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 „Впровадження програмних модулів ігрового додатку на базі Android”

Виконав: студент групи ІПЗ-16д _____ Д.Г. Лаптєва
(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

_____Лаптєва Дар`я Геннадіївна _____
(прізвище, ім’я, по батькові)

1.Тема роботи Впровадження програмних модулів ігрового додатку на базі Android. _____
_____

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 мобільного ігрового додатку._____
_____
_____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Вступ. Аналітичний огляд, з висвітленням наступних питань: Історія створення комп'ютерних ігор, огляд ігрових жанрів. Основна частина, в якої висвітлити: збір вимог та проектування, етап даталогічного проектування. 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 _____ Д.Г. Лаптєва_____
(підпис) (ініціали і прізвище)

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

ЛИСТ ПОГОДЖЕННЯ І ОЦІНЮВАННЯ

дипломної роботи студента гр. ІПЗ-16д Лаптєва Д.Г.

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: 40 сторінок основного тексту, 33 сторінок додатків, 7 рисунків, 10 використаних джерел.

Метою випускної кваліфікаційної роботи є проектування та розробка ігрового додатку.

У роботі було досліджено впровадження програмних модулів ігрового додатку на базі Android. На основі досліджень було створений мобільний додаток «Мозаїка».

Система реалізована відповідно всім вимогам технічного завдання. Зроблено детальний опис процесу розробки системи.

ЗМІСТ

ВСТУП

Актуальність досліджень. Людська потреба бути завжди в курсі справ виводять інформаційні технології на пряму лінію по створенню все нових девайсів і гаджетів . Незручність експлуатації комп'ютерів і ноутбуків зумовлює появу різних міні-комп'ютерів, смартфонів і комунікаторів, в основі яких лежить все та ж операційна система. Лідируючі позиції на сьогоднішній день займають платформи Android і iPhone . Але ці платформи можуть працювати повноцінно тільки за однієї умови - якщо була для них здійснено розробку мобільних додатків

Android є однією з найновіших розробок серед операційних систем. Дана операційна система призначена для широкого кола мобільних пристроїв. Операційна система Android встановлюється найчастіше на комунікатори, а так само на планшетні комп'ютер , смартбуки , і на нетбук . Варто відзначити, що розробники системи Android регулярно експериментують. Найчастіше ці експерименти пов'язані з додаванням в різні технічні пристрої. Є годинник на основі системи Android , телеприставки і багато іншої подібної техніки.

Розробник цієї системи - компанія " Android Inc . ". Пізніше ця компанія була придбана компанією" Гугл "(Google). В даний час розробками і розвитком систем на базі Android займається компанія" Open Handset Alliance "." Open Handset Alliance "включає не тільки Гугл , а й Motorola , HTC , Intel , Samsung і багато інших гіганти в області виробництва техніки. Операційна система Android заснована на базі Linux . Але містить не всі розробки останньої. Пов'язано це з використанням віртуальної машини " Делвік ". Саме на ній відбувається робота всього програмного обладнання. Розробники не варті на місці. Система Android постійно вдосконалюється і впроваджується в усі новітні види техніки. в даний час в індустрії відеоігор напрямок мобільних платформ отримало активний розвиток. Найбільш

популярні ігри по аудиторії значно випереджають гри з ПК і ігрових консолей, а за даними компанії SuperData [1] на ринку мобільних ігор дохід за 2016 рік склав $ 40.6 млрд. при загальних для індустрії 91 $ млрд.

Розробка ігрових додатків для мобільних платформ є найменш складний, витратний і ризикований спосіб для розробника отримати досвід в розробці ігор. Мобільні ігри за структурою значно простіше аналогів з інших платформ, а ціна розміщення готового продукту в таких системах цифрового поширення, як Google Play [2], найчастіше обмежується разовими внеском реєстрації розробника.

Мобільні ігри так само надають розробникові безліч способів розвитку проекту, в тому числі поширення вже отримала певну популярність гри на персональні комп'ютери. Таке розширення можна побачити на прикладах ігор Braveland [3] або Warhammer 40,000: Deathwatch [4], які в даний час розміщені в системі цифрового поширення Steam [5].

За вищенаведеним доводам було прийнято рішення почати розробку мобільного ігрового програми. Система Android була обрана через її максимальної серед мобільних платформ простоті і відкритості, але це не обмежує проект і в процесі подальшого розвитку також можливе поширення на iOS і Windows Phone .

Об’єкт досліджень: Впровадження програмних модулів ігрового додатку на базі Android.

Предмет досліджень: програмні модулі на базі Android.

Завдання дослідження: 1) Проектування та розробка ігрового додатку.

2) Дослідження переметної області.

3)Огляд аналогів.

Методологічна та теоретична основа дослідження: Дослідження програмних модулів ігрового додатку на базі Android.

Методи дослідження: Огляд і аналіз існуючих рішень в області розробок на базі Android.

Практичне значення отриманих результатів: Створення ігрового додатку на базі Android,

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Історія створення комп'ютерних ігор

Історія комп'ютерних ігор бере свій початок задовго до загального буму комп'ютеризації. Перші іграшки були створені програмістами-любителями для себе і своїх колег, в чисто пізнавальних цілях. Природно, створювалися вони на машинах, ще не надто графічними засобами виведення інформації, простіше кажучи - ці ігри були текстовими. Коли стали доступні графічні растрові дисплеї. Н а таких екранах стало можливим зображати не тільки графіки, креслення і таблиці, а й тих, що билися чоловічків, космічні кораблі, якихось абсолютно неймовірних чортиків. Перші іграшки з графічним інтерфейсом це " Digger ", " Shumus ", " Paratrooper ", " Bad Street " і " Saboteur ". Прогрес не стоїть на місці, і ось на зміну домінуючим монохромним і чотирибарвним CGA-дисплеям прийшов EGA з його 16 квітами. Н а базі 16 кольорів вже можна було серйозно працювати, і світ побачив знаменитих " Pirates !" і " Civilization " Сіда Мейера, "LOOM" і " Monkey Island " від LucasArts , " Prince of Persia ". Однак, бум почався лише з появою дисплеїв VGA, дозволяючи видавати зображення або у високій роздільній здатності (640x480) з 16 квітами, або - і це було принциповим нововведенням - з 256 квітами, хоча і в дозволі 320x200. До того ж почали отримувати широке поширення різні звукові плати, такі як Sound Blaster і Adlib , перші накопичувачі CD-ROM, поки ще одне або двохшвидкісні і притому непристойно дорогі, але все ж виконують свою основну функцію

1.2 Огляд ігрових жанрів

3D-шутери (3D Shooter). В іграх даного типу гравець, як правило, діючи поодинці, повинен знищувати ворогів за допомогою холодної та вогнепальної зброї, для досягнення певних цілей на даному рівні, зазвичай, після досягнення заданих цілей гравець, переходить на наступний рівень.

Шутери від першого (First person shooter, FPS). У шутерах від першої особи гравець не бачить персонажа з боку - він спостерігає за тим, що відбувається від імені персонажа і спостерігається гравцем картина збігається з тим, що «бачить» персонаж. Приклади : Doom, Quake, U nreal Tournament, Call of Duty.

Шутери від третього особи (Third person shooter, TPS). У шутерах від третьої особи гравець бачить персонажа з боку з фіксованою (зазвичай зі спини) або довільної точки зору. У ряді ігор реалізована можливість перемикання першої / третя особа і фіксована / довільна камера. Приклади: Tomb Raider , Max Payne , MDK, Grand Theft Auto .

«Криваві» шутери (gore). Суть таких ігор полягає в знищенні величезної кількості ворогів, найчастіше комп'ютерних ботів, великими групами що насуваються на гравця. При цьому гравець повинен мати простір для маневру. Приклади: Serious Sam , Pain killer , Will Rock , Left 4 Dead .

Тактичні шутери . Принципова відмінність від класичних шутерів полягає в тому, що персонаж не зображує героя-одинака, а діє в складі команди. В тактичному шутері зазвичай відтворюється діяльність загонів - взаємодія між бійцями, маневрування і вибір напрямку атаки, підбір команди і її озброєння. В одиночному режимі ці можливості реалізуються ботами, в мережевому - через взаємодію живих гравців. Приклади : Battlefield, Counter-Strike: Condition Zero, Counter-Strike, Counter-Strike Source, Star Wars: Battlefront, Delta Force, Star Wars: Republic Commando, Operation Flashpoint: Cold War Crisis.

Файтинг . Геймплей складається виключно з поєдинків двох і більше супротивників із застосуванням рукопашного бою. Приклади : Mortal Kombat , Street Fighter, Tekken, Virtua Fighter, Dead or Alive, Guilty Gear X

Побий їх усіх. Різновид файтингов , бій в яких відбувається за межами арени і з безліччю супротивників одночасно. Приклади : Oni, Enter the Matrix, The Matrix: Path of Neo

Слешер . Ігри з видом від третьої особи, основною частиною ігрового процесу, в яких є фехтувальні поєдинки із застосуванням холодної та іншої зброї. Приклади : Blade of Darkness, Rune, Enclave, Jedi Academy, God of War, Devil May Cry

Аркада (Arcade). Ігри, в яких гравцеві доводиться діяти швидко, покладаючись в першу чергу на свої рефлекси і реакцію. Ігровий процес простий і не змінюється протягом гри. Аркади характеризуються розвиненою системою бонусів: нарахування очок, поступово відкриваються елементи гри і т. Д. Термін «аркада» по відношенню до комп'ютерних ігор виник за часів ігрових автоматів, які встановлювалися в торгових галереях (arcades). Ігри на них були простими в освоєнні (щоб залучити побільше грають). Згодом ці ігри перекочували на ігрові приставки (консолі) і до сих пір я вляются основним жанром на них.

СТЕЛС - екшен (Stealth-action). Ігри, в яких мається на увазі не битися з більшістю зустрінутих противників, а всіляко уникати можливого контакту з ними, попутно виконуючи поставлені завдання. Стелс-елементи (наприклад, можливість виглядати з-за рогу, притулившись до стіни) часто зустрічаються в іграх різних жанрів. Приклади : Assassin's Creed, Thief, Metal Gear Solid, Tom Clancy's Splinter Cell, Hitman , Manhunt.

Технічні. Гра-симуляція. За допомогою комп'ютера як можна більш повно імітується фізична поведінка і управління будь-яким складним об'єктом технічної системи (наприклад: бойовим винищувачем, автомобілем і т. Д.). Якщо аркадні ігри прагнуть розважити гравця за допомогою різних неможливих явищ, трюків і гостроти сюжету, то головний критерій якості технічних стимуляторів - повнота і реалістичність моделювання його об'єкта

(автомобіля, повітряного судна і т. Д.). Приклади : FlightGear , Live for Speed, Microsoft Flight Simulator, X-Plane, Lock On: Сучасна бойова авіація , DCS Ка -50: Чорна акула , Orbiter, Trainz

Аркадні. Спрощена версія технічних стимуляторів, нерідко з альтернативної фізикою. Принципова відмінність від власне аркад - наявність хоч і спрощеної, але все ж фізичної моделі. Найчастіше з подібною фізикою робляться стимулятори космічних кораблів і автомобілів. Приклади : I л -2 Штурмовик , X-Wing, TIE- Fighter, Wing Commander, Need for Speed, Повний привід , Burnout Paradise

Спортивні (Sport game). Інша назва - « Спортс » (від англ. Sport simulator - стимулятор спорту). Як і випливає з назви - імітація будь-якої спортивної гри, найбільшого поширення набули імітації футболу, хокею, баскетболу, тенісу і гольфу, боулінгу та більярду.

