Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Центр післядипломної освіти
(повна назва)

Кафедра Програмної інженерії
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти – другий (магістерський)

Дослідження методів розпізнавання жестів
щодо реалізації ігрової механіки
(тема)

Виконав: студент 2 курсу, групи ІПЗмзд-18-1
Коверя Д.М.
(прізвище, ініціали)

спеціальності 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

освітньо-наукової програми
(тип програми)
Інженерія програмного забезпечення
(повна назва освітньої програми)

Керівник доцент Назаров О.С.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри, проф. _____ З.В.Дудар

2020 р.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Центр післядипломної освіти

Кафедра Програмної інженерії

Рівень вищої освіти – другий (магістерський)

Спеціальність 121 – Інженерія програмного забезпечення

Тип програми освітньо-наукова програма

Освітня програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студенту Ковері Дмитру Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів розпізнавання жестів щодо реалізації ігрової механіки

затверджена наказом університету від "___" _____20 ___ р. № _____

2. Термін подання студентом роботи до екзаменаційної комісії
22 травня 2020 р.

3. Вихідні дані до роботи *структура та організація ігрового програмного забезпечення, класичні ігрові механіки, методи розпізнавання жестів, навчальне навантаження кафедр, ігровий рушій Unity, бібліотеки з розпізнання жестів. Прикладне ПО для ОС Android та iOS, MVC методологія розробки*

4. Перелік питань, що потрібно опрацювати в роботі *мета роботи, аналіз проблемної галузі і постановка задачі, формалізація предметної галузі, огляд класифікації жестів та існуючих підходів до їх розпізнавання, огляд мабільного ринку ігрового програмного забезпечення, розробка ігрової програмної системи що комбінує розпізнання жестів на динамічну ігрову механіку, виявити особливості реалізації MVC архітектурі у ігровому середовищі*

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спецчастина	доцент Назаров О.С.		

5 Консультанти розділів роботи

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1.	Аналіз предметної галузі	13 березня 2020 р.	виконано
2.	Огляд існуючих методів	02 квітня 2020 р.	виконано
3.	Дослідження методів розпізнавання жестів щодо реалізації ігрової механік	14 квітня 2020 р.	виконано
4.	Підготовка пояснювальної записки	02 травня 2020 р.	виконано
5.	Спецчастина	09 травня 2020 р.	виконано
6.	Нормоконтроль, рецензування	16 травня 2020 р.	виконано
7.	Підготовка презентації та доповіді	16 травня 2020 р.	виконано
8.	Попередній захист	20 травня 2020 р.	виконано
9.	Занесення диплома в електронний архів	22 травня 2020 р.	виконано
10.	Допуск до захисту у зав. кафедри	22 травня 2020 р.	виконано

Дата видачі завдання _____ _____ 2020 р.

Студент _____
(підпис)

Керівник роботи _____ доцент Назаров О.С.

(підпис)

РЕФЕРАТ

Звіт з науково-дослідної практики: 34 с., 10 рис., 7 табл., 3 додатки, 17 джерел.

СТРОКОВІ ЖЕСТИ, РОЗПІЗНУВАННЯ ХМАРЗ ТОЧОК, МОБІЛЬНІ ІГРИ, ІГРОВІ МЕХАНІКИ, ЮНІТІ, ГЕЙМПЛЕЙ, РОЗПІЗНАННЯ ЖЕСТІВ, AGATe, GREAT, GECKo, МАШИННЕ НАВЧАННЯ, НАВЧАНЯ НА ПРИКЛАДАХ.

Метою роботи є методи вводу і розпізнання жестів, алгоритми аналізу, проектування та розробка програмної системи, що імплементує розпізнання жестів у ігрову механіку.

Об'єкт дослідження – процес розпізнавання жестів в ігрових системах.

Предмет дослідження: методи розробки, які базуються на таких інструментах аналізу 2D stroke-gesture як: $P Point-Cloud Recognizer, Gesture RElative Accuracy Toolkit (GREAT) та AGreement Analysis Toolkit (AGATe), розробки ігрового мобільного додатку за допомогою ігрового рушія Unity, що приймає, оброблює та аналізує введені жести.

В результаті роботи розглянуто види жестів, методи розпізнання жестів, алгоритми аналізу, розроблено програмну вводу бази жестів на рівні розробника, реалізовано на базі мобільної платформи ввід та розпізнавання жестів користувача, створено мобільний ігровий додаток, що інтегрує попередні розробки у єдину систему геймплею.

ABSTRACT

Master’s thesis: 34 pages, 10 figures, 7 tables, 3 appendices, 17 sources.

STROKE-GESTURE, POINT-CLOUD RECOGNIZER, MOBILE GAMES, GAME’S MECHANICS, UNITY, GAMEPLAY, GESTURE RECOGNITION, AGATe, GREAT, GECKo, MACHINELEPLESING, LEARNING FROM EXAMPLES.

The aim of the work is methods of input and recognition of gestures, algorithms for analysis, design and development of a software system that implements gesture recognition in game mechanics.

Development methods are based on such 2D stroke-gesture analysis tools as: $ P Point-Cloud Recognizer, Gesture RElative Accuracy Toolkit (GREAT) and AGreement Analysis Toolkit (AGATe), development of a mobile game application using the Unity game engine that receives, processes and analyzes the entered gestures.

As a result, the types of gestures, gesture recognition methods, analysis algorithms are considered, software input of gesture database at the developer level is developed, input and recognition of user gestures are implemented on the basis of mobile platform, mobile game application is created, integrating previous developments into a single gameplay system.

ЗМІСТ

ВСТУП

Сучасний рівень розвитку інформаційних технологій, алгоритмів і методів сприяють появі нових інтерфейсів взаємодії користувачів і пристроїв. Появі, вдосконаленню і розповсюдженню нових інтерфейсів взаємодії сприяє ріст об рахункових можливостей сучасних мобільних пристроїв, ріст рівня якості їх сенсорів.

Ринок прикладного програмного забезпечення останні роки перестав показувати зростання. З іншого боку Google Play Store нараховує близько 350 тис. активних ігрових аплікацій і ця кількість зростає на 4 тис. щомісяця [1]. Щодо Apple App Store відповідні цифри 903 тис. та 7,5 тис. відповідно [2]. Такі цифри свідчать про надвисоку і все загострюючуся конкуренцію за світовий ринок розподілу доходів мобільної ігрової індустрії. А боротися є за що – бо на 2019 рік кількість завантажень додатків мобільних додатків склала 194 млрд. а кількість витрачених коштів гравців сягнула 101 млрд. $ [3].

Все це актуалізує пошук нових ігрових механік, імплементацію у ігрове середовище нових технологій, комбінації існуючих підходів на новий ряд. Яскравим прикладом втілення ідеї синергії апаратних, програмних та гейм дизайнерських рішень стала гра Pokémon Go що наразі нараховує 486 млн. активних гравців [4] та згенерувала 3,87 мрлд $ [5].

Зі зростанням обрахункових можливостей, величини оперативної і постійної пам’яті, якості сенсорів набувають розвитку різноманітні програмно-апаратні та алгоритмічні методи і підходи щодо розпізнавання жестів, покращуються їх здатність до розпізнання, полегшується імплементація у ігрове середовище. Наразі з’явилася можливість не тільки виділити обчислювані ресурси на розпізнання та класифікацію жестів, але й на відображення естетично привабливою графіки, так і обробку управлінської логіки.

Тому перспективною технологією, що може бути імплементована у ігровий процес, ми вважаємо розпізнання жестів. Різноманітність видів жестів, способів їх

введення, чисельні варіанти технологій обробки та інтегрування у ігровий процес робить актуальною проблему дослідження методів розпізнання жестів що реалізації ігрової механіки.

