювальної записки з керівником	04.05.20	
7	Здача готової пояснювальної записки на кафедру	12.05.20	
8	Укладання доповіді і презентації	30.05.20	

Студент _____ В.С. Бухаров_____
(підпис) (ініціали і прізвище)

Керівник роботи _____ _____В.Г. Іванов_____
(підпис) (ініціали і прізвище)

ЛИСТ ПОГОДЖЕННЯ І ОЦІНЮВАННЯ

дипломної роботи студента гр. ІПЗ-16д Бухаров В.С.

Науковий керівник

Доцент, к.т.н. _____ Іванов В.Г.

Оцінка наукового керівника: _____

Рецензент _____

ПІБ, місто роботи, посада

Оцінка рецензента: _____

Кінцева оцінка за результатами захисту:

_____

Голова ЕК

_____ Лифар В.О.

підпис

РЕФЕРАТ

Робота містить: 50 сторінок основного тексту, 22 сторінок додатків, 6 рисунка, 1 таблицю, 10 використаних джерел.

Метою випускної кваліфікаційної роботи є вивчення особливостей є збір теоретичних відомостей про сучасні інструментальні засоби для побудови ігор. Огляд та аналіз популярних представників, розбиття засобів на класи. Постановка задач для тесових проектів ігор-прототипів, у розрізі спрощених вимоги до сучасних ігрових проектів. Детальний аналіз найбільш характерних представників кожного класу, розробка з їх допомогою ігор прототипів. Визначення оптимальних сфер використання для інструментальних засобів спираючись на аналіз процесу розробки прототипі.

В результаті виконаної роботи, було реалізовано ігрової додаток.

Ігрової додаток реалізован відповідно всім вимогам технічного завдання.

ЗМІСТ

ВСТУП

Актуальність досліджень: В наш час не можливо уявити повсякденне життя без постійного притоку інформації. Для задоволення цих потреб використовуються сучасні засоби мультимедіа : телебачення, Інтернет, тощо. Мультимедіа — комбінування різних форм представлення інформаційна одному носієві. Мультимедіа може бути грубо класифікована як лінійна й нелінійна. Аналогом лінійного способу подання може бути кіно. Людина, що переглядає даний документ жодним чином не може вплинути на його зміст. Нелінійний спосіб подання інформації дозволяє людині брати участь у поданній інформації, взаємодіючи якимось чином із засобом відображення мультимедійних даних. Участь людини в даному процесі також називається «інтерактивністю». Такий спосіб взаємодії людини й комп'ютера найбільш повно представлений у категоріях комп'ютерних відеоігор. Відеогра — це електронна гра, в ігровому процесі якої гравець використовує інтерфейс користувача, щоб отримати зворотну інформацію з відео-пристрою. Електронні пристрої, які використовуються для того щоб грати, називаються ігровими платформами.

Наприклад, до таких платформ належать персональний комп'ютер та гральна консоль. Пристрій введення, який використовується для керування грою, називається ігровим контролером. Це може бути, наприклад,джойстик, клавіатура та мишка, геймпад або сенсорний екран. Комп’ютерні відеоігри ігри набули небувалої популярності за останні роки та посіли почесне місце на ринку розваг та дозвілля. Як результат, бурний розвиток індустрії розробки комп‘ютерних ігор. Для задоволення потреб розробників було створено величезну кількості інструментальних засобів та продовжується розробка нових. Загальна ціль всіх інструментальних засобів - полегшення та покращення розробки, використання передових технологій обробки графіки, фізики, забезпечення кросплатформенності розроблених проектів.

Метою даної роботи є збір теоретичних відомостей про сучасні інструментальні засоби для побудови ігор. Огляд та аналіз популярних представників, розбиття засобів на класи. Постановка задач для тесових проектів ігор-прототипів, у розрізі спрощених вимоги до сучасних ігрових проектів. Детальний аналіз найбільш характерних представників кожного класу, розробка з їх допомогою ігор прототипів. Визначення оптимальних сфер використання для інструментальних засобів спираючись на аналіз процесу розробки прототипів.

Об'єкт досліджень: Створення комп'ютерного додатку-гри.

Предмет досліджень: Технології створення додатків, який був би легкий у використанні будь-якому користувачу.

Мета дослідження: Виявити найбільш зручне середовище для створення комп'ютерного додатку-гри.

Завдання дослідження:

а) дослідити та скласти інформаційну модель;

б) детально розкрити технології створення комп'ютерної гри;

в) реалізувати додаток.

Методологічна та теоретична основа дослідження: можливі методи створення комп'ютерного додатку-гри.

Методи дослідження: технології побудови створення комп'ютерного додатку-гри.

Практичне значення отриманих результатів: Одержаний результат буде використовуватися для розваг у сфері ігор.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Мультимедійні ігри

В сфері мультимедіа не останнє місце займає таке явище, як мультимедійні комп'ютерні ігри. Мультимедійні ігри — такі ігри, у яких гравець взаємодіє з віртуальним середовищем, побудованим комп'ютером. Стан віртуального середовища передається гравцеві за допомогою різних способів передачі інформації (аудіальний, візуальний, тактильний). Наразі всі комп'ютерні ігри відносяться до мультімедийних ігор. В такий тип ігор можна грати як в поодинці на локальному комп'ютері або приставці, так і з іншими гравцями через локальну або глобальну мережу. У 2011 році відеоігри були офіційно визнані видом мистецтва урядом США та Національним фондом мистецтв США [13].

1.2 Ігрова індустрія

Перша комп‘ютерна гра «Зоряні війни» вийшла у світ 1962 року. Її завдання полягало в тому, щоб відбити астероїди і напади ворожих космічних кораблів. Згодом було створено багато інших ігор. А з поширенням у 1970–1980 роках потужніших комп‘ютерів електронних ігор побільшало: пригодницькі ігри, ігри-головоломки, стратегічні ігри та ігри «екшн».

Багато ігор імітують різні види спорту, як от хокей на льоду чи гольф. Чимало з них здобули високу оцінку громадськості,оскільки вони дуже цікаві й допомагають у навчанні. Стає дедалі популярнішим онлайн вид комп‘ютерної гри. Її персонажами керує не комп‘ютер, а гравці, які через Інтернет одночасно беруть участь у грі. Їх можуть бути тисячі.

Популярність таких забав пояснюється можливістю поспілкуватися з іншими.

Гравці — розмовляють одні з одними і відчувають себе частиною всесвітньої родини. В Україні щороку зростає кількість людей, що купують комп‘ютерні ігри. Якщо для гравців це просто забавка, то для розробників, виробників та розповсюджувачів — досить вигідний бізнес. Світовий ринок комп‘ютерних ігор оцінюють в сотні мільярдів доларів. На Заході один ліцензійний ігровий диск коштує $40–50. Популярну гру можуть продати накладом від мільйона до кількох десятків мільйонів примірників. Не дивно, що в розробку гри там можуть легко вкласти кілька мільйонів доларів. Ця індустрія приносить великі прибутки і державній скарбниці. Комп‘ютерні ігри стали вже й елементом політики. Парламенти західних країн дискутують щодо законодавчого обмеження насильства в комп‘ютерних іграх або ж стимулювання виробництва ігор як такого.

1.3 Класифікація комп’ютерних ігор

Комп'ютерні ігри в основному класифікуються за жанрами, а також за кількістю гравців. Внаслідок того, що критерії приналежності гри до того чи іншого жанру не визначені однозначно, класифікація ігор недостатньо систематизована, і в різних джерелах дані про жанр конкретного проекту можуть розрізнятися.

Проте, існує консенсус, до якого прийшли розробники ігор, і приналежність гри до одного з основних жанрів майже завжди можна визначити однозначно. Ці найбільш популярні жанри (які об'єднують в собі безліч піджанрів) перераховані нижче. Існують ігри з елементами кількох жанрів, які можуть належати кожному з них (наприклад, серія Grand Theft Auto, Космічні Рейнджери, Rome: Total War і багато інших). Такі

проекти зараховують або до одного з жанрів, який в грі є основним, або відразу до всіх, присутніх в грі, якщо вони в рівній мірі становлять геймплей проекту.

В основі сучасних розподілів відеоігор на жанри лежить вид активності, який найчастіше здійснює гравець в іграх даного жанру. Так відеоігри в загальному можуть поділятися на ігри руху, планування і сюжету або спілкування, дії та контролю. В багатьох класифікаціях визначення жанру відбувається за кількома осями. Наприклад, за двома осями сюжет — свобода дії,або трьома абстракція — симуляція свобода[16]. Проте найчастіше використовуваною класифікацією, хоч і не прийнятою усіма, жанри з якої зустрічаються в більшості існуючих, є наведена нижче, яка виключає осі або багаторівневі поділи:

Action

В іграх такого жанру необхідно використовувати рефлекси та швидкість реакції для подолання ігрових обставин. Це один із базових жанрів і водночас найпоширеніший. Як правило екшн-ігри пов'язані із агресивними діями щодо противників і/або оточення. Персонаж гравця повинен битися, стріляти, переслідувати ціль чи самому уникати переслідування. Стосовно екшн-ігор, де наявні значні елементи пригодницьких ігор, застосовується термін Action-adventure. З-поміж них часто виділяється напрям аркадних ігор, ігровий процес яких вирізняється простотою та легкістю освоєння.

Цей жанр поділяється на велику кількість під жанрів, серед яких основними є: Шутери—вимагають від гравця боротися з противниками шляхом стрілянини. Залежно від перспективи, поділяються на шутери від першої (Wolfenstein 3D) чи третьої особи(Макс Пейн). Існують різновиди як тактичні, в яких ігровий персонаж діє у складі команди (SWAT 4), аркади(Alien Shooter), стелс-екшн, метою якого є приховані дії для виконання

завдань, без прямого знищення противників (серія Hitman). Щодо ігор, де основою ігрового процесу є знищення великих кількостей ворогів, а сама стрілянина в цьому переважає над тактикою і влучністю, застосовується термін Shoot 'em up (R-Type, Touhou, Contra).

Файтинги — імітують ближній бій, як правило один на один, на спеціальних аренах. Приклади:серія Mortal Kombat,Street Fighter. Beat 'em up — подібні на файтинги, з тою різницею, що персонажі вільно переміщуються ігровим світом (а не спеціальною ареною) і борються проти багатьох противників одночасно. Приклади: Golden Axe, Battletoads, Double Dragon.

Платформери — персонаж мусить рухатися, стрибаючи по платформах, та долати перешкоди. Приклади:Mario,Spyro theDragon,Megaman. Лабіринти персонаж рухається лабіринтом з метою знайти вихід, зібрати предмети і/або уникнути пасток і небезпек. Часто в іграх цього жанру є обмеження на час. Приклади: Pac-Man, Boulder Dash.