Спортивний менеджер. Спортивний менеджер - різновид спортивного стимулятора. Відмінною особливістю є те, що при симуляції гравець спостерігає безпосередньо за ігровим процесом і може впливати на хід матчу в режимі онлайн , а при менеджменті настройки тактики, стратегії, трансферів і фінансових операцій вибираються заздалегідь, і гравець переглядає результати вже після матчу. У спортивному менеджері гравець виступає в ролі керуючого власної спортивної командою (спортсменом). Завданням грає стає не тільки перемога в матчах, але і грамотне і успішне управління інфраструктурою свого клубу. Приклади: 11х11 - футбольний онлайн - менеджер (Nekki), Королі льоду - хокейний онлайн - менеджер (Nekki)

Економічні (Business simulation game). Економічні стимулятори, нерідко відносяться також до жанру стратегій, присвячені відображенню економічних, ринкових процесів - частіше за все мова йде про підприємництво; метою гравця, керівного якимось підприємством, є

отримання віртуальної прибутку. У «чистих» економічних стимуляторах відсутні елементи будівництва; гравець повинен управляти вже існуючим комерційним підприємством; ринкові процеси і поведінку конкурентів щодо наближені до реальності. Приклади: Capitalism , Hollywood Mogul , Віртономіка .

Стратегії (Strategy). Гра, що вимагають планування і вироблення певної стратегії для досягнення якоїсь конкретної мети, наприклад, перемоги у військовій операції. Гравець управляє не одним персонажем, а цілим підрозділом, підприємством або навіть всесвіту.

Стратегії реального часу (Real-time strategy , RTS). У цих стратегіях гравці проводять свої дії одночасно. Вони з'явилися дещо пізніше покрокових, першою стала популярною грою цього жанру була Dune II (1992), сюжет якої заснований на однойменному творі Френка Герберта. Уже тоді сформувалися загальні принципи стратегій в реальному часі. Більшість «класичних» стратегій в реальному часі припускають наступний ігровий процес: збір деяких ресурсів; будівництво та зміцнення бази або табору; створення на цій базі бойових одиниць (найм солдат, будівництво техніки); об'єднання їх в групи, штурм і знищення цими групами ворожої бази. Приклади : Command & Con quer, Warcraft , Age of Empires.

Покрокові стратегії (Turn-based strategy, TBS). Покрокові стратегії - ігри, в яких гравці роблять свої дії по черзі. Покрокові стратегії з'явилися раніше RTS і відрізняються значно більшою різноманітністю. Поділ ігрового процесу на ходи відриває його від реального життя і позбавляє гру динамізму, в результаті чого ці ігри не так популярні, як стратегії в реальному часі. З іншого боку, в TBS у гравця набагато більше часу на роздуми, під час здійснення ходу його ніщо не квапить, що дає можливість набагато більш глибокого і ґрунтовного планування. Приклади : X-COM, Civilization, Galactic Civilizations, Heroes of Might and Magic, Disciples.

Варгейми (Wargame). У Варгейми , на відміну від інших видів стратегій, гравець не повинен створювати армію, його мета - перемогти супротивника в бою, використовуючи ті сили, які у нього є в розпорядженні на початку бою. У варгеймах , як правило, робиться наголос на автентичність, реалістичність і історичність. Приклади : Правда про дев'ятої роті , Steel Panthers, Panzer General, Warhammer , Squad B attles.

Глобальні стратегії (Government simulation game), С тратегіі , в яких гравець управляє державою. У його руках не тільки війна і економіка, але і науковий прогрес, освоєння нових земель і дипломатія. У деяких з них поряд з глобальною картою існують місцеві, на яких проходять тактичні битви. Приклади : Master of Orion, Civilization, Medieval: Total War, Rome: Total War, Medieval II: Total War, Empire: Total War, Napoleon: Total War , Shogun 2: Total War.

Стимулятор бога (God game), Стратегічні ігри, в яких гравцеві доведеться виступати в ролі «бога » - таку собі надприродною сутності, що піклується про цілий невеликому народі. Подібні ігри характеризує, як правило, непрямий контроль над окремими ігровими персонажами - ними керує комп'ютер, а роль гравця визначається «надприродним» втручанням в їхнє життя, будівництві будівель, підтримці оптимального стану підопічного суспільства і тому подібному. Багато стимулятори бога не ставлять перед гравцем ніяких конкретних завдань, надаючи йому можливість вільно і необмежено розвивати підопічних суспільство. Приклади: Black & White , Dungeon Keeper , Spore .

Пригоди, адвенчури або квести (Adventure , Quest). Гра-розповідь в якій керований гравцем герой просувається по сюжету і взаємодіє з ігровим світом за допомогою застосування предметів, спілкування з іншими персонажами і рішення логічних задач. Приклади : Space Quest, Ларрі в вихідному костюмі , Siberia, Myst , The Longest Journey

Текстові квести . Спочатку, через малого поширення графічних пристроїв відображення і нестачі ресурсів (пам'яті і потужності процесора), все квести були текстовими. Пізніше цей жанр був названий текстовим квестом . Відмінність від графічних квестів полягає в тому, що гравець взаємодіє з ігровим світом за допомогою командного рядка і інформація про світ виводиться у вигляді текстів і малюнків з друкованих знаків. Приклади: Supe rhero league of Hoboken і т. П.

Графічні квести . Перші графічні квести з'явилися ще на початку 1980-х. Приклади: Королівський квест , Космічний квест , Ларрі у вихідному костюмі.

Головоломки. Крім збору предметів та їх використання, в цих іграх вирішуються різні головоломки, в тій чи іншій мірі інтерновані в сюжет, і на рішення головоломок робиться основний упор. Зазвичай може знадобитися збірка різних, нерідко абсурдних як по вигляду, так і по функціональності, змов. Приклади: Myst , Neverhood .

Екшн-адвенчури (action-adventure). Головним чином заснований на реакції і рефлексах гравця, але є і елементи класичних квестів - предмети і взаємодія з навколишнім оточенням. Приклади: Legend of Zelda , Resident Evil і інших ігор жанру жахів (які теж можуть вважатися екшн-адвенчур).

Стимулятор романтичних відносин (romantic adventure). Так само відомі як романтичні пригоди. За геймплею деяк е з них близькі до RPG, інші - до адвенчур .

Музичні гри. У музичних іграх геймплей будується на взаємодію гравця з музикою. Жанр же може бути будь-який, від головоломок до ритм ігор. Приклади : SingStar , Mad Maestr o !, Wii Music, Boom Boom Rocket

Ритмічні гри . Піджанр музичних ігор, який останнім часом набирає величезну популярність, завдяки аркадні автоматам і спеціальним

контролерам, які часто використовуються в подібних ігор. Але через відсутність першого і дорожнечі другого жанр не поширений в Росії. Основною ідеєю є правильно натискання кнопок показаних на екрані під ритм музики. Приклади : Guitar Hero, Rock Band, Серія Dance Dance Revolution, Hatsune Miku , Project DIVA, Frets on Fire

Рольові ігри (Role playing game, RPG). У головного героя (героїв) і інших персонажів і ворогів (частіше в меншій мірі) присутня деяка кількість параметрів (умінь, характеристик, навичок) які визначають їх силу і здібності. Зазвичай, головна характеристика персонажів і ворогів це рівень, який визначає загальну силу персонажа і визначає доступні навички і предмети екіпіровки. Всі ці параметри треба вдосконалювати шляхом вбивства інших персонажів і ворогів, виконанням завдань цих самих навичок.

Присутній пророблений і великий світ, сильна сюжетна лінія, розгалужені діалоги з різними варіантами відповідей, безліч різних персонажів зі своїми цілями і характерами. Приклади: Mass Effect , Deus Ex , Gothic .

Hack'n'Slash RPG. Рольова гра з акцентом на винищення ворогів, збирання найбільш потужних речей, прокачування характеристик. Зазвичай мають спрощений світ і сюжет, невелика кількість діалогів і варіантів рішення квестів та інших завдань. Приклади: Diablo , Dungeon Siege , Titan Quest .

True RPG (також просто RPG або CRPG). Рольова гра з великою кількістю діалогів, свободою у виборі шляхів вирішення різних завдань, пророблених світом і сюжетом. Приклади: Fallout , Planescape : Torment , Baldur 's Gate , Dragon Age : Orig ins .

JRPG - (Японська РПГ). Рольова гра з пророблених світом і діалогами, але з меншою свободою вибору, найчастіше має дуже захоплюючий, але лінійний сюжет, відсутність вибору в прокачуванні характеристик

персонажа, а також дуже опрацьованих і красивих персонажів. Гра чимось схожа на інтерактивну книгу. Найчастіше створюється японськими розробниками, і існує дуже мало ігор такого роду, створених за межами Японії. Приклади: Final Fantasy , Dragon Quest .

Тактичні RPG. Жанр рольових ігор, який є сумішшю з покрокової стратегією. Гравець управляє невеликою групою воїнів, хоча в деяких тактичних RPG їх число може доходити до декількох десятків. Перші тактичні RPG з'явилися на консолях в Японії. Сьогодні, однак, багато західних і комп'ютерних тактичних рольових ігор. Приклади : Fallout Tactics, Jagged Alliance, Silent Storm, X-COM (UFO), Горький 17, Горький 18.

Головоломки, логічні, пазли (Puzzle). У некомп'ютерною головоломці роль арбітра, що стежить за дотриманням правил, грає або сам гравець (пасьянс), або деякий механічний пристрій (кубик Рубика). З появою комп'ютерів можливості головоломок розширилися, так як написати комп'ютерну програму простіше, ніж сконструювати механічний пристрій. Головоломки, як правило, не вимагають реакції від гравця (проте багато хто веде відлік часу, витраченого на рішення). Приклади: Сапер, Sokoban , Полювання на лисиць, Portal

Традиційні та настільні (Traditional , Board). Комп'ютерна реалізація настільних ігор. Приклади: Шахи, карти, шашки, «М \онополія», серія ігор Warhammer .

Interactive Fiction (дослівно - інтерактивна література, IF; текстові квести ; adventure - « адвенчури ») - жанр комп'ютерних ігор, в якому спілкування з гравцем здійснюється за допомогою текстової інформації. Розвиток цього жанру, в зв'язку з низьким вимогою до ресурсів, почалося досить давно, і не припинилося навіть з появою графічних ігор. Існують два види інтерфейсу - інтерфейс з введенням тексту з клавіатури або інтерфейс у

вигляді меню, де гравець вибирає дію з декількох запропонованих (CYOA - Choose Your Own Adventure).

Ігри в псевдографіка. Різновид текстових ігор, в яких є графічна картинка у вигляді мозаїки, побудована з ASCI I-символів. Приклади: Roguelike .

Поодинокі (singleplayer). Розраховані на гру в поодинці, проти комп'ютера.

Розраховані на багато користувачів (multiplayer). Розраховані на гру кількох людей (зазвичай до 32) по локальній мережі, модему або Інтернету.

Розраховані на багато користувачів на одному комп'ютері (Hot Seat , Splitscreen). На сучасних персональних комп'ютерах бувають рідко, проте часто зустрічаються на старих ПК і приставках. Hot seat - гра по черзі на одному комп'ютері. У режимі splitscreen екран поділяється на частини (зазвичай дві, буває чотири), кожен з гравців грає на своїй частині.

Розраховані на багато користувачів оффлайн-ігри (PBEM). Деякі жанри (спортивні менеджери, покрокові стратегії і т. Д.) Можуть працювати в такому форматі: гравці роблять ходи і відсилають результат через веб або електронну пошту. Незалежно від методу зв'язку (Фідонет , електронна пошта, веб ...), ці ігри мають такі особливості:

Масові онлайнові (Massively multiplayer online; англ . Massively multiplayer online RPG). Масові гри по Інтернету. Велика частина не іграбельна в оффлайн . Найбільш часто зустрічаються жанри - настільні та рольові ігри (MMORPG). Серед них розрізняють також браузерні ігри (ігри, які не потребують установки будь-якого клієнта) з яких виділяється жанр MUD - текстові онлайнові ігри. Приклади : Легенда Спадщина драконів , EVE Online, Lineage 2, World of Warcraft , Warhammer Online: Age of

Reckoning, Ragnarok Online, Ігровий , My Lands, RF Online, OG ame , Everquest 2, Steel Giants.