Реалізацію конкретної програмної технології розпізнавання жестів буде зроблено на язику програмування C# , саму презентацію гри буде розроблено з використанням ігрового рушія Unity, що є одним з найпопулярніших ігрових рушіїв у світі та дозволяє розроблювати як ігрові мобільні аплікації так і десктопні програмні засоби [6].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Історія розвитку мобільної ігрової індустрії

Початок індустрія мобільних ігор бере у 1997 році, коли компанія Nokia випустила Nokia 6110 – перший телефон з вбудованою грою Snake. Гра запускалася з навігаційного меню. В ній потрібно було поглинати точки та нарощувати довжину змії. Гра набула безліч фанатів а сама компанія впроваджувала її варіації у безліч своїх телефонів.

2001 року Siemens випускає свою 45-ту серію сотових телефонів с принципово новими іграми, такими як Stack Attack, Baloon Shooter, Battle Male, головоломка Super Mind, Moove the Box, Race Ace.

2007 року компанія Nokia випускає N-GAGE 2.0 – доопрацьовану платформу для розробки та розповсюдження ігрових додатків. N-GAGE підтримували всі телефони, що виходили після 2007 року. Розповсюдження додатків – тільки онлайн способом. Усі ігри добавлялися у N-GAGE GAMES – програму-клієнт, що дозволяв скачувати нові ігри і спілкуватися з друзями з меню смартфона. Проте сервіс обладав значними недоліками – слабким залізом, відсутністю апаратної підтримки OpenGL 2.0 (це призводило до значних витрат на портування додатків), Nokia ввела значну плату за користування не надав жодної підтримки розробникам. Тож 31 жовтня 2009 року платформу біло ліквідовано.

Та незважаючи на це, ринок ще не було сформовано – користувачі не могли вільно скачувати ігри, розробники не могли їх продавати. Стан речей змінився з виходом Java2ME – компактної версії платформи Java, що представляє собою середу для розробки і виконання мобільних додатків [7]. У користувачів появилась можливість скачувати та встановлювати ігри, а такі студії як EA, Ubisoft, Konami, Disney почали випускати мобільні версії своїх 9а стосу.

Телефони продовжували еволюціонувати, становилися потужнішими та виконували все більше функцій. На ринку з’являється новий клас пристроїв – смартфони.

29 червня 2007 року на ринок США поступив iPhone 2G сенсорний старт фон першого покоління iPhone. 11 липня 2008 року був представлений смартфон другого покоління iPhone 3G. 19 липня було відкрито App Store – магазин додатків, частина онлайн магазину iTunes Store. Він надавав технології розробки та поширення мобільних додатків, дозволяв залучитися до розробки ігрових додатків тисячі розробників з усього світу. Він пропонував прості і чіткі правила розробки мобільних додатків, а також величезну аудиторію, що жадала нових вражень. Для користувачів же смартфонів App Store став революційною технологією – вперше можливо було зручно шукати, завантажувати та грати у безліч ігор, як безоплатних так і оплачуваних.

22 жовтня 2008 року компанією Google був презентований он-лайн магазин для продажу мобільних додатків, що працюють на операційній системі Android — Android Market [8]. Користувачі отримали можливість легко завантажувати додатки, виставляти оцінки, залишати коментарі. З 27 жовтня 2008 розробники мобільних додатків, що конвертуються у нестандартний байт-код для віртуальної машини Dalvik, отримали можливість завантажувати їх безкоштовні аплікації до магазину Android Market. З початку 2009 року – і платні додатки також [9].

На момент виходу Android Market в ньому було доступно лише 50 аплікацій, всі вони були безкоштовними. Проте прості і прозорі правила для розробників, дешевизна ліцензії (одноразовий внесок у 25$) та зростання аудиторії користувачів смартфонами Android призвело до вибухового росту як кількості розробників, так і переліку ігор, що є доступними для завантаження [9].

1.2 Сучасний стан ринку мобільних ігрових додатків

Останні роки індустрія комп'ютерних ігор перебуває на піку своєї активності та прибутковості. Стрімкий розвиток смартфонів, найпотужніші з котрих вже наздоганяють ноутбуки, призвело до зростання кількості, асортименту та якості ігрових програмних продуктів що можуть ними виконуватися. Згідно з [10] у 2019 році світові ігрові магазини згенерували понад 152 млрд. $, показавши річний приріст у 9.6% (див. Рис 1.1).

На мобільні платформу було портовано такі хіти декстопних ігор як Half-Life, Half-Life 2, Counter-Strike 1.6, Deus Ex, GTA SA, Bully, XCOM та багато інших. Гру Pokémon GO встановило на свої телефони понад 800 млн. користувачів. Ігри посідають перше місце як зі кількістю скачувань, так і за різноманіттям, також по часу проведення та обсягу генерованого доходу.

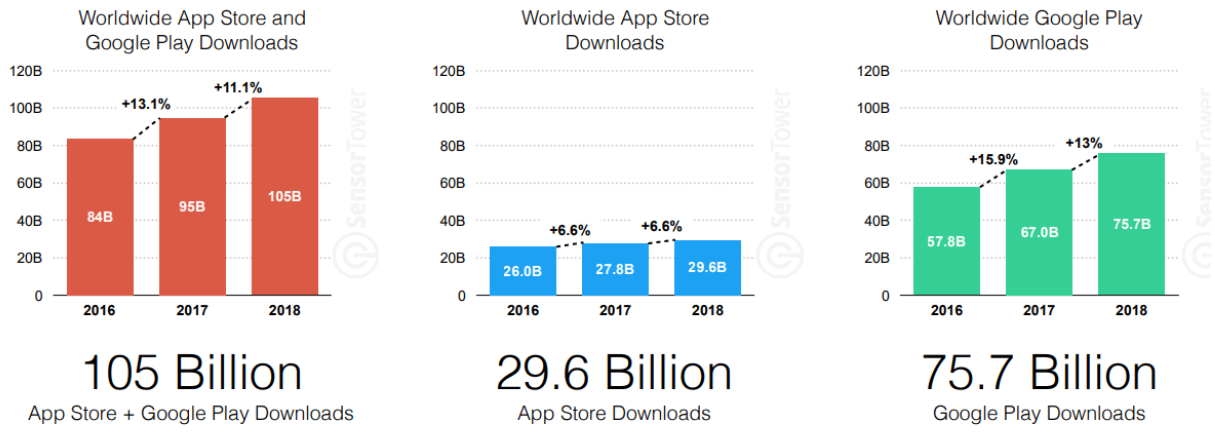

Рисунок 1.1 – Зростання кількості завантажень мобільних додатків

Глобальна аудиторія мобільних ігрових додатків оцінюється у понад 2.2 млрд. [12]. Сумарна кількість ігрового часу, що була витрачена між 2016 та 2018 роками зросла на 50% та сягнула 1350 млрд. годин. Гравці проводять в середньому на рік понад 120 млрд. годин у ігрових додатках. Користувачі в середньому проводить 3 години в мобільному пристрої. Зростання с 2016 — 50%.

Ринкова вартість компаній, націлених на мобільний ринок в 2018 році, в середньому зросла на 360%.

Інтенсивно зростає кількість інсталяцій, особливо у Китаї (див. Рис 1.2).

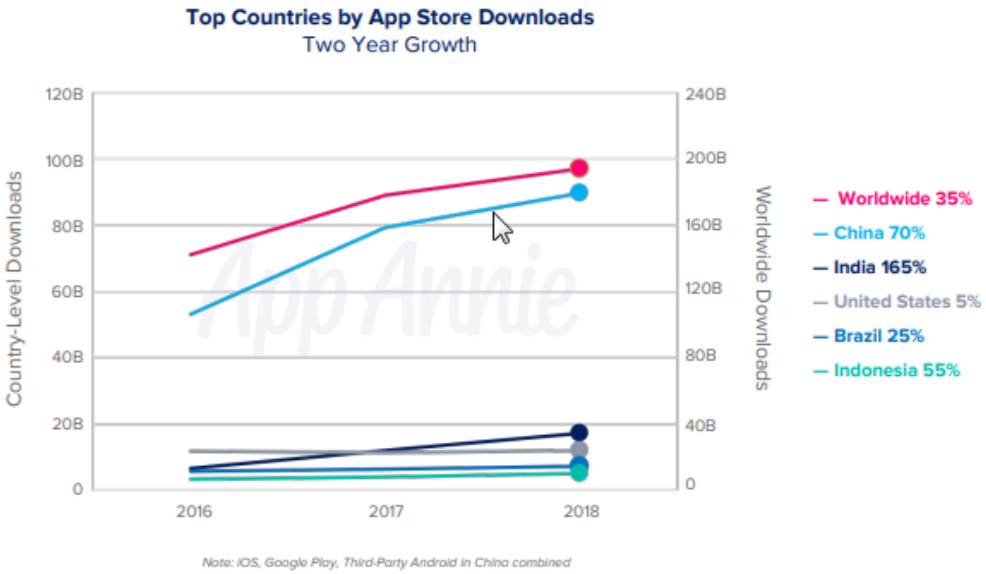

Рисунок 1.2 – Зростання кількості мобільних завантажень

Не менш шокуючими темпами зростають і витрати, що їх роблять користувачі мобільних додатків (див. рис 1.3).