Стратегія

Гравець керує не одним персонажем, а цілим підрозділом, підприємством чи навіть всесвітом. Відповідно до реалізації ігрового часу, стратегічні відеоігри поділяються на два основних різновиди:Покрокові стратегічні ігри — гравець та його противник здійснюють дії один за одним, маючи змогу за один ігровий хід виконати певну кількість операцій. Приклади: Heroes of Might and Magic III,Цивілізація. Стратегічні ігри в реальному часу — і гравець і противник виконують свої дії одночасно, проте часто масштаб часу відрізняється від реального. Наприклад, будівництво триває кілька секунд, а ігрова година складає кілька хвилин реального часу.

Приклади: StarCraft, Age of Empires III. Tower Defense — похідний жанр, в якому гравець керує оборонними баштами аби не пропустити хвилі ворогів.

MOBA — похідний жанр, в якому гравець керує одним персонажем з метою захистити свою базу від ворогів та знищити ворожу. MMORTS — багатокористувацькі он-лайн стратегії в реальному часі, орієнтовані на суперництво/співпрацю з іншими реальними гравцями, щоб досягнути спільної мети. Видом стратегічних ігор, які відображають суто бойові дії, є варгейми. Ігри, в яких гравець управляє масштабними державами (імперіями, планетами, галактиками), при цьому займаючись всіма аспектами їх життя, включаючи торгівлю, науки та війни, мають назву глобальних стратегій. Близьким до них є ігри в бога, де гравець виступає в ролі надприродної істоти, управляє світом через чудеса, правителів, керує силами природи.

Рольова гра

Гравець асоціюється з конкретним персонажем або лідером команди, які діють відповідно до правил своїх ролей. Наприклад, лицар не може того, що чарівник, кожна роль має свої особливості, а часом від неї залежить і розвиток сюжету. Мета ігрового процесу полягає у виконанні різноманітних завдань (квестів) для розвитку одного персонажа або групи. Можливі варіанти дій залежать від обраного образу персонажа, попередньо визначеного чи формованого самим гравцем. Рольові відеоігри часто поєднуються з іншими жанрами, утворюючи Action-RPG,тактичні рольові ігри,MUD-и і т.п.. Особливим випадком рольової гри є MMORPG, багатокористувацькі рольові онлайн-ігри, в яких живі гравці взаємодіють одне з одним у віртуальному світі через мережу Інтернет, що визначає специфіку ігрового процесу. Іноді рольові відеоігри поділяються за дизайном і побудовою сюжету. Так існує умовний поділ на рольові ігри західного зразка та східного

Стимулятор

В широкому розумінні всі ігри є стимуляторами. У вужчому значенні це відеоігри, призначені для складання уявлення про дійсність за допомогою відображення певних реальних явищ та властивостей у віртуальному середовищі. Існує чимало під жанрів, як технічні (управління складними технічними пристроями, авіаційною технікою та інші, наприклад граІл-2 Штурмовик), аркадні (відрізняються від аркад наявністю спрощеної фізичної моделі. Наприклад, X-Wing), спортивні, економічні, побачень та інші.

Пригоди в пригодницьких відеоіграх гравець керує ігровим персонажем, який рухається по сюжету та виконує зумовлені сценарієм завдання, покладаючись на свою уважність та логіку, здійснює пошуки підказок і вирішує загадки. Всередині жанру виділяються основні під жанри:інтерактивна література,інтерактивні фільми та візуальні романи. Часто за аналогією до пригодницьких фільмів пригодницькими називаються ті відеоігри, сюжет яких динамічно розгортається, насичений яскравими подіями, швидкою зміною обстановки, а персонажі проявляють кмітливість та сміливість. Приклади: серія про Індіану Джонса, Disney's Aladdin in Nasira's Revenge,Syberia.

Інші різновиди

Настільна гра — електронні реалізації настільних ігор. Серед них існують як реалізації класичних ігор (шахи,преферанс), так і специфічних як Magic: The Gathering. Головоломки — вимагають від гравця вирішення логічних завдань, передбачення можливих ситуацій. Приклади:Тетріс,Angry Birds.Games with a Purpose — програми, за допомогою яких люди використовують свої знання для вирішення проблем (передусім у наукових дослідженнях), при цьому граючись. Такі ігри є різновидом людино-орієнтованого методу комп'ютерного моделювання. Приклади:Дарвін,C Robots[8].

1.4 Кіберспорт

Кіберспорт — спортивні змагання звідеоігор. Історія електронного спорту почалася з гри Quake, яка мала режим мережевої гри через LAN або інтернет. Завдяки популярності гри Doom, в1997р. в США з'явилася перша ліга електронного спорту — Cyberathlete Professional League (CPL). Від цього року з'явилися багато нових ліг із кіберспорту. Змагання з кіберспорту, у тому числі і міжнародні, проводяться по всьому світі. Найзначнішим з них є турнір World Cyber Games, організований подібно до Олімпійських ігор. WCG були започатковані в 2000 році в Південній Кореї, і з того часу проводиться щороку, у тому числі і в інших країнах. У Південній Кореї кіберспорт отримав найбільший розвиток, отже там навіть існують телевізійні канали, які транслюють змагання з електронного спорту.

Великі змагання проводяться в спеціальних місцях, де публіка може спостерігати за гравцями, що сидять за комп'ютерами, а хід змагань відстежувати на великому екрані, де транслюється ігровий процес. У Південній Кореї, через велике число глядачів, подібні змагання проводять на стадіонах. Менш масштабні змагання проводяться в комп'ютерних клубах. Крім того, змагання можуть проводитися через інтернет. Гра через інтернет володіє деякими недоліками. У різних гравців можуть бути різні затримки передачі інформації через глобальну мережу в зв'язку з її неоднорідністю. Під час гри через інтернет складно виявити шахрайство партнерів. У контрасті, під час гри через локальну мережу всі гравці присутні в одному приміщенні під наглядом організаторів змагання, тому шахраювати набагато важче. Локальна мережа зводить нанівець і проблему затримок, оскільки має достатню і однакову для всіх пропускну здатність. На великих змаганнях призовий фонд може сягати 1000000 доларів або більше.

Найбільший виграш за історію кіберспортивних змагань виграли команда Na`Vi, яка перемогла у фіналі чемпіонату The international з дисципліни Dota 2, отримавши 1000000$. Призові фонди спонсорують компанії, пов'язані з виробництвом комплектуючих і периферії для комп'ютерів. [16]. Для кіберспорту підходять такі жанри відеоігор, як шутери від першої особи,стратегії реального часу і спортивні стимулятори як найбільш видовищні і динамічні. Поточні популярні професійні відеоігри включають: Counter-Strike, DotA2, FIFA, Halo 2, Heroes of Newerth, League of Legends, osu!, Point Blank, Quake,Starcraft, Unreal Tournament, Warcraft, World of Tanks

Розробник комп'ютерної гри має знати всі аспекти створення ігор – технічні (особливості художнього і звукового оформлення, програмування і інтерфейсу) і ділові (маркетинг, управління розробкою проекту, бюджетування). Розробник повинен знати особливості цільової аудиторії, розуміти мету гри, орієнтуватися в ігрових жанрах. Цільова аудиторія – це люди, на яких розраховано гру. Цільовою аудиторією можуть бути діти, чоловіки, жінки, що належать до різних вікових і соціальних груп.

1.5 Поняття розробки

Основними елементами комп'ютерної гри є зображення, звук, інтерфейс.

Зображення – це те, що гравець бачить на екрані. Для створення зображень використовують програмні інструменти і методи. Перед створенням зображення художнику-розробнику необхідно розуміти, як буде використовуватися зображення.

Етапи створення зображення:

1) створення ескізу на папері;

2) підготовка двомірного зображення з використанням 2D-програм;

3) конструювання та візуалізація тримірних об’єктів за допомогою 3D-програм.Комп’ютерна графіка буває 2D (двомірна, об’єкти є плоскими) і 3D (тримірна).

У 3D зображуються об’ємні об’єкти в осях прямокутної системи координат, вісь X – горизонтальна, вісь Y – вертикальна, вісь Z спрямована на спостерігача або від нього. Мінімальним елементом зображення є піксель – точка. Пікселі зображення розташовано по рядках і стовпцях. При значному збільшенні зображення пікселі перетворюються на великі зерна. Ступінчастість зображення згладжують за допомогою методу інтерполяції. Роздільна здатність (resolution) – це кількість пікселів на одиницю розміру зображення по горизонталі і вертикалі. Якість зображення вимірюється в одиницях dpi – кількість пікселів на дюйм зображення. Важливим показником роздільної здатності є відношення ширини пікселя до його висоти (aspect ratio), тому що не всі пікселі мають форму квадрата.

Пікселі зображення характеризуються значеннями координат на екрані і кольором. Колір задається номером і являє собою в палітрі RGB суміш червоного (Red), зеленого (Green) і синього (Blue) кольорів. Кожен піксель може приймати один колір з невеликої (від 2 до 256) кількості кольорів. З урахуванням глибини кольору кількість кольорів розширюється до мільйонів. Глибина кольору – кількість біт, що використовується для зберігання і представлення кольору при кодуванні одного пікселя растрової графіки або відео - зображення.

У ході розробки проекту необхідно маніпулювати зображеннями – вирізати, копіювати, обертати і т.д.:

- Cut – виділений фрагмент зображення видаляється зі сцени;• Copy – виділений фрагмент зображення копіюється в буферну пам’ять і може бути вставлений в іншому місці;
- Paste – фрагмент зображення вставляється з буферної пам’яті в інше місце;
- Skew – перекіс, нахил, викривлення, деформація зображення;
- Rotate – обертання зображення або поворот на заданий кут;
- Resize – зміна розміру, масштабу зображення;
- Crop – обрізання зображення відповідно до заданого контуру;
- Flip – дзеркальне відображення зображення щодо горизонтальної або вертикальної осі.

Крім названих простих операцій можна застосовувати і більш складні.

- спрайт – невеликі зображення (персонажі, предмети і ін.), що можна переміщати по екрану. До спрайту можна застосовувати анімацію. Невеликі спрайти можуть рухатися по великому зображенню. При анімації створюється ряд шаблонних зображень об’єкта, які показують на екрані в певній послідовності, що створює ефект руху;
- маска – зображення, що як трафарет накладається на інше (фонове) зображення.

Маска може робити частину зображення невидимою. У ряді ігор для визначення прозорого кольору використовують певний елемент (зазвичай це останній елемент) з таблиці кольорів. В цьому випадку при візуалізації для імітації прозорості областей, що зафарбовані певним кольором, використовується довільний колір – останній з таблиці кольорів. Для створення частково прозорого зображення з початкового кожному пікселю результуючого зображення привласнюють деяке значення з проміжку між кольором фону і накладеної маски. При згладжуванні (anti-aliasing) нерівностей контура намагаються забезпечити перехід кольору

від краю зображення до кольору фону, при цьому ступінчастість стає менш помітною. Графічні зображення можуть зберігатися в файлах з різними форматами в залежності від апаратних можливостей, дизайну, аспектів безпеки. Основні фактори вибору формату збереження – якість і зручність використання зображення. Слід знаходити компроміс між суперечливими вимогами – якістю зображення та його розміром.