1.3 Конструктори ігор

Для створення сучасної гри потрібно засвоїти величезну кількість інформації з самих різних областей знань. У мережі можна знайти величезну кількість програм по створенню ігор, а в книжкових магазинах досить літератури по створенню ігор без програмування. В обох випадках мова йде про так званих конструкторах ігор.

3D Game Maker - створення ігор різних жанрів (аркадні і гоночні ігри, action і adventure), не потрібно знання мов програмування, повністю тривимірні сцени (близько 100 різних сцен), Понад 500 3D об'єктів (машини, механізми, вороги на чолі з жахливим " босом ", зброя, бонуси, перешкоди та ін.), необмежену кількість сюжетів, сценаріїв і ігрових опцій (більше 12 мільярдів!), понад 320 звукових ефектів і більше 100 мелодій, додавання власних 3D моделей, звуків, музики .І спользование власного зображення в грі, додаткові об'єкти в Інтернеті, підтримка парної гри, редактор розміщення об'єктів на сцені, редактор рівнів, використання різних пристроїв управління (миша, джойстик).

3D Game Studio - це повноцінний ігровий конструктор, ціла система для створення тривимірних ігор. Для створення простих 3D / 2D-ігор не потрібно знати ніяких мов програмування, але для більш складних ігор є скриптовой мову Lite-C .

Ігрові світи робляться досить просто: ви за допомогою перетягування мишею розставляєте на екрані свої об'єкти, ландшафти, персонажів. Далі ставите їм поведінку та інше. У комплекті поставляються додаткові утиліти для розробки гри: створення об'єктів і їх анімації. На завершальному етапі потрібно налаштувати всі параметри камери, об'єктів і скомпілювати гру. 3DGS пропонує хороші графічні можливості, підтримує простий в освоєнні

мову скриптів , є можливість підключення зовнішніх систем на C ++, C # або Delphi . Якщо вирішили зробити серйозний проект в 3D, то даний інструмент вам допоможе досягти якщо не найкращих результатів, то дуже високих. Дуже популярна програма у гейм-девелоперов низького і середнього достатку. У Gamestudio A8 інтегрований фізичний движок PhysX , Enet для роботи в мережі, динамічні тіні, необмежений розмір ландшафту з 200000 об'єктів на рівень, аудіо движок OpenAL , портали для рендеринга , використовуються графічні прискорювачі для розрахунків анімації і карт і т.д. Gamestudio сумісний з версіями A 6 і A7 (крім фізики).

MUGEN - це безкоштовний або умовно-безкоштовний 2D файтинг і зручний движок, який був розроблений компанією Elecbyte . Написаний движок на «C» з бібліотеками « Allegro ». MUGEN працює під управлінням DOS, GNU / Linux і Microsoft Windows . Для тих, кого цікавить створення свого 2D-файтинга. Це не програма, а по суті гра, в якій можна змінювати все, абсолютно все: вигляд персонажів, локації, музику, звуки, штучний інтелект і все інше. Є гра, а ви що хочете, то і міняйте. Можна зробити свою гру з усіма власними компонентами, а можна зробити якийсь сплав, куди напхати всілякі компоненти і персонажів звідусіль. У даної програми є багато фанатів, які створили безліч різноманітних доповнень вигляді локацій, персонажів. Перенесли безліч бійців з безлічі знаменитих ігор даного жанру і не тільки. І тепер можна пограти, наприклад, бійцями гри Mortal Kombat проти бійців з Street Fighter і з'ясувати хто-ж кого сильніше.

3d Action Maker - це, як стверджує автор, перший пострадянський движок для створення тривимірних шутерів без програмування. Перевага 3DAM від інших движків полягає в тому, що він досить легкий в освоєнні і безкоштовний. Звичайно, на ньому можна створювати ігри без програмування, але для просунутих користувачів, для тих, хто хоче створити професійну гру, є можливість скористатися додатковим скриптинга .

З технічних особливостей можна відзначити тільки те, що є можливість створення 3D-ігор, підключення графічних 256-кольорових картинок для заставок. Покращена 3D-графіка за підтримки DirectX . Можливість додавати свої об'єкти. Готові проекти можна компілювати в виконуваний файл exe . Додатково до 3DAM поставляються деякі утиліти в допомогу.

3D Rad - дозволяє створити онлайн ігри, створювати ігри для web-сторінок. Конструктор не обмежує розробника ігор в жанрі, можна створювати RPG, FPS, TPS і т.д. Ігри можна робити як в 3D, так і в 2D. Пропонується AI, можливість скриптування для створення і досягнення складних задач в грі.

Ігри можна створювати будь-якого типу і жанру в 3D просторі. Досить нескладний інструмент, але до інтерфейсу програми потрібно звикати. З недавнього часу на даному конструкторі вважається найзручніше робити гонки з хорошим 3D якістю

Racer - це вільний гоночний 3D-движок-гра-конструктор для кроссплатформенной (Windows , Linux , Mac OS X) розробки (для некомерційного використання). Движок використовує сучасну професійну автомобільну фізику. Чудова реалістична графіка, яка додає максимум реалізму! Використовуються OpenGL-технології. Для повної зручності вашої розробки в пакет включені редактори траси, моделей, фізики автомобілів тощо.

3D Engine Virtools - підходить для створення 3D-ігор практично будь-якого жанру і складності: шутери , квести , симулятори, стратегії, розраховані на багато користувачів ігри і т.д. Прототипи можна створювати буквально в лічені хвилини. При розробці великих проектів додаток легко розділяється на модулі, що дозволяє поєднувати роботу групи програмістів, дизайнерів, моделлер і.т.д. Virtools 3D engine містить всі необхідні компоненти для створення як казуальних, так і великих ігор. На базі

технологій Virtools створений цілий ряд успішних комп'ютерних ігор для різних платформ.

Unity - це професійний мультиплатформенний ігровий движок і інтегрований потужний ігровий редактор для полегшення створення ігор.

Цей движок з себе представляє щось на зразок конструктора ігор, так як є відмінна середу редагування зі зручним для користувача інтерфейсом, що дозволяє створювати гру візуально! Редактор простий і інтуїтивно зрозумілий, налаштовується. Властивості об'єктів налаштовуються кількома кліками миші, щоб призначити текстури, звук, поведінка, скрипти і т.д. Візуальне перетягування і маніпуляція з об'єктами. Клонування об'єктів - іноді потрібно перенести або розмножити складні ігрові об'єкти. Ігрові об'єкти повністю керовані і налаштовуються, інтерактивні.

3D-RPG Builder - один з небагатьох конструкторів, який дозволяє створювати в ньому 3D гри жанру RPG. 3D RPG на ньому виходять досить-таки скромні по графіку (за нинішніми мірками), але при наявності певних умінь дану обставину можна обіграти дизайном. Створення ігор в даному конструкторі ґрунтується на редагуванні рівнів: розставляєте об'єкти і привласнюєте їм властивості. Так само є підтримка скриптового мови LUA, який досить зручний і простий в освоєнні завдяки гарній документованості на багатьох мовах світу, в тому числі і російською. Проте, скрипт застосовується в основному для того, щоб призначати події в грі. Для створення нормальної 3D RPG в 3D RPG Builder'е все ж деякі е написання скриптів потрібно.

Blender - програма для створення анімованної комп'ютерної графіки та виробництва відео. Це по суті 3D-редактор, який підтримується багатьма користувачами завдяки відкритому вихідному коду. Але на даному 3D-редакторі, завдяки вбудованому ігровому движку " Blender Game Engine ", можна створювати і гри будь-якого типу, спрямованості і жанру. Як

виявляється, ігри на ньому можна створювати не вдаючись до скриптовими мови, який, до речі, є (Python) для деяких потреб. Стверджують, що ігри на ньому можна створювати за п'ять хвилин. Є довідки і керівництва як по користуванню, так і по створенню ігор. Можна ще розраховувати на велику ком'юніті користувачів Blender'a , до яких можна звернутися за допомогою. 3D-графіка має всі наворотами OpenGL , різні високі дозволу, підтримка практично всіх популярних аудіо та відео форматів.

RPG Maker - це перша програма для створення ігор жанру jRPG і, мабуть, найпопулярніша. Її люблять багато творці ігор без програмування, і у неї є своя армія шанувальників, в тому числі і серед російськомовного контингенту. І тому досить легко знайти приклади ігор, підручники, статті з даного конструктору. Створення ігор в цьому конструкторі нескладне. Потрібно мінімальне знання місцевого скриптового мови. На демонстраційних прикладах можна зрозуміти суть роботи всіх геймплейних особливостей, як вони виконані за допомогою скриптових команд. При створенні гри у вас є можливість підключати шрифти, звуки (mid , wma , ogg , mp3 і CD-Audio), графіку, відео (wav , avi) та ін. У конструкторі можна реалізовувати всі функції жанру jRPG : битви, прокачування, діалоги , квести, магію та ін.

XtremeWorlds (XW) - це потужний безкоштовний движок-конструктор 2D MMORPG. У XW ви можете зробити свою власну MMORPG без знань програмування в лічені хвилини! Але якщо ви бажаєте більше можливостей, ніж вам доступні без програмування, ви можете використовувати спеціальну скриптовими систему, яка розширює можливості движка і дозволяє вам здійснити деякі нестандартні рішення

Quest3D - це готове комплексне рішення для створення 3D інтерактивних додатків. Головною особливістю програми є можливість створювати досить складні 3D програми без глибоких знань програмування,

будь то 3D архітектурні інтерактивні демонстрації, стимулятори, ігри, додатки віртуальної реальності або що-небудь ще. Розробка програм в Quest3d відбувається не за допомогою класичних мов типу C ++, а за допомогою спеціальних блок-схем, які дозволяють зробити код більш гнучким і наочним, а також значно прискорюють процес розробки додатків. Quest3d дозволяє імпортувати 3D моделі в різних форматах (X, 3DS, Collada), матеріали, текстури (BMP, JPG, PNG і.т.д.) і звуки (MP3, OGG). Крім того, в програмі передбачені редактори параметрів імпортованих ресурсів (матеріалів, геометричних об'єктів, анімованих персонажів і.т.д.), а також повноцінний редактор рівнів, за допомогою якого можна компонувати 3D сцени.

Dungeon Craft - це вільний конструктор ігор на PC в стилі First Person RPG. DC є проектом з відкритим вихідним кодом, розповсюджується безкоштовно. Простий, зручний і, головне, якщо ви фанат ігор Wizardry і Eye of Beholder , то він для вас! Даний конструктор ігор призначається для розробки ігор з ігровий системі AD & D. З Dungeon Craft ви зможете створювати гри типу SSI 's Forgotten Realms Unlimited Adventures (FRUA). Однак в Dungeon Craft можливостей більше, ніж в згаданих хітах жанру, тому можливо створення більш оригінальних ігор жанру.

Blade3D спрощує розробку ігор, роблячи цю діяльність доступною найширшої аудиторії. При цьому має потужні можливостями і функціями. Він містить всі необхідні підсистеми для розробки ігор - візуальний редактор, партікли , скрипти , ландшафтний движок, пост ефекти, систему матеріалів, кісткову анімацію, II, фізику, звук і т.д.