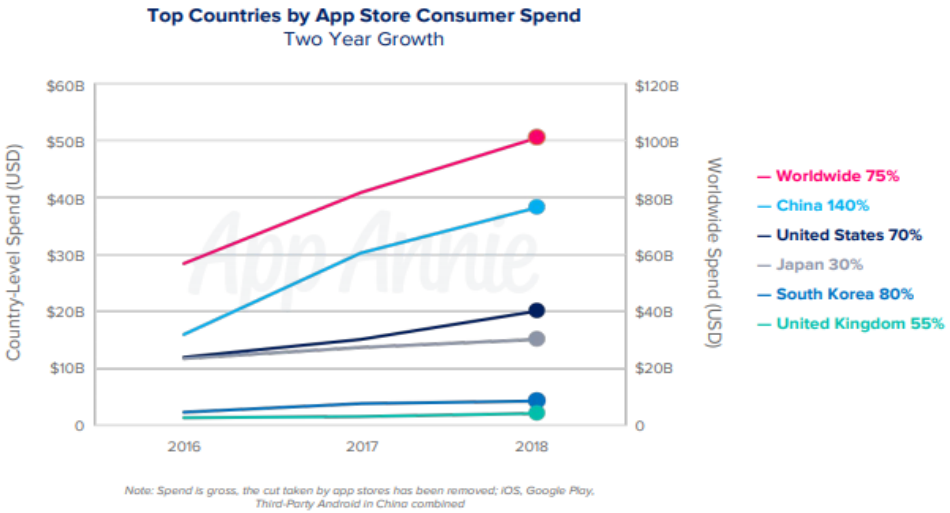

Рисунок 1.3 – Зростання витрат у мобільних додатках

У 2018 році Тім Кук(Timothy Donald Cook, генеральний директор компанії Apple) оголосив, що у AppStore зареєстровано понад 20 млн. розробників, що за понад 500 млн. що недільних відвідувачів платформи [13] Станом на травень 2019 понад 2,2 млн. аплікацій наразі доступно у магазині AppStore. Їх загрузили та встановили понад 8,9 млрд. разів. Близько 40 тисяч аплікацій добавляється щомісяця. 90% відсотків ігрових аплікацій є безкоштовними.

Google Play (до 2012 року Android Market) магазин від Google, що дозволяє власникам пристроїв з мобільною операційною системою Android та іншими завантажувати і купувати різні 13а стосунки, книги, фільми і музику. Загальна кількість скачувань перевищила 330 млрд. (січень 2012-серпень 2018), користувачі витратили понад 85 млрд. доларів (січень 2012-серпень 2018), станом на травень 2019 у магазині для Android понад 2.6млн. аплікацій [2]. Близько 136 тисяч аплікацій добавляється щомісяця. Понад 95% відсотків ігрових аплікацій є безкоштовними. За даними App Annie понад 5000 мобільних додатків заробило понад 1 млн. доларів. Задля більшої ефективності розповсюдження контенту Google Play виокремлює «топ» додатки. Вони розподіляються на:

- топ безкоштовних;
- топ оплачуваних;
- бестселери;
- найбільш швидко зростаючі;
- топ категорії/під категорії.

Така система реклами та розповсюдження додатків зарекомендувала себе з найкращої сторони. Рейтингом обирається найякісніший контент, що максимально задовольняє потреби користувачів. Це являється однією з причин того, що платформа Android і магазин мобільних додатків Google Play користуються популярністю у понад 1 млрд. користувачів.

1.3 Характеристики жестів як об’єкта розпізнання

Жест (від лат. Gestus – рух тіла) – деяка дія або рух людського тіла або його частин, що має визначене значення або зміст, то є представлений знаком або символом.

Широко використовує жести, що включають у себе такі дію, як вказують на що-небудь або кого-небудь (це одне з нечисленних жестів, чий зміст мало різниться в різних країнах), а також використовуються руки і тіла синхронно з ритмами речі, що потрібно вказати деякі слова або коротко.

Розпізнавання жестів рукою стає складним завданням у сфері комп’ютерного зору, особливо після появи мобільних пристроїв. Хоча алгоритми розпізнавання жестів широко застосовуються в системах взаємодії людина-мобільний, важко задовольнити вимоги реального часу та надійність в різних освітлювальних умовах та задньому фоні.

У комп’ютерних інтерфейсах розрізняють два типи жестів: онлайн та офлайн [29]. Ми розглядаємо онлайн-жести, які також можна розглядати як прямі маніпуляції, такі як масштабування та обертання. На відміну від них, жести в режимі офлайн зазвичай обробляються після закінчення взаємодії; е. г. для активації контекстного меню намальовано коло.

Людські руки і тіло мають унікальні візуальними особливостями. У розпізнаванні жестів на основі зображень жести складаються з фрагментів людських рук і / або тіла. Тому використання таких візуальних ознак в ідентифікації жестів цілком обґрунтовано.

Колір – це проста візуальна функція для ідентифікації жестів з фонової інформації. Однак на системи розпізнавання жестів на основі кольорів сильно впливають освітлення і тіні [30]. Ще одна поширена проблема у виявленні кольору шкіри полягає в тому, що колір шкіри людини сильно різниться серед людських рас. Через перерахованих вище проблем, в сучасних підходах, колір

шкіри розглядається тільки як один з багатьох параметрів при ідентифікації жестів.

У розпізнаванні жестів на основі зображення умови освітлення сильно впливають на якість ідентифікації жестів. Тому багато дослідників використовують метод локальних ознак, який не чутливий до умов освітлення. Локальний підхід до об'єктів – це деталізований підхід на основі текстур. Він розкладає зображення на більш дрібні області, які не відповідають частинам тіла [31]. Як показано на Рис. 4, однією з найбільш важливих локальних функцій є перетворення ознак інваріантних об'єктів (SIFT) [32]. Метод SIFT є обертальним, трансляційним, масштабується і частково освітляючи інваріантом. Існує кілька подібних методів локальних ознак, наприклад, SURF і ORBare, запропоновані в більш пізні роки [33, 34]. Як правило, підходи до локальних особливостей також розглядаються тільки як один з безлічі параметрів при ідентифікації жестів. Кілька методів ідентифікації, таких як методи форми та контуру, методи руху і методи навчання, засновані на локальних ознаках.

1.4 Технологія розпізнання жестів Hand Tracking

За останні десятиліття дослідження в області машинного научання достигли величезного прогресу в області комп'ютерного зору і їх алгоритми стали все частіше використовуватися на практиці. Однак до сих пір розповсюдження алгоритмів машинного научання багато в чому стримувалися технічними проблемами, зв'язаними з їх застосуванням. Одна з ключових проблем полягає в тому, що багато алгоритмів, в том числі алгоритми комп'ютерного зору, потребують занадто великих ресурсів для опрацювання в реальному режимі часу на смартфонах и звичайних комп'ютерах. Окрім того, як і в інших областях програмування, перед розробниками повстає питання перенесення їх програм, повторного використання відкритого коду і швидкого розгортання (див. Рис 1.4).