Звукове оформлення має велике значення для гри. Звуковий супровід є компонентом, що визначає якість гри і професіоналізм розробника, задоволеність користувача процесом гри.мЗвук має великий вплив на сприйняття користувача. Звукові ефекти і звуковий супровід можна створити самостійно, записавши відповідні звуки, або з інших джерел. Професійні розробники мають для цього спеціальні інструменти.

Розробка комп'ютерних ігор – це досить чітко налагоджений процес, що має певні етапи, всі ці етапи проходять так чи інакше при створенні ігор. Однак життя зазвичай вносить свої корективи навіть у найчіткіший план. Дуже часто розробники ігор не можуть встигнути доробити гру в скільки-небудь прийнятний термін – яскравий приклад – DukeNukem Forever, випуску якого весь ігровий чекав багато років. Практично завжди після виходу комп'ютерної гри за нею йдуть виправлення – вся справа в тому, що розробники, знову ж таки, не вкладаються у відведені їм терміни.

Тексти програм ігор нерідко «йдуть» в Інтернет і всі грають в новітню гру задовго до її офіційного релізу. Причому тут не можна однозначно сказати, чи шкідливо це для ігрових компаній. З одного боку – шкідливо – адже копії гри потрапляють до користувачів абсолютно безкоштовно (не рахуючи витрат на трафік в Інтернеті). Однак з іншого – часто «йдуть» коди, далекі від досконалості і «витік» лише розігріває інтерес до фінальної версії гри. Треба врахувати, що в ігровому бізнесі існують два типи компаній – розробники (developer) і видавці (publisher). Якщо розробник і видавець збігаються – процес розробки

гри лише виграє – розробнику немає потреби переконувати стороннього видавця в доцільності капіталовкладень у розробку. Розглянемо етапи розробки типової комп'ютерної гри.

Підготовка до виробництва гри – це перший етап роботи над грою. Завдання розробників на цьому етапі – розробити концепцію гри, дизайн персонажів, вибрати засоби для реалізації проекту, створити прототип гри, підготувати план, за яким буде створюватися гра, і узгодити цей план з начальством, або – з компанією, яка планує видавати гру. Як правило, всі сучасні ігри пишуться під конкретного видавця, що вкладає в розробку чималі кошти. Коли всі адміністративні питання вирішені, гра переходить на етап виробництва.

Виробництво – це ключовий етап у створенні гри. Розробники займаються реалізацією раніше створеного плану. Однак початковий план гри піддається змінам – іноді ці зміни відбуваються дуже часто – аж до щоденних коригувань. В ході виробництва гри – особливо це стосується комерційних версій – періодично влаштовується розгляд поточних результатів розробки, до якого команда має представити проект, що досяг певного рівня розвитку. Тобто, наприклад, до одного з таких моментів має бути готовою демо-версія гри, що працює, до іншого – перший рівень і т. д. Як правило, ці проміжні результати служать відмінною рекламою нових ігрових проектів – демо-версії публікують на ігрових сайтах, геймери «приміряють» до цих версій можливості свого обладнання.

Після того, як гру створено, протестовано і налаштованою, настає час її випуску. Як правило, інтерес до цієї події посилено підігрівається видавцем гри – адже не варто забувати, що головна мета видавця – прибуток. Як правило, найбільш успішні ігри з надлишком виправдовують очікування видавців. Ігри для ПК часто виходять з помилками – вся справа в тому, що розробникам вічно не вистачає часу, щоб все як слід налагодити. На щастя, є можливість виправляти помилки на вже встановлених іграх,

встановлюючи патчі (від англійського patch – латка). Цим користуються розробники, випускаючи сируваті ігри і, після цього, цілу низку латочок (патчів) для них. Така практика не поширена для консольних ігор – тут розробники змушені відповідальніше підходити до своєї роботи і випускати повністю робочу гру, яка не потребує втручань.

Будь-яка гра складається з багатьох частин. Але серед них можна виділити дві основні. Перша – це ігрові ресурси, і друга – це програмний код. Ігрові ресурси – це графічні, музичні та інші ресурси, що використовуються для оформлення гри. Розглянемо різні формати, в яких зберігаються ігрові ресурси. Графічні файли використовуються для зберігання зображень. Як правило, це – графічні зображення ігрових об'єктів, які застосовуються в двомірних або в псевдотримірних іграх, коли тримірний вигляд ігрових об'єктів досягається лише художніми засобами, без використання технологій тримірної графіки. Так само ці файли можуть бути використані для створення елементів оформлення гри і як текстури для накладення на тримірні моделі. JPEGJPEG – це стандарт зберігання файлів зображень, розроблений спеціально для зберігання цифрових фотографій. JPEG розшифровується як Joint Photographic Experts Group – саме так називалася робоча група, що розробила цей стандарт. Особливістю JPEG є можливість стиснення зображення з втратами якості. Як правило, в JPEG виділяють 10 рівнів стиснення (від 1 до 10), проте ці зображення можна стискати і з більш точним зазначенням рівня стиснення. Чим сильніше стиснення – тим більше втрати якості. На невисоких рівнях стиснення JPEG-файли практично не містять так званих артефактів стиснення. У випадку з JPEG це проявляється в помітному спотворенні зображення, особливо в тому, що містить чіткі межі між кольорами, чіткі лінії. Справа в тому, що при стисненні за алгоритмом JPEG-зображення розбивається на фрагменти 8x8 пікселів.

Як правило, в JPEG зберігають різні двомірні зображення, які можна використовувати для організації елементів інтерфейсу користувача, для створення різних екранів гри (екран вітання, екрани з інформацією про набрані очки і так далі). В JPEG можна зберігати зображення (їх називають спрайтами), що використовуються в двомірних іграх в якості ігрових об'єктів (переважно – прямокутної форми), в якості фонових зображень.

Найкраще використовувати JPEG для зберігання кольорових зображень з плавними колірними переходами. Зокрема, це можуть бути фотографії, намальовані ігрові об'єкти та ін. JPEG-файли мають розширення JPG або JPEG. Для створення JPEG-файлів з успіхом можна використовувати популярні графічні редактори – такі, як Adobe Photoshop різних версій. PNG Формат PNG (Portable Network Graphics) відрізняється від формату JPG тим, що використовує алгоритми стиснення, що стискають зображення без втрати якості. В результаті виявляється, що зображення, які мають чіткі переходи кольорів (такі, як схеми, графіки, зображення для простих елементів управління) можуть бути досить сильно стиснуті в PNG-файлах.

Якщо зображення має плавні переходи кольорів – кращим вибором для його стиснення буде JPG, хоча й PNG цілком можна використовувати. Як і JPG, даний формат підтримує повнокольорові зображення, проте зображення можна створити з використанням декількох фіксованих кольорів, що йде на користь розміру графічного файлу, і, природно, при правильному підборі кольорів, не позначається на його якості. Формат PNG було розроблено для заміни застарілого, але все ще популярного формату GIF. Основне призначення GIF-файлів – проста інтернет-графіка. PNG-файли мають розширення .PNG.

Для їх створення можна використовувати растрові графічні редактори, – такі, як Adobe Photoshop. Ще одна важлива особливість формату PNG –

підтримка прозорості. Тобто при створенні файлу можна вказати, які з пікселів зображення вважати прозорими. В результаті PNG ідеально підходить для зберігання графічного зображення двомірних ігрових об'єктів складної форми. TGATGA(Truevision Graphic Adapter), TARGA – (Truevision Advanced Raster Graphics Adapter) – це графічний формат, що використовується переважно для зберігання ігрових текстур. Цей формат було спеціально створено для зберігання текстур, він використовується у багатьох існуючих іграх саме для цих цілей. Він підтримує повнокольорові зображення (8, 16, 24, 32-бітові), підтримує 8-ми бітний альфа-канал для зазначення прозорих ділянок зображення. Текстура - це растрове (точкове) зображення, що накладається на поверхню полігону,

Текстури дозволяють надавати тримірним моделям кольору, створювати видимість рельєфу, дрібних деталей моделі. При проектуванні 3D-моделі створення дрібних деталей зазвичай потребує дуже багато ресурсів, використання ж текстур дозволяє отримувати привабливий вигляд моделі, створеної з використанням порівняно невеликої кількості тримірних елементів. Сьогодні тримірні моделі створюються з високим рівнем деталізації – потужності сучасних комп'ютерів вистачає для реалістичної обробки таких моделей. В іграх минулих років моделі зазвичай виглядають схематично (наприклад, профіль колеса автомобіля в іграх, які використовують низько полігональних моделей, може виглядати як багатокутник, а не коло). Текстури розрізняються за глибиною кольору і розподільною здатністю – чим вище те та інше, тим вищою є якість зображення. Відносно текстур існує таке поняття, як піксель – число пікселів, що припадає на мінімальну одиницю текстури. TGA-файли мають розширення TGA.

Для створення TGA-файлів можна застосовувати все той же Adobe Photoshop та інші редактори. Наприклад, безкоштовний редактор Paint.NET (http://www.getpaint.net/) DDSDDS (Direct Draw Surface) – це графічний

формат, розроблений спеціально для використання в DirectX SDK. Його призначено переважно для зберігання текстур. Завдяки особливостям формату – це підтримка збереження зріджених і незжатих текстур, апаратна підтримка, цей формат є ідеальним для зберігання ігрових текстур. DDS-файли мають розширення DDS. Для їх створення можна використовувати плагіни для звичайних графічних редакторів, для роботи з такими файлами створено спеціальні набори утиліт і плагінів. BMPBMP (Bitmap) – це графічний формат, який зберігає зображення у стисненому вигляді. Він підтримує алгоритм стиснення, який проте використовується досить рідко. Створювати BMP-файли можна практично у всіх растрових редакторах. Особливість BMP – широке поширення при створенні графічних інтерфейсів користувача в різних Windows-програмах. BMP-файли зазвичай мають порівняно великий розмір.

Тримірні моделі використовуються в тримірних об’ємних іграх. Для додання реалістичного вигляду на такі моделі зазвичай накладають файли текстур. FBX, XFBX (Autodesk FBX) – це універсальний формат для зберігання тримірних моделей і супутньої інформації. Він використовується у багатьох тримірних додатках, зокрема, дуже популярний в комп’ютерних іграх. Існують плагіни для популярних програм створення тримірної графіки.

Формат X – це формат тримірних файлів, який використовується DirectX. Файли опису шрифтів, SPRITEFONTSPRITEFONT-файл – це xml-файл з настройками шрифту. Він містить інструкції, що стосуються візуалізації шрифту в ігровому вікні XNA-проекту. Файли цього типу мають розширення .SPRITEFONT.