Розробнику дається можливість менше возитися з кодуванням, даючи можливість зосередитися на проектуванні геймплея . Можливість використовувати готові скріптові шаблони. Простий у використанні. Створення гри відбувається в більшій мірі візуалізованно. Підтримує

популярні формати файлів. Тулсет Blade3D дозволяє створювати 3D світи в реальному часі, об'єднуючи їх. Все редагується і настроюється. Всі інструменти пов'язані і вам нема чого переривати роботу - повна функціональність і сумісність. Деякі з плюсів: інтерактивний редактор сцен, C # скрипт едітор , аніматор, підтримка мульти-камери , кроссплатформенная розробка, імпорт 30 форматів графіки, HLSL редактор, імпорт більш ніж 15 форматів моделей, імпорт численних звукових і відео форматів, документації, велика спільнота.

Scrolling Game Development Kit (GameDeve) - це вільний (ліцензія GPL) інструментарій для розробки 2D-ігор аркадного жанру. Програма розроблена на стандарті OpenGL 1.2. Для успішної роботи з програмою необхідний деякий досвід і час на вивчення особливостей програми. Але ця програма може використовуватися і новачком, і навіть без уміння програмування. Досвідчений розробник теж може використовувати даний конструктор для створення своїх ігор, якщо хоче створювати гри перетягуючи об'єкти і клацаючи кнопки, що прискорює процес розробки. Для досвідчених користувачів є скриптовой мова (за подобою Visual Basic), який значно розширює можливості. Сам движок написаний на мові програмування C # з використанням Microsoft .NET Framework 2.0.

РОЗДІЛ 2 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

2.1 Огляд аналогів

В цьому розділі роздивляється приклади ігор, схожих з проектом за жанром, моделі поширення або інших особливостей. При цьому варто враховувати, що в сфері відео ігрової індустрії конкуренція має свої особливості. Користувачі схильні не вибирати один проект з певного списку шляхом порівняння характеристик, як це часто буває з неігровими програмними продуктами. Найчастіше гравці намагаються безліч існуючих проектів, а від якості конкретного проекту залежить кількість часу і грошових коштів, які гравець готовий в цей проект вкласти.

Огляд гри Bird Paradise

Жанр: Аркада

Мітки: Три в ряд, гра без кеша

Bird Paradise є зовсім недавнім проектом від студії Ezjoy , яка вже раніше випускала інші ігри на Android найрізноманітніших жанрів. Варто відзначити, що не так давно ця компанія створила такий незаперечний хіт як Garden Mania , який отримав найпозитивніші і захоплені відгуки від більшості гравців. Після невеликого проміжку часу всі фанати дізналися, що студія не збирається зупинятися на досягнутому, а планує домогтися небувалих висот в створенні ігор обраних студією жанрів. Попередній досвід створення ігор дуже допоміг компанії не тільки в швидкості розробки нового проекту, але і в поліпшенні його сильних якостей і сторін. У новій грі Ezjoy пропонують нам взяти під свій захист кумедних птахів і допомогти їм захистити своє потомство. Варто зауважити, що в плані основний ігровий механіки не відбулося ніяких серйозних змін. Ми так-же повинні об'єднувати птахів однакового кольору в ланцюжки по три і більше елементів, щоб очистити від них ігровий екран, набрати очки і заробити різні бонуси. Однак ,

успішність цієї гри криється далеко не в звичності її ігровий механіки а в тому, з яким підходом і любов'ю розробники створюють і малюють цю гру. У наш час ця компанія є не менше, ніж майстрами в розробці ігор даного жанру. Але як уже говорилося студія Ezjoy не планує зупинятися. Вона постійно удосконалює свої ігрові проекти, щоб зробити їх ще яскравіше і успішніше на ринку. Відмінною рисою даної гри є її дивовижна, красива графіка. Також в грі присутня велика кількість барвистою анімації і різних ефектів. Не можна не відзначити, що в даній грі є і великий вибір рівнів, доступних для проходження гравцеві, безліч різних бонусів, елементів розвитку і інших дрібниць. Bird Paradise однозначно повинна сподобається більшості гравців, яким припадають до смаку аркадні головоломки.

Графіка в грі дійсно чудова і не дарма людям подобається те, як ця гра виглядає на екранах їх смартфонів. Гра з перших секунд запустилася легко, цілісність картинки добре відчувається в будь-яких її елементах і відразу стає помітно, що і сама тематика гри обрана досить здорово і підходить для цього жанру. Птахи намальовані і анімовані дійсно красиво і яскраво. Анімації присутні навіть в головному меню. Птахи виглядають дуже живими. Рухаються на гілці, моргають очима і іноді засипають, що в подальшому виявилося буде відображено і в самому геймплеї гри. Емоційний стиль у цієї гри позитивний, поєднання кольорів теж вельми тепле і симпатичне, що привертає до цієї гри інших людей.

Зі звуком ніяких проблем теж не виявилося. Музика дуже підходяща ігрового процесу на екрані під час гри. Є можливість вимкнути або включити звук в будь-який момент в головному меню в розділі «настройки». Зауважу, що для звуку та музики немає повзунків гучності, тому їх регулювання цілком і повністю зв’язано на кнопки вашого телефону. В налаштуваннях можна тільки вимкнути музику, або повернути її назад. Це спрощення так само дуже здорово виглядає і допомагає швидко підводити звук гри під різні життєві ситуації, витрачаючи при цьому мінімум зусиль.

Рис. 2.1 – Загальний вигляд гри

Головне меню в грі зроблено максимально просто, інтуїтивно і доступно. Всі елементи інтерфейсу в процесі гри відображаються коректно в потрібних місцях екрану. У грі немає великого достатку налаштувань, зате в тих, що є всюди відчувається просто та, дружелюбність, зрозумілість.

Огляд гри Magic Blender

Жанр: Аркада

Мітки: Три в ряд, гра без кеша , таймкіллер

У грі Magic Blender нам дадуть можливість стати чарівником, який вміє варити і створювати різні чарівні зілля. Незважаючи на те, що дана гра виглядає цілком зазвичай в жанрі «три в ряд», в ній є додаткові відмінності в геймплеї і природно, не схожий ігровий дизайн і стиль. Якщо в більшості ігор, подібного жанру, є ліміт на кількість пересувань різних елементів, то тут з'явився ще й ліміт часу, який дається на проходження даного рівня. Часу дається трохи, тому гравцеві доведеться діяти чітко і швидко, щоб встигнути

і перейти на наступний рівень. Рішення повинні бути не тільки блискавичними, але і ефективними. Якщо ви довго не зможете знайти рішення, і ефективних дій виявиться мало, то доведеться починати рівень спочатку. За рахунок цього гра дуже запалює азарт в гравцеві, а також відмінно підходить для коротанія часу. Час буде постійно підганяти гравця, що з часом позитивно позначиться на його реакції і швидкості прийняття ігрових рішень, що вельми чудово для ігор таких жанрів. Гра не вимагає інтернет підключення, тому можна вільно грати, навіть якщо інтернету немає.

Основні особливості гри:

- Приємна барвиста графіка

- Безліч різних рівнів

- Можливість отримати окуляри, після закінчення матчу

Рис 2.2 – Вигляд гри

Як тільки ми запускаємо гру, відразу опиняємося на завантажувальному екрані. Спочатку здається, що гра зроблена досить-таки просто, однак це тільки перше враження. Гра використовує движок для створення ігор - Unity , який представляє з себе величезний набір різних інструментів для створення ігор в 2D або в 3D. Розмір гри становить близько 100 мб ., Що після попередньої гри здається великою цифрою (попередня важила всього 16 мб .)

Головне меню гри так само максимально просто і без зайвих кнопок для натискань, але при цьому містить безліч підказок для гравців, які тільки-

тільки починаються знайомити з цим жанром. Це допомагає їм швидше зрозуміти суть того, що відбувається на екрані і навчитися керувати ігровим процесом. Саме меню зроблено в 3D орієнтації. Персонаж-чарівник виглядає вельми живим і об'ємним , за рахунок третього виміру. У грі немає як таких налаштувань, які викликалися б з головного меню. В принципі, таке рішення теж виглядає непогано. Тому, що в попередній грі, яку ми розглянули, інших налаштувань крім звуку не було. У будь-який момент звук можна дуже просто налаштувати клавішами вгору і вниз на вашому мобільному.

Огляд гри Ice Age: Arctic Blast

Категорія: Аркада

Мітки: Три в ряд, гра без кеша

Ice Age : Arctic Blast - чергова гра в знаменитому жанрі три в ряд, де вам належить відправитися в дуже захоплюючу подорож в місці з дивовижною трійцею - Сідом, Менні , Дієго і знаменитої по серії мультфільмів білкою - Скретом . Всю гру вас будуть супроводжувати знайомі персонажі, а в самій грі найголовнішим завданням буде просте складання дорогоцінних каменів в ряди і їх знищення, яке допомагає гравцеві очистити ігрове простір і заповнити його новими. Подорожуйте в компанії друзів через крижану долину, Діно море і інші незвичайні місця, набирайте достатню кількість очок, щоб розблокувати надалі всі рівні, на яких відкриються можливості вирішити всі ігрові головоломки. Шлях від каменів, ви будете заробляти багато очок, а також отримувати море додаткових бонусів, які спеціально приховані для вас. Гра дуже захоплююча. Все, що відокремлює вас від грандіозного пригоди - це вибір улюбленого героя з мультфільму, з яким в подальшому вам доведеться долати різні перешкоди і прокладати шлях до перемоги з першого до останнього рівня.

Завантажувальний екран гри відразу показує нам знайомих по серії мультфільмів персонажів. Мабуть, це є відмінною рисою даної гри. Тому, що

в інших іграх, які ми розглянули, всі персонажі були намальованими окремо для цього проекту і раніше ніде не використовувалися.

Рис. 2.3 – Вигляд гри

Головне меню гри так само виглядає стильно, в очі відразу впадає 3D формат зображень і картинок. Для самих персонажів в головному меню зробили цікаві і повноцінні 3д сценки та анімації, де вони переміщаються, рухаються і дивляться один на одного. Опрацювання таких сценок явно зайняла у розробників пристойну кількість часу, проте виглядають вони дійсно доречно, жваво. У багатьох гравців після перегляду цих сценок, швидше за все з'явиться навіть посмішка на обличчі, пов'язана зі спогадами з самих мультфільмів, адже персонажі незважаючи на перетворення анітрохи не змінилися характерами і залишилися вірними тим прототипам, якими їх

спочатку зробили. Ще одним плюсом є навіть невелика кількість озвучення персонажів голосами, які використовувалися і в самому мультфільмі. Це зближує гравця з самою грою.

В налаштуваннях тут знаходиться набагато більшу кількість опцій, ніж було в інших проектах. Пов'язано це з тим, що дану гру можна підключати до соціальної мережі Facebook для того, щоб можна було змагатися з друзями і близькими людьми при проходженні різних типів рівнів. Решта ж настройки гри в цілому повторюють аналогічні, однак тут присутні «повзунки» гучності, які я спочатку не очікував побачити після кількох прикладів, де їх повністю перенесли на кнопки твого мобільного. Тут це занесемо, але можна додатково регулювати і вручну. Плюс це чи ні, сказати складно.

2 .2 Висновки по предметної області

За вищенаведеним прикладам можна укласти наступне:

- користувачі не прагнуть до інновацій в жанрі, проста двовимірна графіка і звичні ігрові механіки не роблять негативного ефекту на успішність гри, найбільше значення для гравця мають стилістика, музичний супровід і різноманітність ігрового процесу;

- високу важливість має розвиток проекту після випуску, регулярні оновлення необхідні для підтримки інтересу вже наявної аудиторії;

- необхідно максимально акуратне застосування умовно-безкоштовної моделі монетизації, продаж внутр. ігрового контенту не повинна торкатися ігровий баланс.

2.3 Розробка програми для платформи Android .

JDK - безкоштовно поширюваний компанією OracleCorporation комплект розробника (далі SDK) на мові Java , що включає в себе компілятор Java (javac), стандартні бібліотеки класів Java , приклади, документацію, різні утиліти і виконавчу систему Java (JRE). До складу JDK не входить інтегроване середовище розробки на Java , тому розробник, що використовує тільки JDK, змушений використовувати зовнішній текстовий редактор і компілювати свої програми, використовуючи утиліти командного рядка.