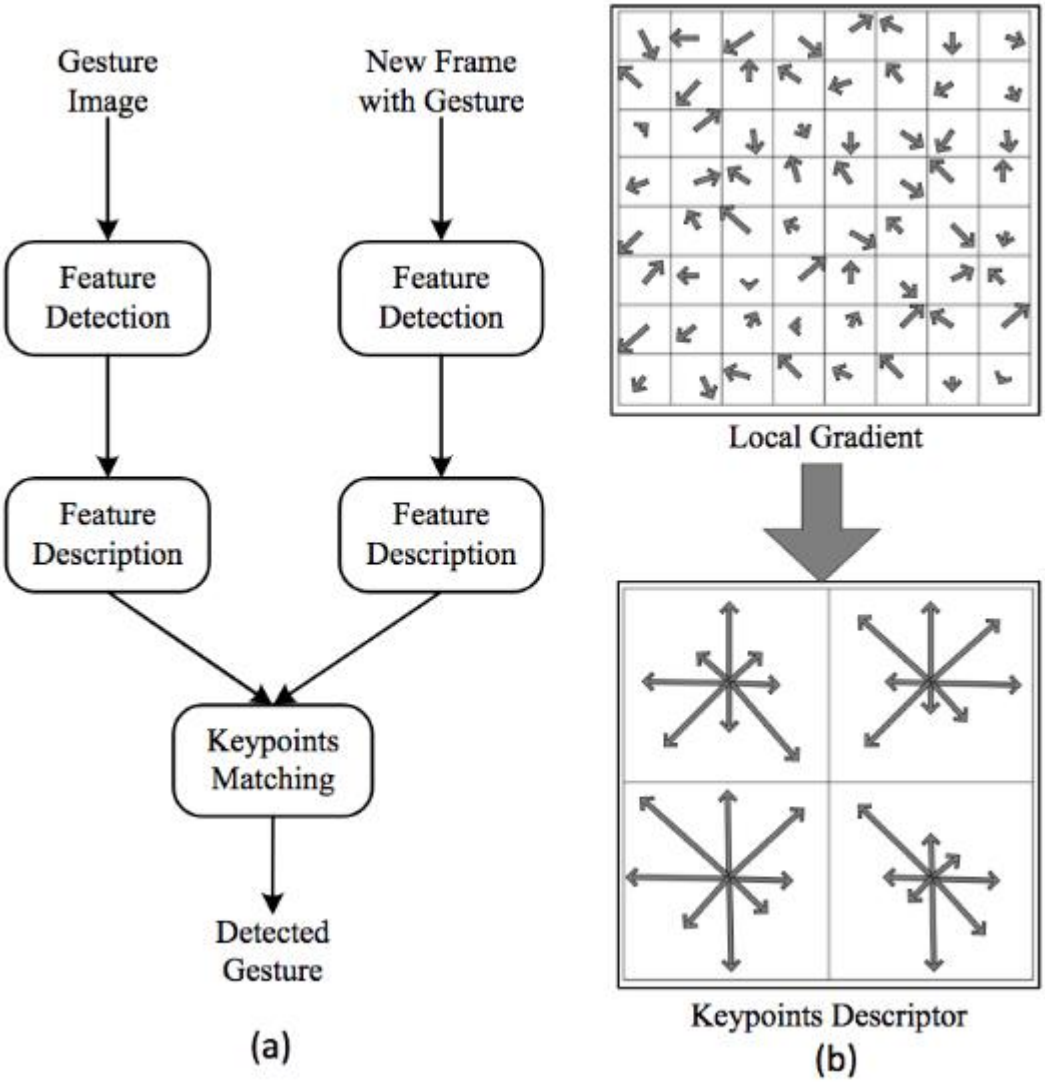

Рисунок 1.4 – SIFT: (a) алгоритм SIFT для ідентифікації жестів; (b) Приклад дескриптора функції SIFT.

Здатність сприймати форму і рух рук може бути важливою складовою в покращенні досвіду роботи користувачів у різних технологічних областях та платформах. Наприклад, він може стати основою для розуміння мови жестів та контролю жестів рукою, а також може забезпечити накладання цифрового вмісту та інформації поверх фізичного світу в доповненій реальності. Приблизно до людей, стійке сприйняття рук у режимі реального часу є вирішальним складним завданням комп’ютерного зору, оскільки руки часто закупорюють себе чи один

одного (наприклад, оклюзії пальця / долоні та тремтіння руки) та не мають високої контрастності.

У 2019 році корпорація Google запропонувала розробникам нове програмне забезпечення на основі машинного навчання для точного відстежуваня жестів смартфонами. Вихідний код та наскрізний сценарії викладені на порталу GitHub. Google представила технологію Hand Tracking на конференції по комп'ютерному зору і розпізнаванню образів в червні 2019 року і невдовзі впровадила її у кросплатформенний фреймворк MediaPipe для прикладного машинного навчання, працюючого з такими задачами, як розпізнавання облич і виявлення об'єктів. MediaPipe – відкрита крос-платформа для побудови трубопроводів для обробки перцептивних даних різних модальностей, таких як відео та аудіо. Такий підхід забезпечує високу точність відстеження рук і пальців, використовуючи машинне навчання (ML), щоб зробити висновок про 21 3D-ключову точку руки лише з одного кадру. Технологія на основі трьох моделей ШІ, працюючих в тандемі. Сітка фокусується на координатах 21 точки ладоні, для чого детектор ладоні – BlazePalm – аналізує кадр і отримає обмежувальний прямокутник, модель орієнтира руки оцінює окреслену область і визначає трьохмірні точки руки, а розпізнавач жестів шукає попередньо обраховану конфігурацію точок в наборі жестів [24] (див. рис 1.5).

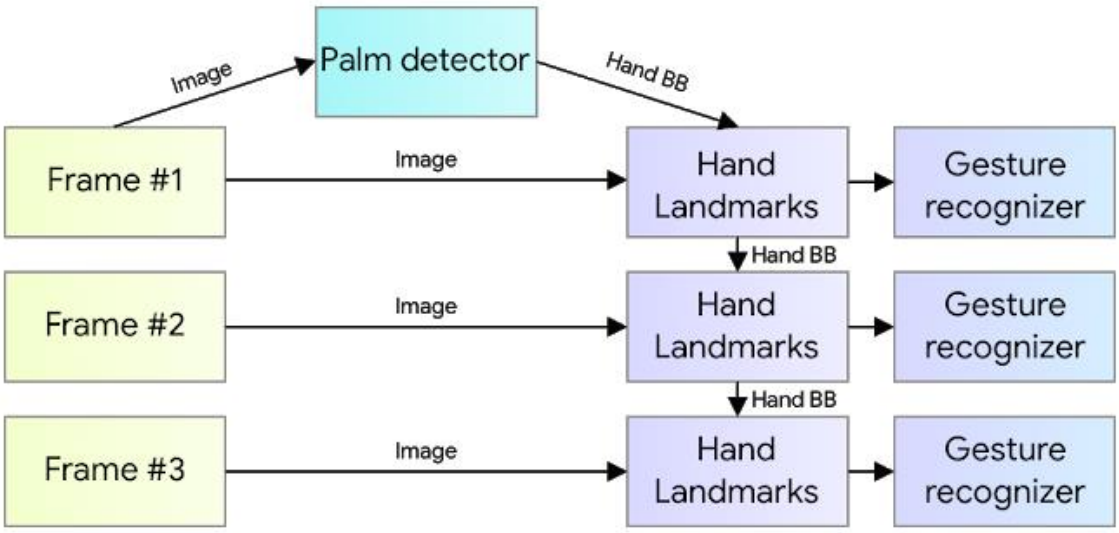

Рисунок 1.5 – Конвеєр розпізнання жестів

Оскільки сучасні підходи покладаються в основному на потужні робочі середовища для висновку, метод досягає в режимі реального часу продуктивності на мобільному телефоні і навіть масштабує для декількох рук. Автори намагались оптимізувати код для роботи на звичайних девайсах користувачів. В якості приклада вони показали демо-версію додатка до смартфонів, котре в реальному часі створює модель руки, що може знагодитися в якості розпізнавання жестів пальцями за допомогою камери [25].