Звукові файли MP3 – стиснений звуковий файл. Цей формат має величезну популярність за рахунок можливості сильного стиснення звуку з прийнятною якістю. WAV – зазвичай являє собою нестиснений файл звуковий файл, як правило, може використовуватися для відтворення

коротких звукових фрагментів (звукових ефектів) високої якості. Всі названі вище формати файлів – це ігрові ресурси, що можуть бути включені в ігровий проект. З огляду на тенденцію до зближення технологій XNA і Silverlight, що спостерігається сьогодні, не можна чітко розділити названі ігрові ресурси між двома технологіями. Ігрова термінологія складається під сильним впливом англомовних назв. Часто в українській мові досить складно знайти відповідну аналогію прийнятим англійським термінам, тому в сфері ігрової термінології існує подвійність – досить часто творці ігор користуються англомовними термінами, в той же час як і україномовні назви так само знаходять застосування.

Розглянемо деякі терміни, які використовуються в ігровій індустрії. Sprite – цей термін часто замінюють україномовним неологізмом спрайт – словник Lingvo 12 визначає поняття «спрайт» як «елемент динамічного графічного відображення». В ігровій індустрії синонімами слова спрайт є такі слова, як «зображення», «картинка», іноді користуються словом «текстура», проте зазвичай це поняття має дещо інше смислове навантаження. Як правило, спрайт – це двомірне зображення, причому у вузькому сенсі слова – це лише зображення, а в широкому – це ігровий об'єкт, який має набагато більш широкий набір можливостей, ніж звичайне зображення. Спрайти мають прямокутну форму, проте в комп'ютерних іграх часто зустрічаються непрямокутні об'єкти. Це досягається за рахунок прозорих областей при малюванні спрайту. Texture – текстура. Зазвичай текстурами називають двомірні зображення, які «накладають» на тримірні моделі. У термінології XNA поняття текстури і спрайту при розробці двовимірних ігрових об'єктів збігається.

Background – фон. Так називається зображення, зазвичай відповідає розмірами ігровому полю, та є фоном для інших зображень. Фон може бути нерухомим і рухомим. Рухомий фон (scrolling background) використовується в іграх, які називаються Скролери (scrollers). Скролінг – це один з

прийнятих ігрових термінів. Він означає прокрутку, переміщення вмісту вікна. Скроллінгові ігри надзвичайно поширені серед двомірних ігор. Наприклад, використання скролінгового фону дозволяє створити ілюзію руху в двомірному гоночному симуляторі. Фон у двомірній грі може складатися з декількох частин, що рухаються з різною швидкістю – це дозволяє створити ефект тримірності ігрового світу.2D-game – двомірна гра – гра, в якій використано двомірні зображення.3D-game – тримірна гра – використовує тримірні моделі та тримірний ігровий світ.

Tile- тайл – невелике зображення, яке використовується для конструювання рівнів в іграх. Tile можна перевести як «мозаїка» або «черепиця». Polygone (полігон, багатокутник) – просторовий багатокутник, який використовується для створення тримірних об'єктів. Як правило, у комп'ютерній графіці використовуються трикутники. Pixel (піксель) – найменший елемент растрового зображення, точка, яка відображається на екрані. Зазвичай в пікселях вимірюють роздільну здатність текстур (наприклад – 1024x768), екранну роздільну здатність монітора, розміри ігрових вікон. Слово Pixel – це абревітура від Picture's ELement.Texe l (тексель) – точка текстури в тримірному просторі. Слово Texel – це скорочення від Texture Element.Voxel (воксель) – точка тримірного зображення.

Це слово – абревіатура від Volumetric Pixel – об'ємний піксель. Texture Filtering (фільтрація текстур) – зменшення спотворень при накладенні текстур на тримірний об'єкт. Camera (камера) – так називають точку в ігровому просторі, з якою гравець бачить ігровий світ. Відносно положення камери ігри можна поділити на ігри від першої особи (камеру розташовано так, що реалізується вид немов очима персонажа), ігри з видом зліва-зверху (стратегії), ігри з видом ззаду – камеру розташовують зазвичай позаду гравця і трохи вище за нього. Існують ігри, де камера може займати різні позиції, наприклад, реалізують вид з очей персонажа і

вид ззаду. Transparency (прозорість) – прозорими можуть бути частини двовимірних зображень – це дозволяє створювати зображення складної форми, які фактично обмежені прямокутником. Прозорими бувають і об’єкти в тримірному ігровому світі наприклад, – це може бути прозора вода або скло, певним чином заломлюють світло. Light Model (модель освітлення) – способи моделювання освітлення об’єктів. Розробка гри починається зі створення концепції гри. Власне кажучи, концепція – це та центральна ідея, навколо якої будується все інше. Концепція гри виражається у вигляді концепт-документа.

Концепт-документ може будуватися за такою схемою:

- вступ;
- жанр і аудиторія;
- основні особливості гри;
- опис гри;
- порівняння і передумови створення гри;
- платформа;
- контакти

РОЗДІЛ 2 АНАЛІЗ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ

2.1 Дослідження існуючих інструментальних засобів

Більшість інструментальних засобів для розробки ігор можна поділити на 3 групи:

1. Фреймворки - це програмні продукти, які спрощують створення і підтримку технічно складних або навантажених проектів. Фреймворк, як правило, містить тільки базові програмні модулі, а все специфічні для проекту компоненти реалізуються розробником на їх основі. Тим самим досягається не тільки висока швидкість розробки, а й велика продуктивність і надійність рішень.

Веб-фреймворк - це платформа для створення сайтів і веб-додатків, що полегшує розробку і об'єднання різних компонентів великого програмного проекту. За рахунок широких можливостей в реалізації бізнес-логіки і високої продуктивності ця платформа особливо добре підходить для створення складних сайтів, бізнес-додатків і веб-сервісів. У той час як фреймворк диктує правила побудови архітектури додатку, задаючи на початковому етапі розробки поведінка за умовчанням, каркас, який потрібно буде розширювати і змінювати відповідно до зазначених вимог.

2. Ігровий рушій-центральний програмний компонент комп'ютерних та відеоігор або інших інтерактивних додатків з графікою, оброблюваної в реальному часі. Він забезпечує основні технології, спрощує розробку і часто дає грі можливість запускатися на декількох платформах, таких як ігрові консолі та настільні операційні системи, наприклад, GNU / Linux, Mac OS X і Microsoft Windows. Основну функціональність зазвичай забезпечує ігровий движок, що включає движок рендеринга («визуализатор»), фізичний движок, звук, систему скриптів, анімацію, штучний інтелект, мережевий

код, управління пам'яттю і багатопоточність. Часто на процесі розробки можна заощадити за рахунок повторного використання одного ігрового движка для створення безлічі різних ігор.

3. Конструктор ігор - програма, яка об'єднує в собі ігровий рушій інтегроване середовище розробки, і, як правило, включає в себе редактор рівнів, що працює за принципом WYSIWYG. Такі програми значно спрощує процес розробки ігор, роблячи його доступним любителям-непрограмістів, і можуть бути використані в початковому навчанні програмуванню [1] В роботі ми будемо досліджувати різноманітних представників всіх цих груп проте будемо приділяти більшу увагу на інструментам які мали реліз протягом останніх 5 років, оскільки світ мультимедія та ігрової індустрії дуже швидко оновлюється та прогресує, то набор інструментів 5-ти річної давності виявитися неактуальним в поточних реаліях .

2.1.1 XNA Framework Microsoft

XNA (англ. XNA's Not Acronymed) — набір інструментів з керованим середовищем часу виконання(.NET), створений Microsoft для полегшення розробки комп'ютерних ігор. Мета XNA в спробі звільнити розробку ігор від написання «повторюваного шаблонного коду» і об'єднати різні аспекти розробки ігор в одній системі. Набір інструментів XNA був анонсований 24 березня 2004 на Game Developers Conference в Сан-Хосе, Каліфорнія. Перший Community Technology Preview XNA Build був випущений 14 березня 2006.XNA Framework ґрунтується на реалізації. NET Compact Framework 2.0для розробки для Xbox 360 і. NET Framework 2.0на Windows. Він включає великий набір бібліотек класів, специфічних для розробки ігор, що підтримує максимальне повторне використання коду на всіх цільових платформах. Фреймворк виконується

на модифікації Common Language Runtime, що оптимізована для ігор. CLR доступне для Windows XP,Windows Vista, і Xbox 360. Так як ігри XNA пишуться для CLR, вони можуть бути запущені на будь-якій платформі, яка підтримує XNA Framework з мінімальними змінами або взагалі без них.

Ігри, які запускаються на фреймворку, технічно можуть бути написані будь-якою .NET-сумісною мовою, але офіційно підтримується тільки мова програмування C# та середовище швидкої розробки XNA Game Studio Express і всі версії Visual Studio 2008.XNA Framework приховує низько рівневі технологічні деталі, пов'язані з розробкою гри. Таким чином, фреймворк піклується про різницю між платформами, дозволяючи розробникам приділяти більше уваги смисловому вмісту гри. XNA Framework інтегрується з декількома інструментами, такими як XACT, для допомоги в створенні контенту. XNA Framework надає підтримку створення та двомірних, і тривимірних ігор і дозволяє використовувати можливості контролерів Xbox 360. Десктопні програми можуть поширюватися безкоштовно під поточним ліцензуванням Microsoft. Існує проект MonoGame, що представляє собою кроссплатформенную open - source реалізацію XNA з додатковими можливостями.

2.2 Construct 2

Construct 2 — це заснований на HTML5 конструктор 2D ігор, розроблений компанією Scirra. Конструктор спрямовано в першу чергу на людей, які не розуміються в програмуванні, дозволяючи швидко створювати ігри способом Drag-and-drop з використанням візуального редактора та логічної системи, заснованої на принципі поведінки та реакції. Construct 2 є прямим нащадком попередньої версії програми, Construct Classic. Функціональні особливості:Система «Події та Дії» Як і в

Construct Classic, основним методом програмування ігор та додатків у Construct 2 є використання «листів подій» (англ. event sheets), що схожі на файли рушія, які використовуються у мовах програмування. Коли виконується умова, задана користувачем в листі подій, слід за нею виконується дія чи функція. Система «Поведінки» Особливістю Construct 2 в порівнянні з іншими конструкторами є так названі «поведінки» (англ. behaviors). Поведінка — це заздалегідь заготовлений набір (шаблон) властивостей об'єкту. Поведінки потрібні для прискорення процесу розробки гри, коли користувач не задає всі властивості сам в листі подій, а просто користається необхідним шаблоном.

Прикладом поведінки є поведінка 8 direction, яка дозволяє переміщувати об'єкт у восьми напрямках за допомогою клавіш. Підтримка сторонніх плагінів. Розробники Construct 2 забезпечують та навіть заохочують створення плагінів від сторонніх розробників. Так, на офіційному сайті можна знайти поради та уроки з написання та налаштування плагінів для коректної роботи. Всі плагіни Construct 2 написано на Javascript.