Android SDK - Android SDK включає в себе інструменти, необхідні для розробки Android-додатків. Комплект засобів розробки, який дозволяє фахівцям з програмного забезпечення створювати додатки для певного пакету програм, програмного забезпечення базових засобів розробки, апаратної платформи, комп'ютерної системи, ігрових консолей, операційних систем і інших платформ.

Eclipse - вільна інтегроване середовище розробки модульних кроссплатформенних додатків. Розвивається і підтримується EclipseFoundation . На даний момент є найбільш зручним засобом розробки для Android .

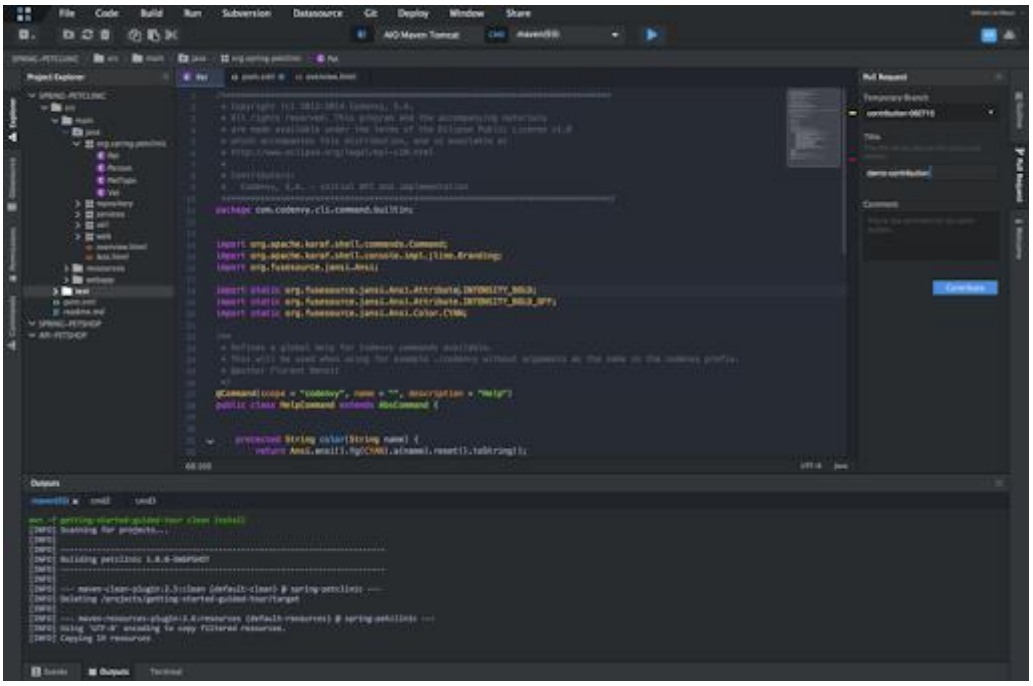

Рис. 2.4 – Вигляд інтерфейсу

ADT плагін - за замовчуванням Eclipse не особливо годиться для розробки Android-додатків. ADT - плагін , який налаштовує середу розробки для використання Android SDK і додає можливість зручної розробки. Плагін досить легко встановлює в середу Eclipse .

SDKmanager - тут ми вибираємо необхідні нам компоненти для розробки додатків, а саме: платформа (API), утиліти (оптимізатори для певних пристроїв, набір готових прикладів і т.д.), ТУЛС (пряма налагодження на ANDROID-пристрої, інтеграція додаткових можливостей і т.д.).

З вибором я зупинився на SDK платформи версії 2.3.3. Саме дана платформа пропонує весь необхідний набір методів необхідний для більш зручної розробки додатків. Так само, ANDROID-пристроїв з версією нижче 2.3 ми вже не знайдемо. Відповідно, додатки створені за допомогою даного SDK можна буде використовувати на будь-якому мобільному пристрої з ОС Android .

AVD (android virtual device) - це емулятор Android - смартфона, на який Eclipse зможе встановлювати, створені нами додатки, і запускати їх там.

Таким чином, в сукупності, ми отримуємо середу розробки з необхідним набором плагінів і віртуальної машини для налагодження і перевірки скомпільованих додатків. Далі можна приступати безпосередньо до розробки додатків.

2.4 Особливості і специфіка розробки

Android це унікальна операційна система. Розробник додатків повинен знати її особливості та нюанси для отримання хорошого результату. Існують деякі труднощі, які потрібно враховувати при розробці. Перерахую їх коротко:

- Додаток вимагає для установки в два рази (або навіть в чотири) більше місця, ніж оригінальний розмір програми;

- Швидкість роботи з файлами на вбудованої флеш-карті падає в десятки разів при зменшенні вільного місця;

- Кожен процес може використовувати до 16 Мб (інколи 24 Мб) оперативної пам'яті.

Android заснований на Linux . Між додатком і ядром лежить шар API і шар бібліотек на нативному коді. Додаток виконується на віртуальній машині Java (Dalvik Virtual Machine).

В Android можна запускати багато додатків. Але одне з них є головним і займає екран. Від поточної програми можна перейти до попереднього або запустити нове. Це схоже на браузер з історією переглядів.

Кожен екран для користувача інтерфейсу представлений класом Activity в коді. Різні Activity містяться в процесах. Activity може навіть жити довше процесу. Activity може бути припинена і запущена знову з збережені всієї потрібної інформації.

Android використовує спеціальний механізм опису дій заснований на Intent . Коли потрібно виконати дію (зробити дзвінок, надіслати листа, показати вікно), викликається Intent .

Також Android містить сервіси подібні демонам в Linux для виконання потрібних дій у фоновому режимі (наприклад, програвання музики). Для обміну даними між додатками використовуються Content prov iders (провайдери вмісту).

Для даної роботи були використані провайдер даних про місцезнаходження і стан в просторі користувацького пристрою.

2. 5 Вибір середовища розробки

Були розглянуті середовища розробки Eclipse [9], IntelliJ IDEA [10] і Android Studio [11]. Всі вони надають повні можливості по розробці додатків на мові Java останньої версії. Android Studio має вбудовані засоби по налаштуванню комплекту засобів розробки Android SDK і емулятор пристроїв з операційною системою Android . В Eclipse і IntelliJ IDEA настройка Android SDK і емуляція пристроїв на ОС Android здійснюється за допомогою офіційних полігонів .

З цих трьох середовищ розробки Android Studio має наступні переваги в контексті розробки для ОС Android :

- початкова орієнтованість IDE з ОС Android ;

- більшу відносно інших IDE кількість навчальних матеріалів і документації по розробці для мобільних платформ;

- простота настройки Android SDK і емуляторів.

З недоліків Android Studio можна відзначити тільки високе споживання системних ресурсів середовищем розробки.

2. 6 Вибір фреймворка

На сучасному етапі розвитку інформаційних технологій практично жоден серйозний проект не розробляється на одній мові програмування в його базової комплектації без залучення сторонніх ресурсів, такі як графічні та математичні бібліотеки, візуальні редактори або кошти кроссплатформенной розробки.

У разі ігрового програми першочерговості важливість представляє собою вибір фреймворку , так званого «движка», який відповідає в основному за життєві цикли внутрішніх процесів і логіку графічної частини програми.

Часто необдуманий вибір фреймворку в контексті конкретної гри може привести до серйозних проблем в оптимізації додатки, а значить і до безлічі негативних відгуків від користувачів і низьку популярність гри в системах цифрової поширення. Відповідний же фреймворк навпаки забезпечує розробнику максимальні швидкість і якість реалізації.

Був обраний фреймворк libGDX . Цей движок розробляється з 2014 року, в графічній частині заснований на OpenGL і орієнтований на мову Java , підтримуючи при цьому C і C ++. Ключовою особливістю мови є модульна система забезпечення кроссплатформенной розробки: основний код пишеться в одному загальному модулі, а спеціальні використовуються для складання проекту на різних платформах.

Таким чином з введенням мінімальної кількості коду тільки в спеціальних модулях можна забезпечити запуск і коректну роботу програми на наступних системах: Windows , Android , Mac OS, iOS , Linux , і браузери з підтримкою WebGL .

Незважаючи на те, що проект спочатку був розрахований на платформу Android , можливість забезпечення платформ без значних витрат часу є вкрай важливою, так як відкриває широкі можливості для розвитку гри після релізу і публікації першої версії.

Багато розробники найпопулярніших проектів на певному етапі були змушені переписувати більшу частину проекту на іншому фреймворку або навіть мові програмування для поширення гри на лише одну нову платформу. Найбільш часто такими платформами відбуваються між Android і iOS .

Так само серед переваг libGDX можна відзначити наступні якості:

- відкрита ліцензія Apache 2.0;

- висока продуктивність графіки при відносно низькій складності її розробки. Це особливо помітно в 2D, що ідеально підходить для розроблюваного проекту;

- наявність великого обсягу документації та навчальних матеріалів, в тому числі російськомовних.

Варто також відзначити, що libGDX рідко використовується в дійсно великих ігрових проектах. Пояснюється це в першу чергу його новизною, а так неповної реалізацією деяких складних 3D-алгоритмів. Фреймворк більше орієнтований на незалежні розробки незалежних студій і розробників, створення перших проектів і введення в сферу розробки відеоігор.

Додатки для Android в своїй роботі використовує вікно (аналогічно Windows), проте в даній системі вищевказані вікна носять іншу назву - Activity . Як і в Windows , кожне вікно має свій життєвий цикл і свої особливості. При створенні нового вікна викликається метод onCreate (), при розробці даний метод визначається і в ньому відбувається ініціалізація програми та його компонентів. Далі викликаються методи onStart () і onResume (). Обидва методи викликаються перед відображенням вікна при його створенні, або відновленням і (при перемиканні з іншої програми, при розгортанні згорнутого програми та тп). При згортанні викликаються методи onPause () і onStop (). При закритті програми і вікна викликається onDestory (), в даному методі можна зберегти призначені для користувача дані і параметри. Повний опис та послідовність виклику методів можна знайти на офіційному сайті.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ГРИ

3.1 Гра на Xamarin.Android

Гра мозаїка для Android. Исходник гри для Android написаний на мові C # на платформі Xamarin.Android . Исходник представляє гру Мозаїка, де картинки можна пересувати і таким чином становити красиві візерунки. Xamarin надбудова для середовища .NET дозволяє створювати Android додатки на мові C # з використанням всіх можливостей популярного мови. Xamarin.Android забезпечує отримати повний доступ до нативному (рідному) Android SDK без будь-яких обмежень. Програмування додатків на мові C # за допомогою Xamarin інтуїтивно зрозуміло і дозволяє швидко перебудовуватися в програмуванні найрізноманітніших десктопових і веб додатків для Windows , Lunix , Android , iOS .