Розпізнання жестів базується на тому, що зверху на передбачуваний скелет руки розробники застосовують простий алгоритм для отримання жестів. По-перше, стан кожного пальця, 18апр.. зігнуті або прямі, визначається накопиченими кутами стиків. Потім програма відображає набір стану пальців на набір заздалегідь визначених жестів. Цей простий, але ефективний метод дозволяє модулю оцінити основні статичні жести з розумною якістю. Існуючий трубопровід підтримує підрахунок жестів з різних культур, наприклад Американські, європейські та китайські, а також різні ручні знаки, включаючи «великий палець вгору», кулак, «добре», «рок» та «павук».

1.5 Розпізнання жестів як хмар з точок

В даний час зростає використання та розповсюдження сенсорних пристроїв введення, таких як iPad, iPhone та Android.

Також зростає кількість розробок додатків на основі сенсорного вводу, сприяє зростаючій потребі в інструментах для підтримки розвитку для таких платформ. Нові програми можуть вимагати жесту розпізнавання з урахуванням нових жестів, які просто не побудовані у існуюче програмне забезпечення.

Сучасні методи розпізнавання жестів, такі як прихованих марківські моделі [26], засновані на характеристиках статистичних класифікаторів

[27] або суміш класифікаторів [28] як правило, потребують значних технічних знань для їх розуміння та розвитку для нових платформ або знань з інших галузей, таких як теорія графів [29]. На противагу цьому підходи до хмарного розпізнання точок вирішує цю проблему, пропонуючи дешеві, просто щоб зрозуміти та втілити ще, але високоефективні підходи розпізнавання жестів [2, 3, 23]. Ці підходи, вчені Radu-Daniel Vatavu, Lisa Anthony та Jacob O. Wobbrock називають $ -сімейство методів, включаючи лише прості геометричні обчислення та прямі внутрішні уявлення. Крім того, алгоритми дуже доступні через публікацію псевдокоду, який розробники може використовувати для власних платформ.

Попри те $ - сімейство аналізаторів мають обмеження. Наприклад, $ 1 і Protractor обробляють лише жести без скасування. $ N і $ N- Protractor зосередилися на виправленні цього обмеження, додаючи підтримку мультиліній. Окрім того, суттєвою проблемою є порядок та напрямок обведеня можуть суттєво відрізнятися між користувачами, що мають на увазі один і той же символ. Зі зростанням складності символу, кількості штрихів, що його утворює, у геометричній прогресії зростає об’єм перестановок, що їх має запам’ятовувати обчислювач, і це стає серйозною проблемою навіть для персональних комп’ютерів не кажучи вже про мобільні пристрої. З цієї позиції алгоритм $ P дає високу точність, низьку складність і низьку перешкоди для прийняття. Експерименти показали середню точність 98%

для $ P, що перевищує $ N як залежно від користувача, так і тестування незалежне від користувача. Саме з цієї причини реалізацію алгоритму хмарних точкових розпізнавань жестів $ P було обрано для імплементації розпізнавання жестів щодо реалізації ігрової механіки.

1.6 Висновки по розділу та постановка задачі дослідження

Мобільні ігрові системи – величезний клас додатків, надзвичайно велика сфера комп'ютерної інженерії за такими показниками як: кількість задіяного персоналу, обсягом генерує мого доходу, різноманітністю жанрів, видів, обсягом скачувань, часом, що його проводять у мобільних додатках гравці по всьому світу.

Та найголовніший висновок – ринок мобільних ігор стрімко зростає. За обсягом оборотного капіталу він вже випередив ігри для персональних комп'ютерів а також більшість напрямів системного чи прикладного програмування. У зв'язку з цим істотно посилюється конкуренція як між фірмами та ігровими тайтлами, так і між напрямками та категоріями ігрових продуктів.

Все це ставить перед розробниками необхідність у пошуку нових напрямків розробки ігрового процесу, інтеграції у ігровий всесвіт останніх досягнень у різноманітних галузях комп'ютерної інженерії. Інтеграція та комбінація різноманітних підходів та створення на основі них нових інноваційних видів геймплею є справжнім викликом часу у надскладному конкурентному середовищі, яке собою являє на даному етапі розробка мобільних ігрових додатків.

Однією з перспективних напрямів такої імплементації є впровадження алгоритмів розпізнавання жестів. На даному етапі розвитку індустрії ігор, доволі мало ігрових систем робить процес розпізнання жестів за основу гемплею. В той же час, як нам вдалося дізнатися з аналізу наукових робіт і публікацій, в останні часи появилося доволі багато алгоритмів розпізнавання та класифікації жестів, що мають достатній рівень відсотку впізнання а також помірне використання пам'яті та обчислюваних ресурсів мобільного пристрою, щоб бути впровадженим у ігрові механіку.

Таким чином виникає потреба у створенні нової ігрової програмної системи, що поєднала би у собі досягнення алгоритмів розпізнавання жестів, інтегрувала у

ігрову механіку та забезпечила гнучкий інтерфейс користувача. Даний програмний продукт повинен враховувати обмеженні обчислювальні можливості мобільних телефонів а також забезпечувати швидкий відгук на введені символи. Як бажана опція повинна бути забезпечена графічна привабливість за захоплюючий ігровий процес.

2 ПОБУДОВА МОДЕЛІ ВЗАЄМОДІЇ АНАЗІЛАТОРУ ЖЕСТІВ ТА ІГРОВОЇ МЕХАНІКИ

2.1 Побудова моделі розпізнання жестів

Останнім часом розробники інтелектуальних систем все більше уваги приділяють автоматичному розпізнаванню жестів за допомогою візуальних систем. Такий інтерес викликаний природним характером і зручністю використання інтерфейсу на основі жестів, а також можливістю його застосування в більшості областей людської діяльності. Постановка завдання розпізнавання жестів комплексна і враховує неоднозначну природу статичних і динамічних жестів, проблеми виділення руки на навколишньому фоні, умови освітлення і перешкоди. Рішення завдання в більшості випадків передбачає вибір алгоритмів з використанням комп'ютерних ресурсів.

2.1.1 Постановка проблеми розпізнання жестів

В роботі розглянуто алгоритми аналізи стосовно розробки та впровадження у ігровому процесі. Оскільки управління відбувається в режимі реального часу, то необхідний алгоритм, який не потребує великих обчислювальних витрат. Ранні технології розпізнавання жестів припускали використання маркерів, прикріплених до кінчиків пальців користувача. За допомогою відповідного алгоритму визначалися наявність маркера і його колір, виконувалася ідентифікація пальців, задіяних для формування жесту. Однак використання маркерів накладає деякі обмеження на роботу користувача, і, як наслідок, переваги стали надавати безконтактним технологіям.

Сучасні методи використовують більш прогресивну техніку на основі комп'ютерного зору. Розпізнавання жестів може здійснюватися методом створення простору кривих, суть якого полягає в знаходженні граничних контурів руки. Даний підхід досить надійний і інваріантний до переміщень і обертанням руки, проте вимагає великих обчислювальних витрат. Також було запропоновано алгоритм розпізнавання положення руки за допомогою зображень скелета руки. В даному випадку застосовується багатокамерна система для знаходження центру гравітації руки і найбільш віддалених від нього точок, забезпечуючи, таким чином, інформацію про становище кінчиків пальців, яка використовується для побудови зображень скелета руки і, відповідно, розпізнавання. Решта методів розпізнавання жестів використовують спеціальні методи порівняння, дескриптори Фур'є, нейромережі, гістограми положення, фільтрацію точок. До прикладу можна навести розрахунок інтегрального зображення, його здійснюють за формулою (2.1):

$$I(x, y) = \sum_{i=j=1}^{i \le W,\ j \le H} J(i,\ j), \tag{2.1}$$

де I(x, y) – елемент інтегрального зображення;

W — ширина вихідного зображення;

H — висота вихідного зображення;

J(i, j) — значення пікселя вихідного зображення.