2.3 Blender Game Engine

Blender — пакет для створення тривимірної комп'ютерної графіки, що включає засоби моделювання, анімації, вимальовування, після-обробки відео, а також створення відеоігор. Особливостями пакету є малий розмір, висока швидкість вимальовування, наявність версій для багатьох операційних систем —FreeBSD, GNU/Linux, Mac OS X, SGI Irix 6.5, Sun Solaris 2.8 (sparc), Microsoft Windows, SkyOS, MorphOS та Pocket PC. Пакет має такі функції, як динаміка твердих тіл, рідин та м'яких тіл, систему гарячих клавіш, велику кількість легко доступних розширень, написаних

мовою Python. Починаючи з версії 2.61 з'явились функції "відстеження камери" (англ. camera tracking), та "захоплення руху" (англ. motion capture або mocap). Програма є вільним програмним забезпеченням та розповсюджується під ліцензією GNU GPL. Ігровий рушій Blender Game Engine (BGE), є вбудованим компонентом Blender. У своєму складі він має вбудований фізичний движок Bullet, підтримку мережі через скрипти Python, графічний рендер з повноцінною підтримкою шейдеров. Blender надає чудовий інтерфейс конструктора, з можливістю легко налаштувати ігрову логіку Або як альтернатива написання скриптів Python безпосередньо звертатися до API. Для створення гри немає необхідності виходити з програми, більше того наявні інструменти що полегшують тестування , наприклад перегляд роботі шейдерів без запуску графічного рушія, тощо.

Ігровий рушій Blender використовує систему графічних «логічних цеглин» (поєднання «датчиків», «контролерів» і «приводів») для контролю руху й відображення об'єктів у рушії. Ігровий рушій також може бути розширений за допомогою набору Python палітурки. Як Blender, він використовує OpenGL як шар крос платформної графіки, для зв'язку з графічним обладнанням.

2.4 Easel JS

Easel JS - Javascript бібліотека що надає можливість працювати з Canvas у графічному режимі включаючи повний список ієрархічного відображення, модель взаємодії основного і допоміжних класів, щоб полегшити роботу з з 2D-графікоюв Canvas. EaselJS забезпечує рішення для роботи з багатою графікою і інтерактивністю в HTML5 Canvas. EaselJS також має вбудовану підтримку.

Особливостей Canvas, такі як тіні і CompositeOperation. Ticker, глобальний heartbeat, на яке об'єкти можуть підписатися.

Фільтри, в тому числі ColorMatrixFilter, AlphaMaskFilter, AlphaMapFilter і BlurFilter. Утиліта ButtonHelper, полегшує створення інтерактивних кнопок Sprite Sheet Utils і Sprite Sheet Builder, щоб допомогти побудувати і керувати функціональністю SpriteSheet під час виконання.Всі сучасні браузери, які підтримують Canvas будуть підтримувати EaselJS. Проте продуктивність може різнитися між платформами, наприклад, Android Canvas має погану апаратну підтримку, і працює, в середньому, повільніше ніж більшість інших браузерів.

2.5 Game MakerGame

Maker — один з найвідоміших конструкторів ігор. Написаний на Delphi Доступний для ОС Windows. Будучи професором Утрехтського університету Марк Овермарс почав розробляти Game Maker як навчальний посібник для своїх студентів. Створення ігор в ньому не потребує попереднього знайомства з будь-якою змов програмування. Гра в Game maker будується як набір ігрових об'єктів. За їх зовнішній вигляд відповідають спрайт, а поведінка задається шляхом опису реакцій на події. Для цього можна використовувати графічне представлення програм (близьке до блок-схемами) у вигляді послідовності іконок-дій. Програмування за допомогою дій відбувається в режимі drag-n-drop.

Наприклад, для того щоб почати умовний оператор, потрібно перетягнути на панель дій восьмикутник з іконкою, що означає тип перевірки, а потім, можливо, ввести будь-які значення в форму, що з'явилася. Для більш просунутих користувачів є скриптова мова GML схожий на

JavaScripti C++,є можливість створення власних бібліотек дій, використовуючи Library Maker.

Розрахований в основному на створення двовимірних (2D) ігор будь-яких жанрів. Також підійде для створення різних презентацій і т. п. Game Maker розповсюджується на умовах Shareware, безкоштовна версія обмежена в функціональності, а при запуску ігор на стрічці завантаження гри показується логотип Game Maker. Остання версія 8.1. Більше Game Maker не підтримується, його місце зайняло кроссплатформений розвиток проекту -Game Maker: Studio.

2.6 Unity

Unity — багатоплатформовий інструмент для розробки ігор, що працює на операційних системах Windows і OS X. Створені за допомогою Unity застосування працюють під системами Windows, OS X, Android, Apple iOS, Linux, а також на гральних консолях Wii, PlayStation 3 і XBox 360.Є можливість створювати інтернет-застосунки за допомогою спеціального під'єднуваного модуля для браузера Unity, а також за допомогою експериментальної реалізації в межах модуля Adobe Flash Player. Застосування, створені за допомогою Unity, підтримують DirectX та OpenGL.

Редактор Unity має простий Drag & Drop інтерфейс, який легко налаштовувати, що складається з різних вікон, завдяки чому можна проводити налагодження гри прямо в редакторі. Рушій підтримує три сценарних мови: C #, JavaScript (модифікація). Проект в Unity ділиться на сцени (рівні) - окремі файли, що містять свої ігрові світи зі своїм набором об'єктів,сценаріїв, і налаштувань. Сцени можуть містити в собі як, об'єкти (моделі), так і порожні ігрові об'єкти – тобто ті які не мають моделі.

Об'єкти, в свою чергу містять набори компонентів, з якими і взаємодіють скрипти.

У об'єктів з видимої геометрією також за замовчуванням присутній компонент Mesh Renderer, що робить їх модель їх видимою. Також Unity підтримує фізику твердих тіл і тканини, фізику типу Ragdoll (ганчіркова лялька). У редакторі є система успадкування об'єктів; дочірні об'єкти будуть повторювати всі зміни позиції, повороту і масштабу батьківського об'єкта. Скрипти в редакторі прикріплюються до об'єктів у вигляді окремих компонентів. При імпорті текстури в рушій можна згенерувати alpha-канал, mip-рівні, normal-map, light-map, карту відображень, проте безпосередньо на модель текстуру прикріпити не можна - буде створено матеріал, з яким буде призначений шейдер, і потім матеріал прикріпиться до моделі. Редактор Unity підтримує написання і редагування шейдерів. Крім того він містить компонент для створення анімації, анімацію також можна створити попередньо в 3D-редакторі та імпортувати разом з моделлю, а потім розбити на файли. У Unity вбудована підтримка мережі. Ігровий рушій повністю пов'язаний із середовищем розробки. Це дозволяє випробовувати гру прямо в редакторі. Має вбудований генератор ландшафтів

2.7 Unreal Engine 3

Unreal Engine 3 - Написаний мовою C++, рушій дозволяє створювати ігри для більшості операційних систем і платформ: Microsoft Windows, Linux, Mac OS і Mac OS X, консолей Xbox, Xbox 360, PlayStation 2, PlayStation Portable, PlayStation 3, Wii, Dreamcast і Nintendo GameCube. У березні 2010 робота рушія була продемонстрована на комунікаторі Palm Pre, що базується на мобільній платформі webOS.

Для спрощення портування рушій використовує модульну систему залежних компонентів: підтримує різні системи рендерингу (Direct3D, OpenGL, Pixomatic), відтворення звуку (EAX, OpenAL, DirectSound3D), засоби голосового відтворення тексту, розпізнавання мовлення (тільки для Xbox360, PlayStation 3, Nintendo Wii і Microsoft Windows), модулі для роботи з мережею й підтримка різних пристроїв вводу. Для гри у мережі підтримуються технології Windows Live, Xbox Live, і GameSpy, включаючи до 64 гравців (клієнтів) одночасно.

Попри те, що офіційно засоби розробки не містять у собі підтримки великої кількості клієнтів на одному сервері, рушій використовувався для створення MMORPG-ігор. Один з найвідоміших представників жанру, Lineage II, використовує рушій Unreal Engine.5 листопада 2009 року був випущений пакет Unreal Development Kit, безкоштовна версія Unreal Engine 3 для некомерційного використання з можливістю купівлі дешевої комерційної ліцензії.

Для описання логіки гри використовується C++ подібний UnrealScript. Усі елементи ігрового рушія представлені у вигляді об'єктів, що мають набір характеристик, і клас, який визначає доступні характеристики. Серед основних класів і об'єктів можна виділити наступні:

Актор (actor) — базовий клас, що містить усі об'єкти, які мають відношення до ігрового процесу й мають просторові координати.

Пішак (pawn) — фізична модель гравця або об'єкта, керованого штучним інтелектом. Метод керування описаний спеціальним об'єктом, такий об'єкт називається контролером. Контролер штучного інтелекту описує лише загальну поведінку пішака під час ігрового процесу, а такі параметри як «здоров'я» (кількість пошкоджень, після яких пішак перестає функціонувати) або, наприклад, відстань, на якій пішак звертає увагу на звуки, задаються для кожного об'єкта окремо.

Світ, рівень (world, game level) — об'єкт, що характеризує загальні властивості «простору», наприклад, силу тяжіння й туман, у якому розташовуються всі актори. Також може містити в собі параметри ігрового процесу, як, наприклад, ігровий режим, для якого призначений рівень.

2.8 Turbulenz

Бібліотеки двигуна реалізовані та оптимізовані на JavaScript для підтримки швидких ітерації ігрового коду і даних. Двигун виконується безпосередньо в браузері, і включає в себе деякі з наступних функцій:Асинхронне завантаження ресурсів і обмінювати; Лінивий оцінка оновлень сцени; Багатопоточная перевірка і виконання

Візуалізація ефектів і частинок. Shader на основі режиму негайного відправлення; Підтримка мульти-техніка, багатоходової, мульти-матеріали; Динамічна вершина, індекс і обробки текстур буфера; Відкладений надання підтримки необмежені вогні; Знімні ефекти POST і колекція ефектів; Експонентний карти тіней і оклюзія запитів; Великі частинок і ефект системи; GUI система / HUD підтримки декількох мов і шрифтів

Фізика, Зіткнення і анімація. Жорсткі тіла, зіткнення примітиви і обмежень; Рей і опуклі запити розгортки; Велика колекція вбудованих контролерів анімації; Скелет анімації з кватерніонів

Мережа, Мультіплеер і соціальні мережі. Стиснення, шифрування, надійної і ненадійною передачі повідомлень; Компенсація мережу лаг, клієнт / сервер і p2p архітектури; Інтеграція з популярними соціальними мережами, включаючи Facebook; Автоматичний програвач входу і доступу автоматично обробляються

Сцена і управління ресурсами. Видимість запитів через портали, усіченого або перекриттям коробки; Сортування і угрупування для видимості і оптимальної обробки; Пропускна здатність і апаратне масштабування за допомогою динамічного вибору активів

2.9 Java

Java - строго типізований об'єктно-орієнтована мова програмування, розроблений компанією Sun Microsystems (в подальшому придбаної компанією Oracle). Розробка ведеться співтовариством, організованим через Java Community Process, мова і основні реалізують його технології поширюються за ліцензією GPL. Права на торговельну марку належать корпорації Oracle.