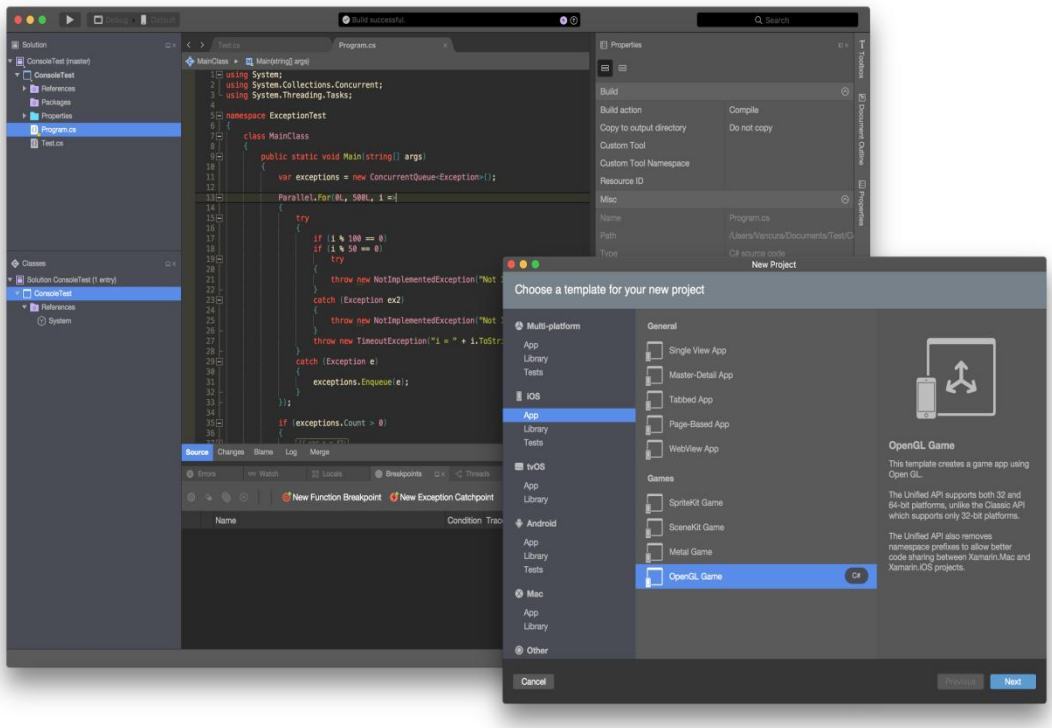

Рис. 3.1 - Xamarin Studio

3.2 Інтерфейс на Android.RelativeLayout

Компонування елементів інтерфейсу гри Мозаїка базується на макеті RelativeLayout , зручному контейнері для розміщення елементів і груп елементів. Використання RelativeLayout надає можливість створювати позиції елементів в точних одиницях. Контейнер хоч і носить назву відносний, але дозволяє використовувати абсолютні координати для позиціонування елементів. В даному исходнике це і було використано. Макет RelativeLayout створюється в XML дизайнера, а дочірні елементи, картинки упаковані в ImageView , створюються і додаються на поле макета програмним способом. Перед показом ігрового поля короткочасно демонструється заставка.

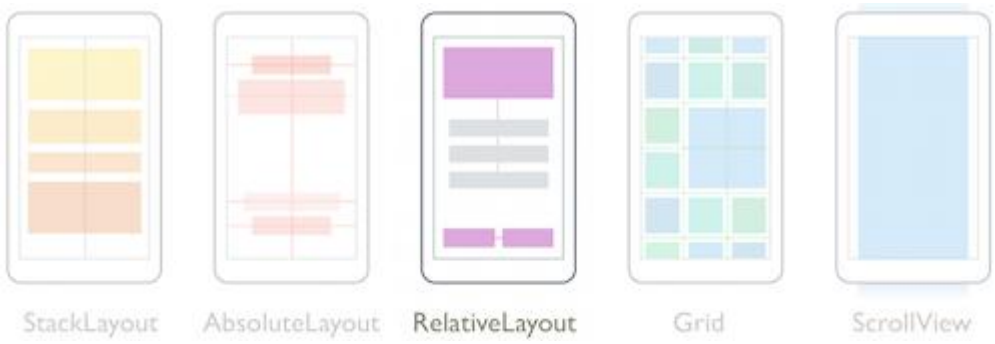

Рис. 3.2 – Макет RelativeLayout

3.3 Активні елементи на Android.ImageView

Для управління частинками Мозаїки, файли картинок упаковані в контейнер Android.ImageView . Така оболонка для зображень має багато корисних властивостей. За допомогою ImageView можна позиціонувати зображення на макеті по піксельним координатами, масштабувати по горизонталі і вертикалі, додавати колірну маску для вкладеного зображення та ін. Методи успадковані від батьківського класу View SetY (...), SetX (...) розміщують контейнери картинок по осях X і Y в будь-якому місці

батьківської макета RelativeLayout в піксельних одиницях. Використовуючи просторові координати, можна ефективно управляти положенням картинок на екрані смартфона або планшета. Уявлення ImageView створюються і додаються в батьківський контейнер програмним способом. Картинки для наповнення ImageView завантажуються з ресурсів Drawable і далі програмно кожна в своє уявлення.

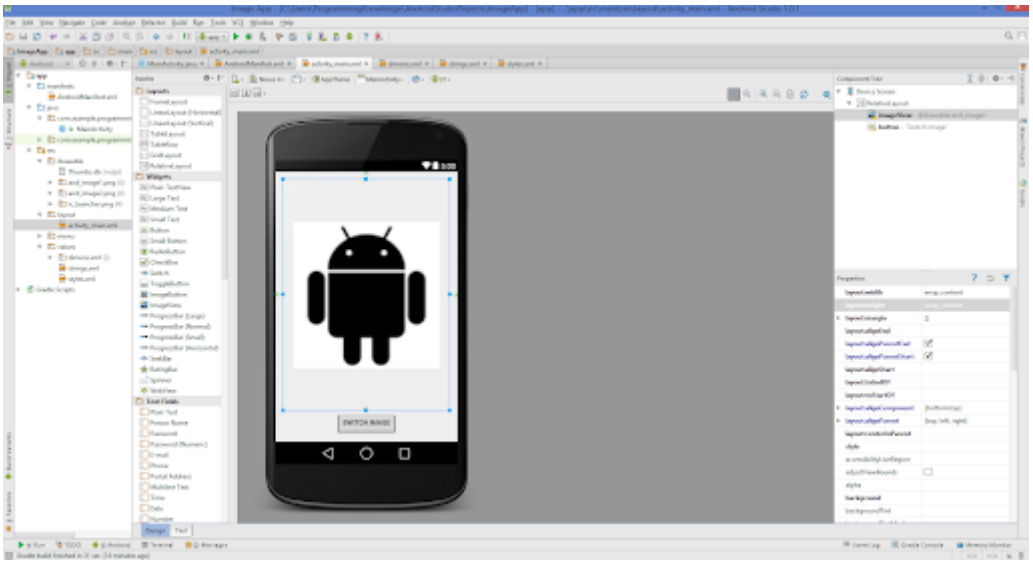

Рис 3.3 - Android.ImageView

3.4 Отримання розмірів RelativeLayout

Отримати програмно ширину і висоту контейнера RelativeLayout під час створення або відновлення неможливо. Протягом роботи OnCreate (...), OnStart (..), OnResume (...) створюються тільки об'єкти візуальних класів, при цьому всі методи і властивості вимірювання на виході видають нульові значення. Фактично вони ще не «знають» як розташує їх на екрані батьківський контейнер. Надійне отримання ширини і висоти макета для розміщення елементів ImageView гарантовано після повного створення дерева уявлень. На жаль, подібно програмування в Windows , в Android немає подій OnShow (), де можна перед показом елемента (вікна) дізнатися його розміри. Але вихід завжди є: можна запросити розміри цікавить нас об'єкта

RelativeLayout в відкладеної задачі, яка отримує доступ до інтересуемого розмірами після завершення підготовки призначеного для користувача інтерфейсу.

3.5 Підгонка розмірів ImageView на весь екран

Розміри уявлень ImageView , а значить і картинок в них, вираховуються автоматично при візуалізації MainActivity (Activity це одиниця екрану з призначеним для користувача інтерфейсом, Activity в додатку може бути кілька). Зображення завантажені в ImageView квадратні, а фізичні розміри вираховуються в пікселях. За основу розрахунків, в даному исходнике гри, прийнята ширина головного макета в вертикальному положенні. По вертикалі картинки розташовуються на екрані до максимального заповнення. При завантаженні на смартфоні, планшеті або іншому Android пристрої з різними розмірами дисплеїв візуально виходить гармонійне заповнення різнокольоровими квадратиками.

3.6 Просторові координати

Координати позицій і розмірність елементів ImageView обчислюються після створення дерева уявлень MainActivity і зберігаються в масиві прямокутників RectF RectPositionImages . Зберігання координат окремо від контейнерів картинок дозволяє контролювати їх положення на екрані дисплея і коригувати в разі потреби. Просторовими координати названі, тому що вони не прив'язані до елементів макета і тільки віртуально ділять простір головного контейнера на осередки. Просторові координати обчислюються на основі кількості осередків по ширині екрану. Елементи ImageView розміщуються на позиціях в контейнері RelativeLayout обчислюються в пікселях. При створенні програми створюється ігрове поле з розмірами вічок вирахуваних відповідно до розміру екрана даного Android пристрою.

3.7 Подія дотику Touch

Для взаємодії користувача з грою Мозаїка застосовується подія Touch. Ця подія є реагування на дотик пальцем або стилусом до сенсорного екрану. Дотик до об'єкта ImageView викликає видима зміна розмірів і прозорості картинки. Така корисна зворотний зв'язок дає інформацію гравцеві Мозаїки про те, що він дійсно вибрав необхідний квадратик і може його переміщати в бажане місце для створення задуманого узору. В процесі переміщення також є зворотний сигнал про те, що квадратик знаходиться над потрібним місцем і його можна відпустити. При цьому колір переміщуваної картинки змінюється якщо вона знаходиться на допустимій відстані від цільового місця розміщення. Після відпускання картинки (подсобитіе MotionEventActions.Up) активна картинка обмінюється координатами позиції з нижележащей картинкою.

3.8 Використання властивості View.Tag

Властивість Tag класу Android.View призначене для зберігання інформації безпосередньо в об'єкті класу. ImageView є спадкоємцем класу View і відповідно теж має дане властивість. Властивість зручно тим, що в ньому можна зберігати будь-який об'єкт. У исходнике гри Мозаїка ImageView.Tag використовується для зберігання інформації про поточну і нової позиції власника властивості при виконанні події Touch (...). Для цього створено клас Positions в якому зберігатися координатна інформація для необхідних переміщень кольорових квадратиків.

ВИСНОВОК

Android є однією з найновіших розробок серед операційних систем. Дана операційна система призначена для широкого кола мобільних пристроїв. Операційна система Android встановлюється найчастіше на комунікатори, а так само на планшетні комп'ютер , смартбуки , і на нетбук . Варто відзначити, що розробники системи Android регулярно експериментують.

У роботі було досліджено впровадження програмних модулів ігрового додатку на базі Android.
На основі досліджень було створений мобільний додаток «Мозаїка».

СПИСОК ЛІТЕРАТУРИ

1. SuperData Research [Електронний ресурс]: офіц. сайт. - Режим доступу: https://www.superdataresearch.com/.

2. Google Play [Електронний ресурс]: офіц. сайт. - Режим доступу: https://play.google.com/store.

3. Додаток Braveland в Google Play [Ел. ресурс]. Реж. доступу: https://play.google.com/store/apps/details?id=com.tortugateam.braveland.v2&hl=ru.

4. Warhammer 40,000: Deathwatch: Tyranid Invasion [Електронний ресурс]: офіц. сайт. - Режим доступу: http://rodeogames.co.uk/deathwatch.

5. Steam [Електронний ресурс]: офіц. сайт. - Режим доступу: store.steampowered.com.

6. Додаток Battle for Wesnoth в Google Play [Ел. ресурс]. Режим доступу: https://play.google.com/store/apps/details?id=it.alessandropira.wesnoth112&hl=ru.

7. Стандартна громадська ліцензія GNU (GPL) [Електронний ресурс]: офіц. сайт. - Режим доступу: https://www.gnu.org/licenses/gpl-3.0.ru.html.

8. Додаток King's Bounty: Legions в Google Play [Електронний ресурс]. Режим доступу: https://play.google.com/store/apps/details?id=com.kranx.kbl.