Для більш ефективного розрахунку ознак Хоара використовують рекурентний варіант даної формули (2.2):

$$I(x, y) = i(x, y) + I(x, y-1) + I(x-1, y) - I(x-1, y-1) \tag{2.2}$$

де I(x, y) – елемент інтегрального зображення;

i(x, y) — значення пікселя вихідного зображення.

При розпізнаванні використовується фіксований набір жестів, за допомогою якого задаються певні команди для управління ігровим процесом в режимі реального часу. Тому швидкодію і простота алгоритму мають велике значення.

Такий підхід включає сегментацію зображення руки на основі колірних характеристик шкіри та обмежень розмірності. Обмеження розмірності є необхідною умовою, так як велика розмірність викликає величезні обчислювальні витрати. Всі рухи руки описуються за допомогою базисних векторів. Для зменшення розмірності простору спостережуваних векторів без істотної втрати інформації рекомендується застосовувати аналіз головних компонент (АГК), а для відображення характерних особливостей - аналіз незалежних компонент (АНК) [9].

Для того щоб система відповідала потрібними реакціями на певні сукупності зовнішніх впливів, необхідно підключити процес навчання, який полягає в адаптації системи до конкретних рухів рук користувача (заданому набору жестів). Як об'єкти навчання виступають візуальні зображення рук.
Отже, на основі цих попередніх процедур обробки генерується сигнал, який несе інформацію про жести на зображенні. Далі жест порівнюється з набором жестів з бази даних і, в разі успішної класифікації, йому присвоюється певна команда. На виході системи формується керуючий сигнал, зрадник команду, на основі якої автоматичний пристрій виконує ту чи іншу дію.

2.1.2 Базовий алгоритм розпізнання жестів

Базовий алгоритм розпізнавання жестів на нашу думку повинен складатися з чотирьох етапів:

1) Навчання системи.

Алгоритм повинен дозволяти ігровому середовищу ідентифікувати жест у вхідному зображенні як одну з певних команд. Кожна ідентифікована команда

буде використовуватися для управління ігровим процесом, виконання тих чи інших завдань. Жестам можуть бути надані різні значення в залежності від функцій робота. При завданні команд також можна використовувати інформацію про кількість пальців. Наприклад, один палець може означати - «рух вперед», два - «назад», три - «направо», чотири - «наліво», п'ять - «стоп». Рекомендується використання набору жестів з алфавіту глухонімих, що дозволить зробити гру, можливої для людей, які не мають можливості користуватися традиційними засобами інтерфейсу.

2) Перетворення вхідного зображення.

Для зменшення розмірності простору спостережуваних векторів без істотної втрати інформації застосовується аналіз головних компонент (АГК). Вхідні вектора являють собою відцентрувати і приведені до єдиного масштабу зображення рук. АГК складається в лінійному ортогональному перетворенні вхідного вектора X розмірності N у вихідний вектор Y розмірності M, N <M. При цьому компоненти вектора Yє некорреліруваними, а загальна дисперсія після перетворення залишається незмінною. Матриця X складається з усіх прикладів зображень жестів навчального набору.

Вибір перших M головних компонент розбиває векторний простір на головне (власний) простір $F = \{\Phi_i\}^{M}_{i-1}$, що містить головні компоненти, і його ортогональне доповнення.

Вхідне зображення, за допомогою обчислених раніше матриць, розкладається на набір лінійних коефіцієнтів, званих головними компонентами. сума головних компонент, помножених на відповідні власні вектора, є реконструкцією зображення. Хоча аналіз головних компонент ефективно використовується для скорочення розмірності простору, з його допомогою складно відображати характерні особливості, так як його базисні вектори представляють глобальні характеристики. Для вирішення цієї проблеми використовується метод аналізу незалежних компонент (АНК).

Завданням аналізу незалежних компонент є розкладання спостережуваних випадкових змінних x_j, що описують рух рук, в лінійну комбінацію

незалежних випадкових величин u_k (2.3):

$$x_j = \sum_{j=1}^{N} a_{ij}\, u_j = a_{i1}u_1 + a_{i2}u_2 + \ldots + a_{iN}u_N \qquad (2.3)$$

де x_j – спостережна випадкова змінна;

a – елемент змішаної матриці;

u – випадковий вектор з компонентами $u_1,..,u_N$;

N – розмірність.

Алгоритм обчислення незалежних компонент спирається на центральну граничну теорему, яка стверджує, що за певних умов сума незалежно розподілених випадкових величин прагне до нормального розподілу в міру збільшення кількості доданків. Використовуючи це твердження, пошук незалежних компонентів, як лінійних комбінацій спостережуваних змінних, ведеться таким способом, щоб отримати незалежні випадкові величини, розподіл яких максимально далеко від нормального.

Ми повинні врахувати можливості різної орієнтації жесту, оскільки один і той саме жест може задаватися під різними кутами. Для вирішення цього питання застосовується двовимірне обертання до всіх точок, записаних, пов'язаних з координатами Z і X, таким чином лінія між двома плечима перетворюється на вісь X. Цей кут обчислюється між віссю X і лінією між двома плечима, і виконувана операція обертання вказана нижче (2.4):

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \qquad (2.4)$$

де x`, y` – координати об’єкту у матриці сенсору;

θ – кут операції повороту по віссі x;

x, y – координати об’єкту у реальному середовищі.

3) Сегментація зображення руки на основі ключових характеристик.

У якості ознаки, який використовується для відділення руки від фону на зображенні, можна використовувати колір шкіри. В даному випадку для реалізації сегментації застосовується піксельна модель шкіри. Модель формується виходячи з інформації про кольоровості (тон і насиченість), отримана в результаті попереднього навчання, яке безпосередньо полягає в розміщенні руки користувача в області так званого навчального квадрата. Пікселі, укладені в цю область, використовуються для навчання моделі, після чого виділені пікселі перетворюються з колірного простору RGB в простір HSL, звідки потім виходить інформація про кольоровості. Значення колірного тону H і насиченості S для кожного обраного пікселя утворюють набір x = (x_1,…,x_n), де n - кількість відліків (пікселів), x_i = (H_i,S_i) - значення колірного тону і насиченості i-го пікселя. Для представлення функції щільності ймовірності, яка описує приналежність пікселів до кольору шкіри, обрана Функція Гауса щільності ймовірності (ГФЩЙ). Значення параметрів, що входять в ГФЩЙ (середнє значення x і коваріаційна матриця Σ), обчислюються з набору пікселів з використанням стандартних методів. В результаті ймовірність того, що новий піксель x = (H,S) відповідає за кольором шкіри, може бути обчислена як (2.5):

$$P(\vec{x}) = \frac{1}{\sqrt{(2\pi)^2|\Sigma|}} exp\left(-\frac{1}{2}\ (\vec{x} - \bar{x}) \times \Sigma^{-1} \times (\vec{x} - \bar{x})^T\right) \qquad (2.5)$$

де P ($\vec{x}$) – вірогідність відповідності кольору пікселя;

$\overrightarrow{x}$ – значення кольорового тону і насиченості;

$\bar{x}$ – середнє значення кольору;

Σ – коваріаційна матриця.

В кінцевому підсумку результатом процесу сегментації буде подання руки у вигляді репера області шляхом застосування алгоритму об'єднання пікселів, що задовольняють висловом. Отримані таким чином результати є інваріантними до фону і різних умов освітлення.

4) Механізм класифікації (моделювання).