Програми Java транслюються у спеціальний байт-код, тому вони можуть працювати на будь-якої комп'ютерної архітектурі, для якої існує реалізація віртуальної Java-машини. Дата офіційного випуску - 23 травня 1995 року. На 2019 рік Java - один з найпопулярніших мов програмування.

Спочатку мова називалася Oak («Дуб»), розроблявся Джеймсом Гослінгом для програмування побутових електронних пристроїв. Через те, що мова з такою назвою вже існував, Oak був перейменований в Java [4]. Названий на честь марки кави Java, яка, в свою чергу, отримала найменування однойменного острова (Ява), тому на офіційній емблемі мови зображена чашка з гарячою кавою. Існує й інша версія походження назви мови, пов'язана з алюзією на каву-машину як приклад побутового пристрою, для програмування якого спочатку мова створювався. Відповідно до етимологією в російськомовній літературі з кінця двадцятого і до перших років XXI століття назва мови нерідко переводилося як Ява, а не транскрибувати.

В результаті роботи проекту світ побачив принципово новий пристрій, кишеньковий персональний комп'ютер Star7, який випередив свій час більш ніж на 10 років, але через велику вартість в 50 доларів не зміг зробити переворот в світі технології і був забутий.

Пристрій Star7 не користувалося популярністю на відміну від мови програмування Java і його оточення. Наступним етапом життя мови стала розробка інтерактивного телебачення. У 1994 році стало очевидним, що інтерактивне телебачення було помилкою.

З середини 1990-х років мова стала широко використовуватися для написання клієнтських додатків і серверного програмного забезпечення. Тоді ж певний поширення набула технологія Java-аплетів - графічних Java-додатків, вбудованих в веб-сторінки; з розвитком можливостей динамічних веб-сторінок в 2000-і роки технологія стала застосовуватися рідко.

У веб-розробці застосовується Spring Framework, для документування використовується утиліта Javadoc.

Програми на Java транслюються в байт-код Java, який виконується віртуальною машиною Java (JVM) - програмою, обробній байтовий код і передавальної інструкції обладнанню як інтерпретатор.

Перевагою подібного способу виконання програм є повна незалежність байт-коду від операційної системи і устаткування, що дозволяє виконувати Java-додатки на будь-якому пристрої, для якого існує відповідна віртуальна машина. Іншою важливою особливістю технології Java є гнучка система безпеки, в рамках якої виконання програми повністю контролюється віртуальною машиною. Будь-які операції, які перевищують встановлені повноваження програми (наприклад, спроба несанкціонованого доступу до даних або з'єднання з іншим комп'ютером), викликають негайне переривання.

Часто до недоліків концепції віртуальної машини відносять зниження продуктивності. Ряд удосконалень кілька збільшив швидкість виконання програм на Java:

- застосування технології трансляції байт-коду в машинний код безпосередньо під час роботи програми (JIT-технологія) з можливістю збереження версій класу в машинному коді,

- широке застосування переносних орієнтованого коду (native-код) в стандартних бібліотеках,

- апаратні засоби, що забезпечують прискорену обробку байт-коду (наприклад, технологія Jazelle, підтримувана деякими процесорами архітектури ARM).

За даними сайту shootout.alioth.debian.org, для семи різних завдань час виконання на Java становить в середньому в півтора-два рази більше, ніж для C / C ++, в деяких випадках Java швидше, а в окремих випадках в 7 разів повільніше. З іншого боку, для більшості з них споживання пам'яті Java-машиною було в 10-30 разів більше, ніж програмою на C / C ++. Також примітно дослідження, проведене компанією Google, згідно з яким відзначається істотно нижча продуктивність і більше споживання пам'яті в тестових прикладах на Java в порівнянні з аналогічними програмами на C ++.

Ідеї, закладені в концепцію і різні реалізації середовища віртуальної машини Java, надихнули безліч ентузіастів на розширення переліку мов, які могли б бути використані для створення програм, що виконуються на віртуальній машині. Ці ідеї знайшли також вираз в специфікації загальномовна інфраструктури CLI, закладеної в основу платформи .NET компанією Microsoft.

Програми, написані на Java, мають репутацію більш повільних і займають більше оперативної пам'яті, ніж написані на мові C. Проте,

швидкість виконання програм, написаних на мові Java, була істотно поліпшена з випуском в 1997-1998 роках так званого JIT-компілятора у версії 1.1 на додаток до інших особливостей мови для підтримки кращого аналізу коду (такі, як внутрішні класи, клас StringBuffer, спрощені логічні обчислення і так далі). Крім того, була проведена оптимізація віртуальної машини Java - з 2000 року для цього використовується віртуальна машина HotSpot. Станом на лютий 2012 року, код Java 7 приблизно в 1,8 рази повільніше коду, написаного на мові Сі.

Деякі платформи пропонують апаратну підтримку виконання для Java. Наприклад, мікроконтролери, що виконують код Java на апаратному забезпеченні замість програмної JVM, а також засновані на ARM процесори, які підтримують виконання байткода Java через опцію Jazelle.

РОЗДІЛ 3 ПРОГРАМАНА РЕАЛІЗАЦІЯ

Далі ми розглянемо зовнішній вигляд реалізації гри. Дана гра була реалізована одною з описаних мов – Java.

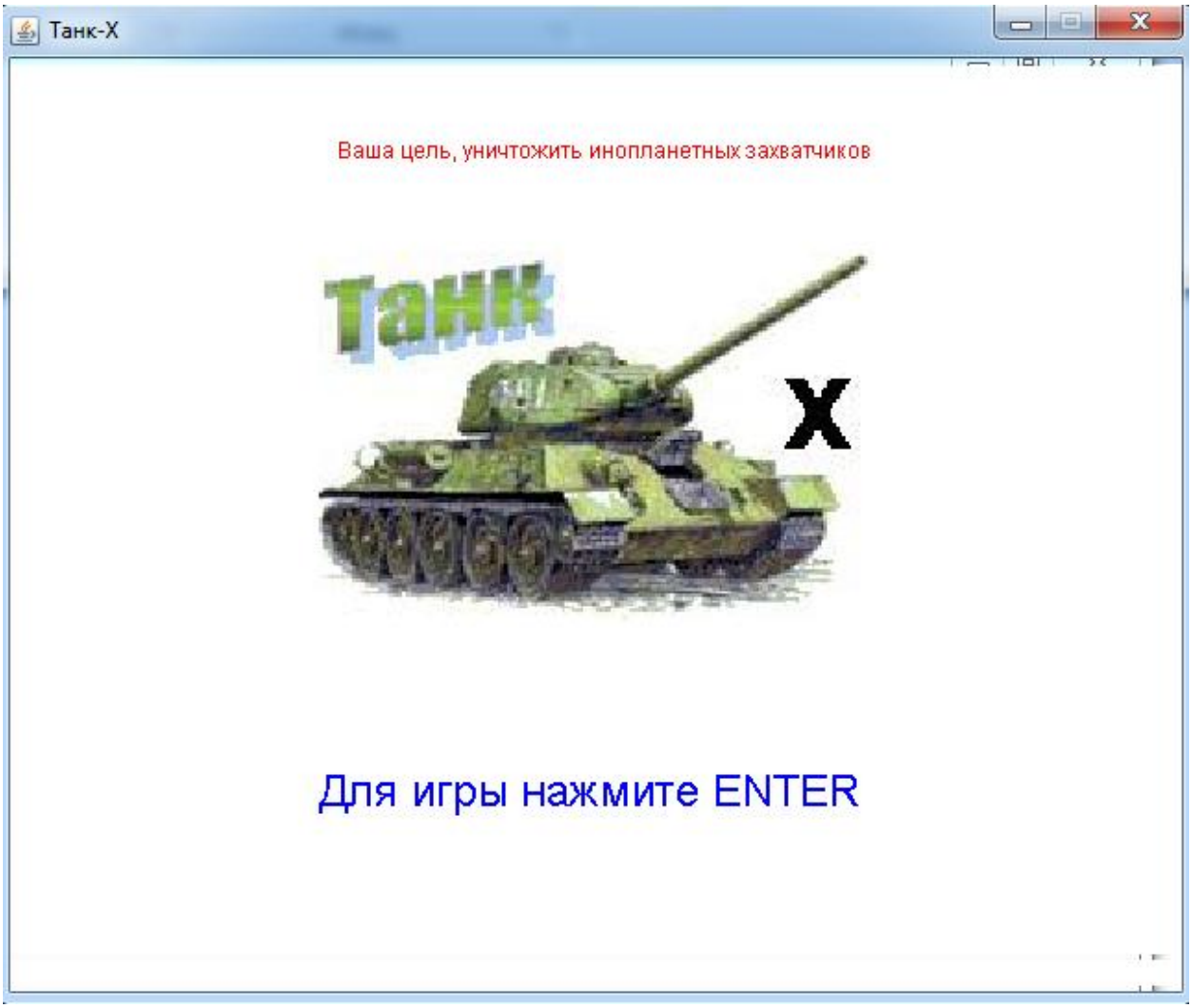

Рис. 3.1 – Вхід у гру

Задача гри: вбивати ворогів за допомогою танку, яким ми керуємо за допомогою стрілок на клавіатурі. Щоб стріляти натискаємо «пробіл».

Рис. 3.2 – 1 рівень

Після перемоги гра автоматично переходить на наступний рівень.

Всього дано 10 життів.