9. Eclipse [Ел. ресурс]: офіц. сайт. - Режим доступу: https://eclipse.org/.

10. IntelliJ IDEA [Електронний ресурс]: офіц. сайт. - Режим доступу: https://www.jetbrains.com/idea/.

11. Android Studio [Електронний ресурс]: офіц. сайт. - Режим доступу: https://developer.android.com/studio/index.html.

12. Creative Commons License [Електронний ресурс]: офіц. сайт. - Режим доступу: https://creativecommons.org/licenses/by/3.0.

13. Додаток Rogue's Tale в Steam [Електронний ресурс]. Режим доступу: http://store.steampowered.com/app/265990/Rogues_Tale/?l=russian..

ДОДАТОК А

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;
using Android.App;
// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
[assembly: AssemblyTitle("AndroidMosaic")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("InterestPrograms.RU")]
[assembly: AssemblyProduct("AndroidMosaic")]
[assembly: AssemblyCopyright("Copyright InterestPrograms.RU ©  2020")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]
[assembly: ComVisible(false)]
// Version information for an assembly consists of the following four values:
//
//      Major Version
//      Minor Version
//      Build Number
//      Revision
//
```

```
// You can specify all the values or you can default the Build and Revision
Numbers
// by using the '*' as shown below:
// [assembly: AssemblyVersion("1.0.*")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
namespace AndroidMosaic
{
   [Activity(Label = "@string/app_name", Theme = "@style/AppTheme",
MainLauncher = true, ScreenOrientation =
Android.Content.PM.ScreenOrientation.Portrait)]
   public class MainActivity : AppCompatActivity
   {
      // Переменные для фиксации положения картинки
      // при перемещении пальцем.
      private float DeltaX;
      private float DeltaY;

      // Координаты мест расположения картинок в пространстве экрана.
      private RectF[] RectPositionImages;
      // Количество позиций по горизонтали (наименьшему размеру экрана).
      readonly private int NumberRectHorizontal = 9;
      protected override void OnCreate(Bundle savedInstanceState)
      {
         base.OnCreate(savedInstanceState);
         Xamarin.Essentials.Platform.Init(this, savedInstanceState);
```

```
        // Set our view from the "main" layout resource
        SetContentView(Resource.Layout.activity_main);

        RelativeLayout layoutMain =
this.FindViewById<RelativeLayout>(Resource.Id.LayoutMain);

        // Добавляем в поток интерфейса асинхронную задачу прорисовки
        // картинок только после получения действительных
        // размеров главного макета-контейнера.
        // Задача исполнится после готовности
        // пользовательского интерфейса.
        layoutMain.Post(() =>
            {
                ComputePos(layoutMain);

                ShufflePositions();

                InitImages(layoutMain);
            }
        );
       }
      // Вычисление позиций и размеров квадратиков
      void ComputePos(RelativeLayout layoutMain)
      {
        int widthLayout = layoutMain.Width;
        int heightLayout = layoutMain.Height;
```

```
        // Размер картинок высчитывается точно для горизонтали,
        // чтобы гармонично смотрелось по ширине.
        int widthRect = widthLayout / NumberRectHorizontal;
        // Картинка квадратная.
        int heightRect = widthRect;

        // Количество строк до заполнения контейнера по высоте.
        int numberRectVertical = heightLayout / heightRect;


        // Все позиции в пространстве.
        RectPositionImages = new RectF[NumberRectHorizontal *
numberRectVertical];


        // Расчет позиций в пространстве.
        int countPosY = 0;
        int countPosX = 0;
        for (int i = 0; i < RectPositionImages.Length; i++)
        {
           var rect = new RectF
           {
              Left = widthRect * countPosX,
              Top = widthRect * countPosY,
```

```
            };
            rect.Right = rect.Left + widthRect;
            rect.Bottom = rect.Top + heightRect;

            RectPositionImages[i] = rect;

            if (countPosX > 0 && countPosX % (NumberRectHorizontal - 1) == 0)
            {
                countPosX = -1;
                countPosY++;
            }
            countPosX++;
        }


    }


    // Перемешивание позиций
    private void ShufflePositions()
    {
        RelativeLayout layoutMain =
this.FindViewById<RelativeLayout>(Resource.Id.LayoutMain);

        // Перемешивание позиций для случайного начального узора.
        Random rnd = new Random();
```

```
            // Лаконичный (кракткий и ясный) перемешивания массива.
            // Сортировка элементов массива по псевдослучайным числам.
            RectPositionImages = RectPositionImages.OrderBy(x => rnd.Next()).ToArray();

            // Второй способ перемешивания (классический).
            //for (int i = 0; i < RectPositionImages.Length; i++)
            //{
            //    int r = rnd.Next(0, RectPositionImages.Length);
            //    RectF temp = RectPositionImages[i];
            //    RectPositionImages[i] = RectPositionImages[r];
            //    RectPositionImages[r] = temp;
            //}
        }

        private void InitImages(RelativeLayout layoutMain)
        {
            // Главный контейнер
            //RelativeLayout layoutMain = this.FindViewById<RelativeLayout>(Resource.Id.LayoutMain);

            // Удалим картинки которые были нужны только для визуализации в дизайнере.
            layoutMain.RemoveAllViews();
```

```
int[] picture = { Resource.Drawable.white, Resource.Drawable.green,
Resource.Drawable.blue,
    Resource.Drawable.red,  Resource.Drawable.white,
Resource.Drawable.white,
    Resource.Drawable.white,  Resource.Drawable.yellow,
Resource.Drawable.white};


int countPicture = 0;
for (int i = 0; i < RectPositionImages.Length; i++)
{
   var iv = new ImageView(this);
   iv.Touch += ImageView_Touch;
   layoutMain.AddView(iv);


   // Позиция картинки на экране.
   iv.SetX(RectPositionImages[i].Left);
   iv.SetY(RectPositionImages[i].Top);
   iv.LayoutParameters.Width = (int)RectPositionImages[i].Width();
   iv.LayoutParameters.Height = (int)RectPositionImages[i].Height();


   // Запомним назначенную позицию картинки в собственном классе.
```

```
        // Картинка сама будет носителем информации о своих позициях.
        var pos = new Positions
        {
            Position = RectPositionImages[i]
        };

        iv.Tag = pos;



        iv.SetImageResource(picture[countPicture]);

        countPicture++;
        if (countPicture == picture.Length) countPicture = 0;
    }

}



private void ImageView_Touch(object sender, View.TouchEventArgs e)
{

    // Координаты курсора
    MotionEvent motionEvent = e.Event;
```

```
            // Принимаем абсолютные координаты курсора.
            float cursorX = motionEvent.RawX;
            float cursorY = motionEvent.RawY;


            // Главный макет для доступа ко всем картинкам.
            RelativeLayout layoutMain =
this.FindViewById<RelativeLayout>(Resource.Id.LayoutMain);


            // Активная картинка
            ImageView ivSender = (ImageView)sender;


            // Позиции активной картинки в пространстве экрана
            Positions posSender = (Positions)ivSender.Tag;


            // Центр картинки сделаем посередине, чтобы её
            // было видно из-под пальца.
            ivSender.PivotX = ivSender.Width / 2;
            ivSender.PivotY = ivSender.Height / 2;
```

```
// Прикасаемся, т.е. нажимаем.
if (motionEvent.Action == MotionEventActions.Down)
{

    // Увеличиваем активную картинку для удобного
    // вождения пальцем.
    ivSender.ScaleX = 2.0f;
    ivSender.ScaleY = 2.0f;

    // Делаем картинку полупрозрачной, чтобы было видно
    // картинки под ней.
    ivSender.Alpha = 0.5f;

    // Поднимаем над всеми.
    ivSender.BringToFront();



    // Запоминаем данную позицию активной картинки.
    posSender.Position = new RectF
    {
        Left = ivSender.GetX(),
        Top = ivSender.GetY(),
        Right = ivSender.GetX() + ivSender.Width,
```

```
            Bottom = ivSender.GetY() + ivSender.Height


        };



        // Постоянная дельта на время вождения картинки.
        // Дельта это разница между координатным положением
        // картинки по данной оси и местом соприкосновения пальца.
        // Измеряется в абсолютных единицах.
        // Расстояние от места прикосновения до начала координат активной картинки.
        DeltaX = cursorX - ivSender.GetX();
        DeltaY = cursorY - ivSender.GetY();
    }



    // Перемещение активной картинки в поисках места для создания узора.
    if (motionEvent.Action == MotionEventActions.Move)
    {
        // Из координат курсора, во время перемещения, вычитаем расстояние от места прикосновения
        // до начала координат выбранной картинки.
        // Благодаря этому картинка относительно пальца будет неподвижна,
        // и будет двигаться точно под пальцем.
```

```
ivSender.SetX(cursorX - DeltaX);
ivSender.SetY(cursorY - DeltaY);

// Отлавливаем место возможного отпускания картинки.
for (int i = 0; i < RectPositionImages.Length; i++)
{
   if (ivSender.GetX() < (RectPositionImages[i].Left + 30) &&
      ivSender.GetX() > (RectPositionImages[i].Left - 30) &&
      ivSender.GetY() < (RectPositionImages[i].Top + 30) &&
      ivSender.GetY() > (RectPositionImages[i].Top - 30))
   {
      // Если место найдено просигналим изменением цвета
      // передвигаемой картинки.
      ivSender.SetColorFilter(Color.DarkViolet);

      // Устанавливаем флаг место найдено.
      posSender.isFoundPos = true;


      // Запоминаем новую позицию для активной картинки.
      posSender.newPos = RectPositionImages[i];

      break;
   }
```

```
            // Если отдалились от места возможного приземления
            // снимаем цветовую сигнализацию перемещаемой картикни.
            ivSender.ClearColorFilter();

            // Снимаем флаг обнаружения места приземления.
            posSender.isFoundPos = false;
        }
    }


    // Поднимаем палец.
    if (motionEvent.Action == MotionEventActions.Up)
    {

        // Возвращаем картинке нормальный масштаб.
        ivSender.ScaleX = 1;
        ivSender.ScaleY = 1;

        // Восстанавливаем непрозрачность.
        ivSender.Alpha = 1.0f;

        // Отпуская курсор принимаем новые координаты для данной картинки.
        // А картинка которая была уже на этой позиции отправляется на прежнее
        // место активной картинки.
```

```
            if (posSender.isFoundPos == true)
            {

                // Извлекаем информацию о нижележащей картинке под
перемещаемой.
                for (int img = 0; img < layoutMain.ChildCount; img++)
                {
                    ImageView imageView =
(ImageView)layoutMain.GetChildAt(img);

                    if (imageView.GetX() == posSender.newPos.Left &&
imageView.GetY() == posSender.newPos.Top)
                    {
                        // --- Если картинку нашли ---

                        // Поднимаем её над всеми картинками.
                        imageView.BringToFront();

                        // Перемещаем с анимацией картинку на старое место
передвигаемой картинки.
                        imageView.Animate().TranslationX(posSender.Position.Left);
                        imageView.Animate().TranslationY(posSender.Position.Top);

                        break;
                    }
                }
```

```
            // Активная картинка устанавливается на новое место.
            ivSender.Animate().TranslationX(posSender.newPos.Left);
            ivSender.Animate().TranslationY(posSender.newPos.Top);

            // Сбрасываем флаг найденного места.
            posSender.isFoundPos = false;

            // Восстанавливаем настоящий цвет.
            ivSender.ClearColorFilter();
        }
        else
        {
            // Если нижележащая картинка не найдена,
            // возвращаем активную картинку на прежнее место.
            ivSender.Animate().TranslationX(posSender.Position.Left);
            ivSender.Animate().TranslationY(posSender.Position.Top);

            // Восстанавливаем настоящий цвет.
            ivSender.ClearColorFilter();
        }
    }