Для кожного зображення рук обчислюються його головні компоненти, згідно раніше описаним алгоритмом. Зазвичай береться від 5 до 200 головних компонент. Процес розпізнавання полягає в порівнянні головних компонент невідомого зображення з компонентами всіх інших зображень. Для цього зазвичай застосовують будь-яку метрику (найпростіший випадок - Евклідова відстань). При цьому передбачається, що зображення рук згруповані в кластери у власному просторі. з бази даних (або тренувального набору) вибираються зображення-кандидати, які мають найменша відстань від вхідного (невідомого) зображення. Для класифікації можна використовувати метод порівняння еталонів (Template Matching), який полягає у виділенні областей рук на зображенні, і наступному порівнянні цих областей для двох різних зображень. кожна збіглася область збільшує міру схожості зображень.

Однією з важливих технік у цій сфері є мапінг точок. Припустимо, що p^t являє собою точку у вхідному зображенні T^t, тоді p^t може бути представлена як лінійна комбінація усіх точок $\upsilon \frac{t}{i}$ для всіх T^t (2.6):

$$p^t = \sum_{i=0}^{3} \alpha_i v_i^t, \tag{2.6}$$

де p^t – точка у вхідному зображенні T^t;

a_i – коефіцієнт даної проекції;

$\upsilon \frac{t}{i}$ – лінійна комбінація усіх точок.

Для порівняння областей використовуються найпростіші алгоритми, на кшталт попіксельного порівняння. Недолік цього методу полягає в тому, що він вимагає багато ресурсів як для зберігання ділянок, так і для їх порівняння. З причини того, що використовується найпростіший алгоритм порівняння, зображення повинні бути зняті в строго встановлених умовах.

2.2 Розпізнання жестів в контексті ігрового процесу

Розпізнавання жесту відноситься до математичної інтерпретації людських рухів обчислювальним пристроєм. Щоб взаємодіяти з людиною, ігри повинні правильно розуміти людські жести, сенсорний ввід і діяти відповідно до введеної і розпізнаної команди в достатній мірі точності.

Щоб розпізнати жести в контексті ігрового процесу, корисно дослідити загальну і спрощену модель обробки інформації. Sanchez-Nielsen представила [38] узагальнену модель обробки інформації, що складається з чотирьох етапів (див. рис 2.1). Такими етапами є:

— збір даних (данні надходять по двох каналах інформації – аналіз сенсорного вводу гравця на мобільному додатку а також аналіз зображення з камери спостереження);

— аналіз даних (данні сенсорного вводу для розпізнавання 2D строкових жестів; данні з камери спостереження – для 3D жестів та/або міміки гравця);

— прийняття рішення. Зіставляючи отриманні при обробці математичні моделі з тими, що були закладенні на етапі навчання, програма обробки даних повинна відповісти на 2 питання – чи був зроблених будь-який жест з тих, що були закладенні в програму обробки та аналізу жестів, і якщо відповідь та перше питання – так, то який саме з початкову введених жестів було насправді введено в даний момент;

— відклик. Найголовніша з точки зору користувача частина збору інформації та її обробки. Базуючись на результатах 3-го етапу система (в нашому випадку ігровий інтерфейс користувача) потрібна відреагувати передбачуваним і однозначним способом. Якщо жест було введено коректно – дати знати який саме жест був розпізнаний, якщо ж сигнатура введеного жесту не співпала з тими, що були введені в базу аналізатору – повідомити про це користувача.

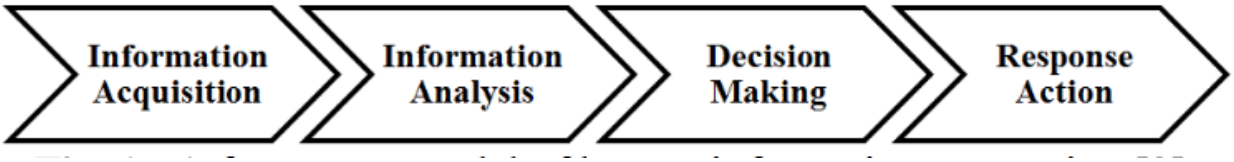

Рисунок 2.1 – Чотирьохступенева модель обробки інформації

Існує п'ять основних частин, пов'язаних з розпізнаванням жестів для HRC: збір даних з датчиків, ідентифікація жестів, відстеження жестів, класифікація жестів і відображення жестів (див. рис 2.2).

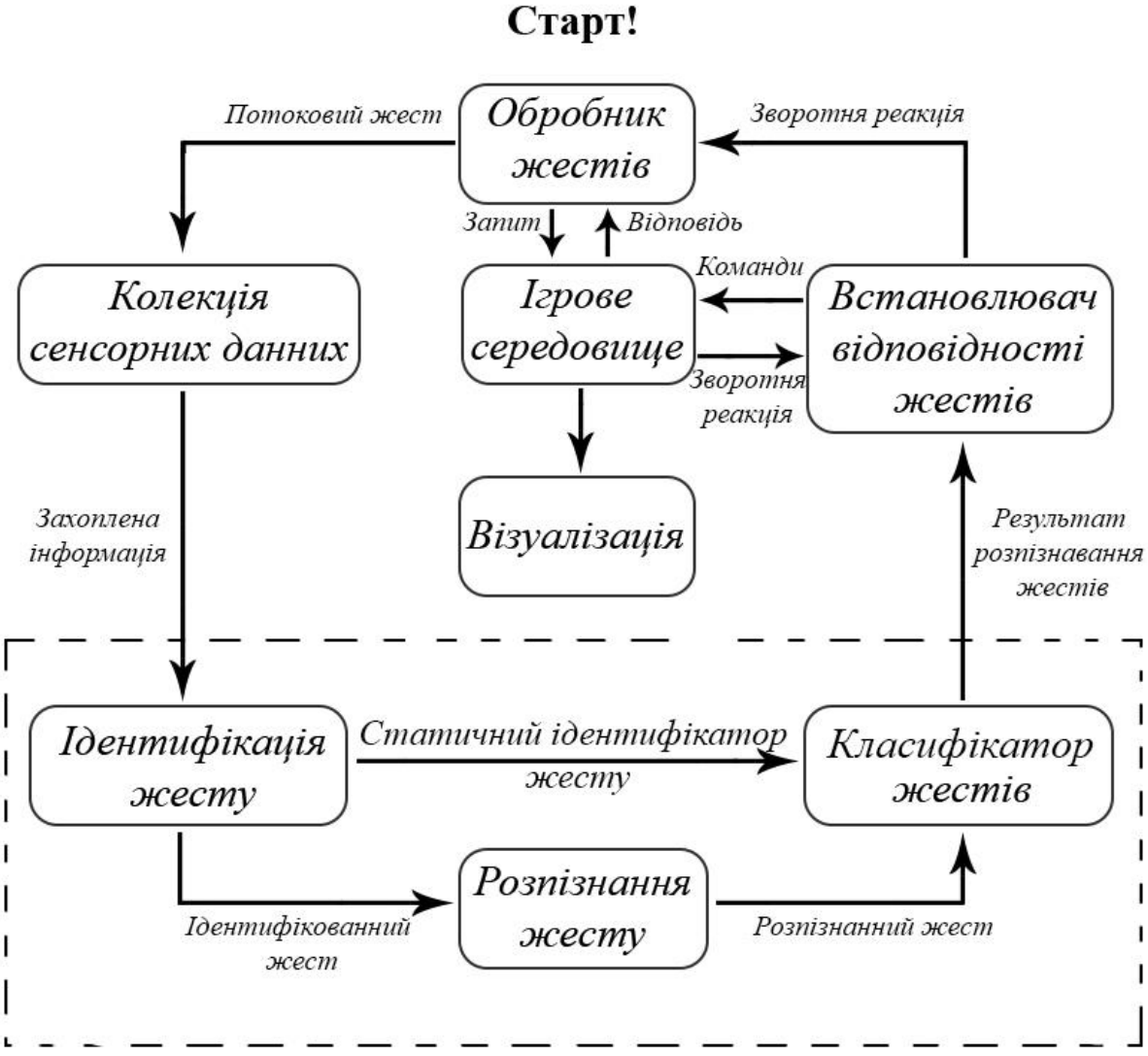

Рисунок 2.2 – Циклічна модель обробки інформації у ігровому середовищі

У порівнянні зі стандартною моделлю, алгоритм роботи аналізатору жестів, що імплементован у інтегроване ігрове оточення зазнав деяких змін і деякого ускладнення.