Рис. 3.3 – 2 рівень

Рис. 3.4 – 3 рівень

Рис. 3.5 – 4 рівень

Рис. 3.6 – 5 рівень

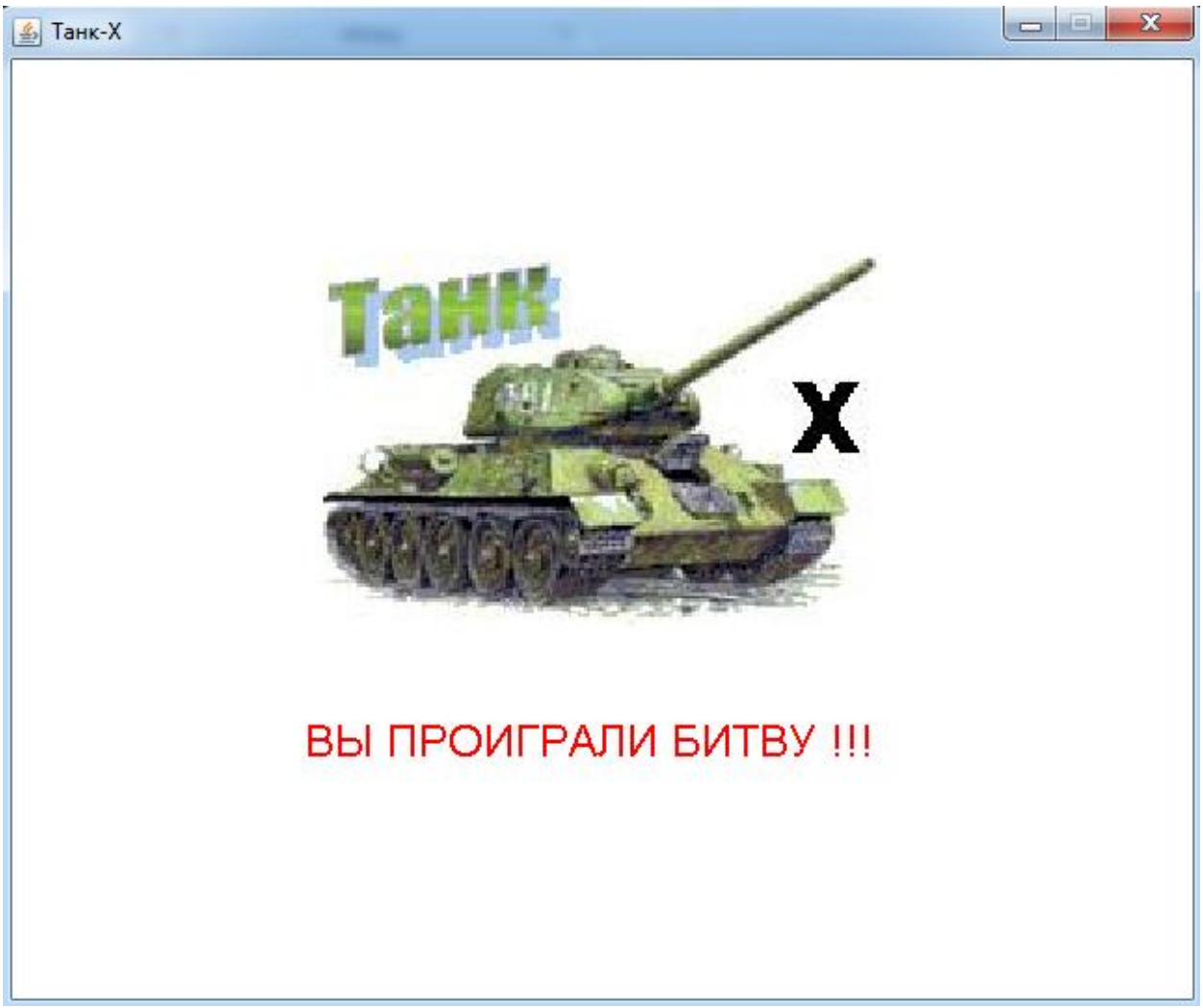

Рис 3.7 – Вікно поразки

Лістинг програми наведений в Додатку А.

ВИСНОВОК

У ході виконання дипломної роботи були виконані такі завдання:

1) досліджено розвиток індустрії комп’ютерних ігор. Були освітленні основні принципи та процеси при розробці мультимедійних ігор. Розглядались дві головні течії в індустрії ігор – комерційна та інді розробка. Також була сформулювана актуальність роботи та визначенні задачі;
2) розглянуто найбільш популярні сучасні засоби для розробки ігор. Проведено розбиття їх на класи та проаналізовано їх базові заявлені функціональні можливості;
3) виготовлення ігрового додатку.

СПИСОК ЛІТЕРАТУРИ

1.Офіційний сайт Construct2.-www.scirra.com.

2.Офіційний портал розробників XNA -https://msdn.microsoft.com/ru-RU/games-development-msdn

3.Офіційний сайт MONO game -http://www.monogame.net

4.Офіційний сайт Phaser

5.Офіційний githab репозиторій Pixijs https://github.com/pixijs/pixi.js

6. https://ru.wikipedia.org/wiki/Java

7. Смирнова Е. О. Психология и педагогика игры : учебник и практикум / Е. О. Смирнова, И. А. Рябкова. — Москва : Изд–во Юрайт, 2016. — 223 с. — ISBN 978–5–9916–6807–1.

8. Bethke E. Game development and production / Erik Bethke. — Texas : Wordware Publishing, Inc., 2003. — 432 p. — ISBN 1–55622–951–8.

9. Brathwaite B. Challenges for Game Designers / Brenda Brathwaite, Ian Schreiber. — Charles River Media, 2009. — 347 p. — ISBN 1–58450–580–X.

10. Chandler H. M. The Game Production Handbook / Chandler Heather Maxwell. — 2nd ed. — Hingham, Massachusetts : Infinity Science Press, 2009. — 442 p. — ISBN 978–1–934015–40–7