}
```

```
        public override void OnRequestPermissionsResult(int requestCode, string[] permissions, [GeneratedEnum] Android.Content.PM.Permission[] grantResults)
        {
            Xamarin.Essentials.Platform.OnRequestPermissionsResult(requestCode, permissions, grantResults);

            base.OnRequestPermissionsResult(requestCode, permissions, grantResults);
        }


    }

    // Собственный класс обязательно наследуем от Java.Lang.Object
    // иначе свойству элементов Tag не сможем
    // присвоить объект класса.
    class Positions : Java.Lang.Object
    {
        // Текущая позиция
        public RectF Position;

        // Новая позиция
        public RectF newPos;
```

```
        // Флаг обнаружения места приземления.
        public bool isFoundPos;
    }
}
namespace AndroidMosaic
{
    [Activity(Label = "@string/app_name", Theme = "@style/AppTheme",
MainLauncher = true, ScreenOrientation =
Android.Content.PM.ScreenOrientation.Portrait)]
    public class MainActivity : AppCompatActivity
    {
        // Переменные для фиксации положения картинки
        // при перемещении пальцем.
        private float DeltaX;
        private float DeltaY;

        // Координаты мест расположения картинок в пространстве экрана.
        private RectF[] RectPositionImages;

        // Количество позиций по горизонтали (наименьшему размеру экрана).
        readonly private int NumberRectHorizontal = 9;

        protected override void OnCreate(Bundle savedInstanceState)
```

```
{
    base.OnCreate(savedInstanceState);
    Xamarin.Essentials.Platform.Init(this, savedInstanceState);
    // Set our view from the "main" layout resource
    SetContentView(Resource.Layout.activity_main);



    RelativeLayout layoutMain =
this.FindViewById<RelativeLayout>(Resource.Id.LayoutMain);

    // Добавляем в поток интерфейса асинхронную задачу прорисовки
    // картинок только после получения действительных
    // размеров главного макета-контейнера.
    // Задача исполнится после готовности
    // пользовательского интерфейса.
    layoutMain.Post(() =>
      {
        ComputePos(layoutMain);

        ShufflePositions();

        InitImages(layoutMain);
      }
    );
```

```
}

// Вычисление позиций и размеров квадратиков
void ComputePos(RelativeLayout layoutMain)
{
    int widthLayout = layoutMain.Width;
    int heightLayout = layoutMain.Height;


    // Размер картинок высчитывается точно для горизонтали,
    // чтобы гармонично смотрелось по ширине.
    int widthRect = widthLayout / NumberRectHorizontal;


    // Картинка квадратная.
    int heightRect = widthRect;


    // Количество строк до заполнения контейнера по высоте.
    int numberRectVertical = heightLayout / heightRect;
```

```
        // Все позиции в пространстве.
        RectPositionImages = new RectF[NumberRectHorizontal *
numberRectVertical];



        // Расчет позиций в пространстве.
        int countPosY = 0;
        int countPosX = 0;
        for (int i = 0; i < RectPositionImages.Length; i++)
        {
            var rect = new RectF
            {
                Left = widthRect * countPosX,
                Top = widthRect * countPosY,
            };
            rect.Right = rect.Left + widthRect;
            rect.Bottom = rect.Top + heightRect;

            RectPositionImages[i] = rect;

            if (countPosX > 0 && countPosX % (NumberRectHorizontal - 1) == 0)
            {
                countPosX = -1;
                countPosY++;
            }
            countPosX++;
```

```
        }


    }


    // Перемешивание позиций
    private void ShufflePositions()
    {
        RelativeLayout layoutMain =
this.FindViewById<RelativeLayout>(Resource.Id.LayoutMain);

        // Перемешивание позиций для случайного начального узора.
        Random rnd = new Random();

        // Лаконичный (кракткий и ясный) перемешивания массива.
        // Сортировка элементов массива по псевдослучайным числам.
        RectPositionImages = RectPositionImages.OrderBy(x =>
rnd.Next()).ToArray();

        // Второй способ перемешивания (классический).
        //for (int i = 0; i < RectPositionImages.Length; i++)
        //{
        //    int r = rnd.Next(0, RectPositionImages.Length);
        //    RectF temp = RectPositionImages[i];
        //    RectPositionImages[i] = RectPositionImages[r];
```

```
        //    RectPositionImages[r] = temp;
        //}
    }

    private void InitImages(RelativeLayout layoutMain)
    {
        // Главный контейнер
        //RelativeLayout layoutMain =
this.FindViewById<RelativeLayout>(Resource.Id.LayoutMain);

        // Удалим картинки которые были нужны только для визуализации в
дизайнере.
        layoutMain.RemoveAllViews();

        int[] picture = { Resource.Drawable.white, Resource.Drawable.green,
Resource.Drawable.blue,
            Resource.Drawable.red,  Resource.Drawable.white,
Resource.Drawable.white,
            Resource.Drawable.white,  Resource.Drawable.yellow,
Resource.Drawable.white};

        int countPicture = 0;
        for (int i = 0; i < RectPositionImages.Length; i++)
        {
```

```
var iv = new ImageView(this);
iv.Touch += ImageView_Touch;
layoutMain.AddView(iv);


// Позиция картинки на экране.
iv.SetX(RectPositionImages[i].Left);
iv.SetY(RectPositionImages[i].Top);
iv.LayoutParameters.Width = (int)RectPositionImages[i].Width();
iv.LayoutParameters.Height = (int)RectPositionImages[i].Height();


// Запомним назначенную позицию картинки в собственном классе.
// Картинка сама будет носителем информации о своих позициях.
var pos = new Positions
{
   Position = RectPositionImages[i]
};

iv.Tag = pos;


iv.SetImageResource(picture[countPicture]);

countPicture++;
```

```
            if (countPicture == picture.Length) countPicture = 0;
        }

    }


    private void ImageView_Touch(object sender, View.TouchEventArgs e)
    {

        // Координаты курсора
        MotionEvent motionEvent = e.Event;


        // Принимаем абсолютные координаты курсора.
        float cursorX = motionEvent.RawX;
        float cursorY = motionEvent.RawY;


        // Главный макет для доступа ко всем картинкам.
        RelativeLayout layoutMain =
this.FindViewById<RelativeLayout>(Resource.Id.LayoutMain);


        // Активная картинка
```

```
ImageView ivSender = (ImageView)sender;


// Позиции активной картинки в пространстве экрана
Positions posSender = (Positions)ivSender.Tag;


// Центр картинки сделаем посередине, чтобы её
// было видно из-под пальца.
ivSender.PivotX = ivSender.Width / 2;
ivSender.PivotY = ivSender.Height / 2;



// Прикасаемся, т.е. нажимаем.
if (motionEvent.Action == MotionEventActions.Down)
{

    // Увеличиваем активную картинку для удобного
    // вождения пальцем.
    ivSender.ScaleX = 2.0f;
    ivSender.ScaleY = 2.0f;

    // Делаем картинку полупрозрачной, чтобы было видно
    // картинки под ней.
```

```
            ivSender.Alpha = 0.5f;

            // Поднимаем над всеми.
            ivSender.BringToFront();



            // Запоминаем данную позицию активной картинки.
            posSender.Position = new RectF
            {
                Left = ivSender.GetX(),
                Top = ivSender.GetY(),
                Right = ivSender.GetX() + ivSender.Width,
                Bottom = ivSender.GetY() + ivSender.Height

            };



            // Постоянная дельта на время вождения картинки.
            // Дельта это разница между координатным положением
            // картинки по данной оси и местом соприкосновения пальца.
            // Измеряется в абсолютных единицах.
            // Расстояние от места прикосновения до начала координат активной картинки.
            DeltaX = cursorX - ivSender.GetX();
            DeltaY = cursorY - ivSender.GetY();
```

```
        }


        // Перемещение активной картинки в поисках места для создания
узора.
        if (motionEvent.Action == MotionEventActions.Move)
        {
            // Из координат курсора, во время перемещения, вычитаем
расстояние от места прикосновения
            // до начала координат выбранной картинки.
            // Благодаря этому картинка относительно пальца будет
неподвижна,
            // и будет двигаться точно под пальцем.
            ivSender.SetX(cursorX - DeltaX);
            ivSender.SetY(cursorY - DeltaY);


            // Отлавливаем место возможного отпускания картинки.
            for (int i = 0; i < RectPositionImages.Length; i++)
            {
                if (ivSender.GetX() < (RectPositionImages[i].Left + 30) &&
                    ivSender.GetX() > (RectPositionImages[i].Left - 30) &&
                    ivSender.GetY() < (RectPositionImages[i].Top + 30) &&
                    ivSender.GetY() > (RectPositionImages[i].Top - 30))
                {
                    // Если место найдено просигналим изменением цвета
```

```
            // передвигаемой картинки.
            ivSender.SetColorFilter(Color.DarkViolet);

            // Устанавливаем флаг место найдено.
            posSender.isFoundPos = true;


            // Запоминаем новую позицию для активной картинки.
            posSender.newPos = RectPositionImages[i];

            break;
        }

        // Если отдалились от места возможного приземления
        // снимаем цветовую сигнализацию перемещаемой картикни.
        ivSender.ClearColorFilter();

        // Снимаем флаг обнаружения места приземления.
        posSender.isFoundPos = false;
    }
}


// Поднимаем палец.
if (motionEvent.Action == MotionEventActions.Up)
```

```
        {

            // Возвращаем картинке нормальный масштаб.
            ivSender.ScaleX = 1;
            ivSender.ScaleY = 1;

            // Восстанавливаем непрозрачность.
            ivSender.Alpha = 1.0f;

            // Отпуская курсор принимаем новые координаты для данной картинки.
            // А картинка которая была уже на этой позиции отправляется на прежнее
            // место активной картинки.
            if (posSender.isFoundPos == true)
            {

                // Извлекаем информацию о нижележащей картинке под перемещаемой.
                for (int img = 0; img < layoutMain.ChildCount; img++)
                {
                    ImageView imageView = (ImageView)layoutMain.GetChildAt(img);

                    if (imageView.GetX() == posSender.newPos.Left && imageView.GetY() == posSender.newPos.Top)
                    {
```

```
                    // --- Если картинку нашли ---

                    // Поднимаем её над всеми картинками.
                    imageView.BringToFront();

                    // Перемещаем с анимацией картинку на старое место передвигаемой картинки.
                    imageView.Animate().TranslationX(posSender.Position.Left);
                    imageView.Animate().TranslationY(posSender.Position.Top);

                    break;
                }
            }

            // Активная картинка устанавливается на новое место.
            ivSender.Animate().TranslationX(posSender.newPos.Left);
            ivSender.Animate().TranslationY(posSender.newPos.Top);

            // Сбрасываем флаг найденного места.
            posSender.isFoundPos = false;

            // Восстанавливаем настоящий цвет.
            ivSender.ClearColorFilter();
        }
        else
        {
```

```
                // Если нижележащая картинка не найдена,
                // возвращаем активную картинку на прежнее место.
                ivSender.Animate().TranslationX(posSender.Position.Left);
                ivSender.Animate().TranslationY(posSender.Position.Top);

                // Восстанавливаем настоящий цвет.
                ivSender.ClearColorFilter();
            }
        }

    }

    public override void OnRequestPermissionsResult(int requestCode, string[]
permissions, [GeneratedEnum] Android.Content.PM.Permission[] grantResults)
    {
        Xamarin.Essentials.Platform.OnRequestPermissionsResult(requestCode,
permissions, grantResults);

        base.OnRequestPermissionsResult(requestCode, permissions,
grantResults);
    }
```

```
}

// Собственный класс обязательно наследуем от Java.Lang.Object
// иначе свойству элементов Tag не сможем
// присвоить объект класса.
class Positions : Java.Lang.Object
{
    // Текущая позиция
    public RectF Position;

    // Новая позиция
    public RectF newPos;

    // Флаг обнаружения места приземления.
    public bool isFoundPos;
}
```