ДОДАТОК А

```
package tankx;
import java.awt.image.*;
import java.awt.*;
import java.awt.geom.AffineTransform;
import java.awt.event.*;
import javax.sound.sampled.Clip;
final class bullet {
    private final static int BUFSIZE = 32;
    private short[] x = null;
    private short[] y = null;
    private char[] dx = null;
    private char[] dy = null;
    private int   cnt = 0;

    public bullet(){
        x   = new short[bullet.BUFSIZE];
        y   = new short[bullet.BUFSIZE];
        dx  = new char[bullet.BUFSIZE];
        dy  = new char[bullet.BUFSIZE];
        cnt = 0;
    }

    public void add(int px, int py, int pdx, int pdy){
        if(cnt < bullet.BUFSIZE){
            x[cnt]  = (short)px;
```

```
        y[cnt]  = (short)py;
        dx[cnt] = (char)pdx;
        dy[cnt] = (char)pdy;
        ++cnt;
    }
}

public void remove(int index){
    if(index < cnt){
        --cnt;
        for(int i = index; i < cnt; ++i){
            x[i]  = x[i + 1];
            y[i]  = y[i + 1];
            dx[i] = dx[i + 1];
            dy[i] = dy[i + 1];
        }
    }
}

public void reset(){
    cnt = 0;
}

public int at_x(int i){
    return (int)x[i];
```

```
    }

    public int at_y(int i){
        return (int)y[i];
    }

    public void move_all(){
        for(int i = 0; i < cnt; ++i){
            x[i] += (short)dx[i];
            y[i] += (short)dy[i];
        }
    }

    public int count() {
        return cnt;
    }
}
public class xUserTank {
    private BufferedImage  itank;
    private BufferedImage  ishot;
    private bullet         shots;

    private int     x, y;
    private long    delay_mt;
    private int     key_ctrl;
```

```
private double  angle;
private int     offset;
private int     speedX, speedY;
private int     look;
private int     life;
private Clip    sfire, stank;

public final static long DELAY      = 10L;
public final static int  LOOK_LEFT  = 0;
public final static int  LOOK_RIGHT = 1;
public final static int  LOOK_UP    = 2;
public final static int  LOOK_DOWN  = 3;

public xUserTank(){
   itank    = null;
   ishot    = null;
   x        = 0;
   y        = 0;
   delay_mt = 0L;
   key_ctrl = 0;
   angle    = 0.0d;
   offset   = 0;
   speedX   = speedY = 0;
   look     = xUserTank.LOOK_UP;
   life     = 0;
```

```
    sfire    = null;
    stank    = null;
    shots    = new bullet();
    this.Create();
}

//22x38  38x22
public boolean Create(){
    try {
        itank  = xUtil.ToClip32Bits(this.getClass().getResource("image/tank.jpg"));
        ishot  = xUtil.ToClip32Bits(this.getClass().getResource("image/shot.jpg"));
        sfire  = xUtil.LoadSound(this.getClass().getResource("image/fire.wav"));
        stank  = xUtil.LoadSound(this.getClass().getResource("image/tank.wav"));

        offset = itank.getWidth() / 2;
    } catch(Exception e){
        e.printStackTrace();
        return false;
    }
    return true;
}

public void Init(int px, int py) {
    x       = px;
```

```
    y       = py;
    life    = 10;
    delay_mt = 0L;
    key_ctrl = 0;
    angle   = 0.0d;
    speedX  = speedY = 0;
    look    = xUserTank.LOOK_UP;
    shots.reset();
    this.Stop();
}
public void Stop(){
    sfire.stop();
    stank.stop();
}
public void Draw(Graphics2D dc, xField field, long time){
    boolean isdel;
    int ix, iy;

    for(int i = 0; i < shots.count(); ++i){
        ix   = shots.at_x(i);
        iy   = shots.at_y(i);
        isdel = (ix < 0 || iy < 0 || ix >= 640 || iy >= 480);

        if(! isdel){
            switch(field.Offset(ix / xField.CELL_SIZE, iy / xField.CELL_SIZE)) {
```

```
        case xField.OPEN_PATH:
        case xField.WOOD_PATH:
            break;
        case xField.BASE_PATH:
        case xField.BRICK_PATH:
            field.AddBoom(ix, iy, 1, false);
            isdel = true;
            break;
        default:
            isdel = true;
            break;
        }
    }

    if(isdel){
        shots.remove(i);
        --i;
        continue;
    }
    dc.drawImage(ishot, ix, iy, null);
}

AffineTransform affmat = dc.getTransform();
affmat.rotate(angle, x + offset, y + offset);
dc.setTransform(affmat);
```

```
    dc.drawImage(itank, x, y, null);

    affmat.setToIdentity();
    dc.setTransform(affmat);

    if((time - delay_mt) > xUserTank.DELAY){
        delay_mt = time;
        if(this.test_collision(key_ctrl, field)){
            x += speedX;
            y += speedY;
        }
        shots.move_all();
    }
}


public void KeyPress(int key, xField field){
    int dx, dy;
    if(key == KeyEvent.VK_SPACE) {

        dx = dy = 0;
        switch(look){
        case xUserTank.LOOK_DOWN:
            dy = 3;
            break;
```

```
            case xUserTank.LOOK_UP:
               dy = -3;
               break;
            case xUserTank.LOOK_LEFT:
               dx = -3;
               break;
            case xUserTank.LOOK_RIGHT:
               dx = 3;
               break;
            }
            shots.add(x + offset - ishot.getWidth()/2, y + offset - ishot.getHeight()/2,
dx, dy);

            if (! sfire.isRunning()) {
               sfire.setFramePosition(0);
               sfire.start();
            }

         } else {

            switch(key){
            case KeyEvent.VK_A:
            case KeyEvent.VK_LEFT:
               speedX = -1;
               angle  = -Math.PI * 0.5d;
               look   = xUserTank.LOOK_LEFT;
```

```
    break;
case KeyEvent.VK_D:
case KeyEvent.VK_RIGHT:
    speedX = 1;
    angle  = Math.PI * 0.5d;
    look   = xUserTank.LOOK_RIGHT;
    break;
case KeyEvent.VK_W:
case KeyEvent.VK_UP:
    speedY = -1;
    angle  = 0.0d;
    look   = xUserTank.LOOK_UP;
    break;
case KeyEvent.VK_S:
case KeyEvent.VK_DOWN:
    speedY = 1;
    angle  = Math.PI;
    look   = xUserTank.LOOK_DOWN;
    break;
}

if(this.test_collision(key, field)){
    key_ctrl = key;

    if(! stank.isRunning())
```

```
                stank.loop(Clip.LOOP_CONTINUOUSLY);

        } else {
            stank.stop();
            key_ctrl = 0;
        }
    }
}


public void KeyUp(int key){
    if(key != KeyEvent.VK_SPACE){
        stank.stop();
        key_ctrl = speedX = speedY = 0;
        this.Reset();
    }
}


public int getX(){
    return x;
}

public int getY() {
    return y;
```

```
    }

    public int getSize(){
        return itank.getWidth();
    }

    public int getLife(){
        return life;
    }

    public int decLife(){
        return --life;
    }

    public bullet getBullets(){
        return shots;
    }


//**************************************************************************
**********************************************************

     private boolean test_collision(int key, xField field){
          boolean iscl;
          int offx, offy;
```

```
switch (key) {
case KeyEvent.VK_A:
case KeyEvent.VK_LEFT:

    offx = (x - 1) / xField.CELL_SIZE;
    iscl = (x - 1) < 0;
    if (iscl || field.IsClose(offx, y / xField.CELL_SIZE) ||
        field.IsClose(offx, (y + itank.getHeight()) / xField.CELL_SIZE)) {
        this.Reset();
        return false;
    }

    break;
case KeyEvent.VK_D:
case KeyEvent.VK_RIGHT:

    offx = (x + itank.getWidth() + 1) / xField.CELL_SIZE;
    iscl = (x + itank.getWidth() + 1) >= 640;
    if (iscl || field.IsClose(offx, y / xField.CELL_SIZE) ||
        field.IsClose(offx, (y + itank.getHeight()) / xField.CELL_SIZE)) {
        this.Reset();
        return false;
    }

    break;
```

```
case KeyEvent.VK_W:
case KeyEvent.VK_UP:

    offy = (y - 1) / xField.CELL_SIZE;
    iscl = (y - 1) < 0;
    if (iscl || field.IsClose(x / xField.CELL_SIZE, offy) ||
        field.IsClose((x + itank.getWidth()) / xField.CELL_SIZE, offy)) {
        this.Reset();
        return false;
    }

    break;
case KeyEvent.VK_S:
case KeyEvent.VK_DOWN:

    offy = (y + itank.getHeight() + 1) / xField.CELL_SIZE;
    iscl = (y + itank.getHeight() + 1) >= 480;
    if (iscl || field.IsClose(x / xField.CELL_SIZE, offy) ||
        field.IsClose((x + itank.getWidth()) / xField.CELL_SIZE, offy)) {
        this.Reset();
        return false;
    }

    break;
}
```

```
        return true;
    }

    public void Reset(){
       speedX = speedY = key_ctrl = 0;
    }
}
package tankx;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.JOptionPane;
import tankx.xField;
import java.awt.Color;
public class DlgFrame extends JFrame {
    JPanel    pane;
    Image     ilogo       = null;
    Font      font        = null;
    xUserTank user_game   = null;
    xField    field_game  = null;
    Timer     timer_update = null;
    int       level       = 0;
    int       state       = 0;
    boolean   ilevel      = false;
    long      lvltime     = 0L;
```

```
private final static int GAME_START  = 1;
private final static int GAME_PLAY   = 2;
private final static int GAME_OVER   = 3;
private final static int GAME_WINNER = 4;


public DlgFrame() {
    try {
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        jbInit();
    } catch (Exception exception) {
        exception.printStackTrace();
    }
}

/**
 * Component initialization.
 *
 * @throws java.lang.Exception
 */
private void jbInit() throws Exception {
    pane = (JPanel) getContentPane();
    pane.setLayout(null);
```

```
        setSize(new Dimension(640, 512 + 20));
        setResizable(false);
        setTitle("Òàíê-Õ");
        setBackground(Color.WHITE);

        font  = new Font("Arial", Font.PLAIN, 24);

        ilogo =
javax.imageio.ImageIO.read(this.getClass().getResource("image/logo.jpg"));
        state = GAME_START;

        field_game = new xField();
        if(! field_game.Create(640, 480)){
           JOptionPane.showMessageDialog(this, "Îøèáêà ïðè ñîçäàíèå õîëñòà !",
"Îøèáêà", JOptionPane.ERROR_MESSAGE);
           return;
        }

        user_game = new xUserTank();

        ActionListener calltimer = new ActionListener() {
           public void actionPerformed(ActionEvent evt) {
              DlgFrame.this.timer();
           }
        };
        timer_update = new Timer(10, calltimer);
```

```
    timer_update.start();
    pane.setBorder(null);
    pane.setDoubleBuffered(false);

    addKeyListener(new DlgFrame_keyAdapter(this));
}


//
public void timer(){
    this.repaint();
}


@Override
public void paint(Graphics dc) {
    Font      ftmp;
    Graphics2D dc2;
    long time = System.currentTimeMillis();

    if(state == GAME_PLAY) {
        field_game.Erase();

        user_game.Draw(field_game.GetDC(), field_game, time);
        field_game.Draw(time, user_game);
```

```
        if (ilevel) {
            dc2  = field_game.GetDC();
            ftmp = dc2.getFont();
            dc2.setFont(font);
            dc2.setColor(Color.YELLOW);
            dc2.drawString("ÓÐÎÂÅÍÜ - " + (level + 1), getWidth() / 2 - 60,
getHeight() / 2 - 60);
            dc2.setFont(ftmp);
            if ((time - lvltime) > 1500L) {
                lvltime = time;
                ilevel = false;
            }
        }

        dc.drawImage(field_game.Handle(), 0, 29, null);

        dc.setColor(Color.WHITE);
        dc.fillRect(0, 512, getWidth(), 14);
        dc.setColor(Color.BLUE);
        dc.drawString("Æèçíü: " + user_game.getLife(), 10, getHeight() - 8);
        dc.drawString("Óðîâåíü: " + (level + 1), getWidth() - 80, getHeight() - 8);

        field_game.IsBulletsCollision(user_game);
        field_game.IsCollision(user_game);

        if(user_game.getLife() < 0){
```

```
            field_game.Stop();
            user_game.Stop();
            dc.setColor(Color.WHITE);
            dc.fillRect(0, 0, getWidth(), getHeight());
            state = GAME_OVER;
        }
        if(field_game.getCountAliens() == 0){
            if(! field_game.StartLevel(++level, user_game)){
                field_game.Stop();
                user_game.Stop();
                state = GAME_WINNER;
            }
            ilevel  = true;
            lvltime = System.currentTimeMillis();
        }

    } else if(state == GAME_START) {
        dc2 = (Graphics2D)field_game.GetDC();
        dc2.setColor(Color.WHITE);
        dc2.fillRect(0, 0, getWidth(), getHeight());
        dc2.drawImage(ilogo, (getWidth() - 300)/2, 100, 300, 200, null);

        dc2.drawString("Óïðàâëåíèå â èãðå: êëàâèøè âëåâî, âïðàâî, âíèç, ââåðõ,
ñòðåëüáà-ïðîáåë", 97, 490);
        dc2.setColor(Color.RED);
```

```
            dc2.drawString("Âàøà öåëü, óíè÷òîæèòü èíîïëàíåòíûõ çàõâàò÷èêîâ", 180,
50);


            ftmp = dc.getFont();
            dc2.setFont(font);
            dc2.setColor(Color.BLUE);
            dc2.drawString("Äëÿ èãðû íàæìèòå ENTER", 170, 400);
            dc2.setFont(ftmp);
            dc.drawImage(field_game.Handle(), 0, 29, null);
            dc.setColor(Color.WHITE);
            dc.fillRect(0, 512, getWidth(), 14);
        } else if(state == GAME_OVER){
            dc2 = (Graphics2D)field_game.GetDC();
            dc2.setColor(Color.WHITE);
            dc2.fillRect(0, 0, getWidth(), getHeight());
            dc2.drawImage(ilogo, (getWidth() - 300)/2, 100, 300, 200, null);
            ftmp = dc2.getFont();
            dc2.drawString("Íà÷àòü çàíîâî íàæìèòå ENTER", 220, 400);
            dc2.setFont(font);
            dc2.setColor(Color.RED);
            dc2.drawString("ÂÛ ÏÐÎÈÃÐÀËÈ ÁÈÒÂÓ !!!", 160, 370);
            dc2.setFont(ftmp);
            dc.drawImage(field_game.Handle(), 0, 29, null);
            dc.setColor(Color.WHITE);
            dc.fillRect(0, 512, getWidth(), 14);
        } else if(state == GAME_WINNER){
```

```
            dc2 = (Graphics2D)field_game.GetDC();
            dc2.setColor(Color.WHITE);
            dc2.fillRect(0, 0, getWidth(), getHeight());
            dc2.drawImage(ilogo, (getWidth() - 300)/2, 100, 300, 200, null);
            ftmp = dc2.getFont();
            dc2.drawString("Íà÷àòü çàíîâî íàæìèòå ENTER", 220, 400);
            dc2.setFont(font);
            dc2.setColor(Color.GREEN);
            dc2.drawString("* * * ÏÎÇÄÐÀÂËßÞ ÂÛ ÏÎÁÅÄÈËÈ * * *", 95, 370);
            dc2.setFont(ftmp);
            dc.drawImage(field_game.Handle(), 0, 29, null);
            dc.setColor(Color.WHITE);
            dc.fillRect(0, 512, getWidth(), 14);
        }
        dc.dispose();
    }

    // îáðàáîò÷èê íàæàòèå êëàâèøè ìûøè
    public void game_keyPressed(KeyEvent e) {
        if(state == GAME_PLAY)
            user_game.KeyPress(e.getKeyCode(), field_game);
        else if((state == GAME_START) || (state == GAME_OVER) || (state ==
GAME_WINNER)){
            if(e.getKeyCode() == KeyEvent.VK_ENTER){
                level   = 0;
                ilevel  = true;
```

```
            lvltime = System.currentTimeMillis();
            field_game.StartLevel(level, user_game);
            state   = GAME_PLAY;
          }
       }
    }

    public void game_keyReleased(KeyEvent e) {
       if(state == GAME_PLAY)
          user_game.KeyUp(e.getKeyCode());
    }
}



class DlgFrame_keyAdapter extends KeyAdapter {
    private DlgFrame adaptee;
    DlgFrame_keyAdapter(DlgFrame adaptee) {
       this.adaptee = adaptee;
    }

    public void keyPressed(KeyEvent e) {
       adaptee.game_keyPressed(e);
    }
```

```
    public void keyReleased(KeyEvent e){
        adaptee.game_keyReleased(e);
    }
}
```