1. Робота аналізатору є преривною, тобто такою, що активується та дезактивується в залежності від стану ігрового процесу (система може не дозволяти вводити жестові данні на паузі, при зміні ігрових локацій, браку і користувача енергії тощо).

2. Система вводу інформації повинна надавати зворотній зв'язок на кшталт інформація вводиться та розізнається коректно, причому до того як сам аналізатор зможе дати відповідь – чи було введено будь-який жест коректно чи ні.

3. Попередньо повинна бути проведена класифікація жестів: відстеження руху жестів класифікується відповідно до заздалегідь визначеними типами жестів. Грубий класифікатор полегшить роботу більш тонкого та складного алгоритму класифікації жестів.

До прикладу кандидат C порівнюється з кожним темплейтом T_i щоб вирахувати середню відстань між сусідніми точками використовуючи точка-до-точки порівняння (2.7):

$$d_i = \frac{\sum_{k=1}^{N} \sqrt{(C[k]_x - T_i[k]_x)^2 + (C[k]_y - T_i[k]_y)^2}}{N} \quad (2.7)$$

де k – індекс точки для кожного символу;

x, y – координати кожної точки;

T_i – результуючий темплейт з найменшою дистанцією до C.

4. Відображення жесту: результат розпізнавання жестів перетвориться в керуючі команди. На даному етапі необхідно як продемонструвати користувачеві попередньо ідентифікований жест у графічному вигляді, так і виконати ігрові команди, що їх за собою викликає введення жесту за умовами попередньо означених механік та правил ігрового процесу (так у відповідь на введений жест можуть відчинятися чи зачинятися двері, вмикатися чи вимикатися освітлення,

рухатися у певному напрямку заздалегідь означенні предмети, викликати накопичення манни, чи генерування деякої конкретної атакуючої чи оборонної дії). Важливо що на конкретно введених жест система могла детерміновано відповісти певною дією и послідовністю дій.

5. Зважаючи на п.4 важливе значення набуває швидкість реакції відповіді на ввід гравця. Бо ж якщо час зворотного відгуку буде більшим за мінімальний час введення наступного жесту ми ризикуємо опинитися у ситуації, коли користувач на певну дію отримуватиме у відповідь непередбачуваний результат.

6. Центральним елементом стає ігрове середовище (яке бере на себе відповідальність по узгодженню команд, вводу користувача, подіям ігрової механіки, відповідає на запити цих систем, активує чи дезактивує їх) та його візуалізацію (бо ж треба візуалізувати не тільки перипетії ігрового процесу але й введення жестів, етапи їх розпізнавання а також команди, що вони активували).

2.3 Жестовий інтерфейс

Жестова інтерфейс – це підмножина системи введення для графічного призначеного для користувача інтерфейсу для пристроїв, оснащених спеціальними або пристроями введення (відмінними від клавіатури), або сенсорними екранами, і дозволяє емулювати клавіатурні команди (або поєднання клавіш) за допомогою жестів (розчерків, англ. Gesture). Основною мотивацією розробки таких інтерфейсів є поліпшення ергономічності управління, з відмовою від звичного для комп'ютерних програм меню програми.

Подібний інтерфейс може бути реалізований як за допомогою пристроїв координатного введення з можливістю зчитування координати однієї точки дотику (миша або графічний планшет), так і таких, в яких є можливість зчитування координат більш ніж однієї точки (т.з. мультидотик, multitouch) -

сенсорні екрани і панелі. Останні стали широко застосовуватися в інтерфейсах безлічі сучасних смартфонів з сенсорним екраном (напр. IPhone) і ноутбуків (як з тачпадом, так з сенсорним екраном) та інших мобільних пристроїв.

У разі пристроїв з великим розміром екрану - наприклад, планшетних ПК, розчерки-жести є стандартними функціями інтерфейсу управління і пір'яного введення. У разі кишенькових пристроїв (КПК, мобільних телефонів і т.п.), На відміну від класичних графічних інтерфейсів користувача, через малі фізичних розмірів екрану для твору розчерку потрібна менша точність позиціонування, ніж для доступу до традиційних елементів графічного інтерфейсу - натискання «кнопки» або вибору пункту меню.

До переваг можна віднести ергономічність інтерфейсу, особливо для початківців користувачів і легкість у використанні, а значить, використання звичних жестів, інтуїтивно зрозумілих для користувача. Легкість, з якою можна навчити нового користувача. Кастомізацію розпізнавання жестів під специфічні потреби користувача. Більш широкий спектр задіяних у роботі м’язів, що позитивно впливає на роботу опорно-рухового апарату.

До недоліків відносяться висока ймовірність помилок, так як репрезентація жестів у різних людей може відрізнятися і потрібні відповідні апарати та обладнання, такі як камера, сенсор, датчики і так далі. Це приносить до додаткових витрат. Серйозним недоліком можна вважати незручності, що виникають під час користування жестовим інтерфейсом у місцях скупчення людей (адже більшість офісних приміщень у наш час – це відкриті простори (open space).

До різновидів систем жестового інтерфейсу можна віднести TrackIR – першу з систем відстеження рухів голови, які вийшли на масовий ринок. Це пристрій введення, що розробляється компанією NaturalPoint, забезпечує псевдо-віртуальну реальність на персональному комп'ютері. Воно може стежити за рухами голови користувача за координатами X, Y і Z. Отримані дані використовуються в програмах (іграх) для перетворення реальних поворотів голови в віртуальні. Наприклад, в авіасимулятор гравець може оглядати кабіну.

Чутливість налаштовується, щоб запобігти таким повороти, коли користувач не може нормально дивитися на екран.

На персональному комп’ютерові елементом жестового інтерфейсу може слугувати жести мишею. Жести мишею - спосіб управління програмами в комп'ютері за допомогою рухів (жестів) миші, які розізнаються, класифікуються та перетворюються в команди. Ідея методу полягає в заміні навігації по командам меню на введення команд за допомогою знаків, намальованих на площині екрану рухами миші. «Малювати» команди може бути швидше і простіше, ніж шукати потрібний пункт меню або використовувати гарячі клавіші. Крім того, такий спосіб полегшує роботу для тих, кому важко користуватися клавіатурою. Базові жести – на кшталт рух ліворуч для «повернутися на крок назад», рух праворуч для «повернутися на шаг вперед або виконати дії повторно» цілком інтуїтивно зрозумілі та застосовуються вже у деяких програмних продуктах (наприклад браузері Opera). Жест фіксується, якщо натиснути і утримувати старт-кнопку (зазвичай це права кнопка миші) і одночасно «накреслити» мишею потрібну фігуру. Залежно від програми рух може відображатися на екрані у вигляді сліду (StrokeIt, Maxthon) або не з'явитися (яндекс.браузер, Opera).

На мобільних пристроях елементом жестового інтерфейсу слугує технологія мультитач. Мультитач (від англ. Multi-touch - «множинний дотик») - функція сенсорних систем введення (сенсорний екран, сенсорна панель), що здійснює одночасне визначення координат двох і більше точок дотику. Мультитач використовується в жестових інтерфейсів для, наприклад, збільшити або зменшити зображення: при збільшенні відстані між точками дотику відбувається збільшення зображення. Крім того, мультитач-екрани дозволяють працювати з пристроєм одночасно декільком користувачам. Таким прикладом є гра для iPad Cut the Buttons, яка має режим гри для двох гравців, кожен з яких керує віртуальними ножицями двома пальцями. Мультитач дозволяє не тільки визначити взаємне розташування декількох точок дотику в кожен момент часу, але і визначити пару координат для кожної точки дотику, незалежно від їх положення відносно один одного і кордонів сенсорної панелі. Правильне

розпізнавання всіх точок дотику збільшує можливості інтерфейсу сенсорної системи введення. Коло розв'язуваних завдань при використанні функції мультитач залежить від швидкості, ефективності і інтуїтивності її застосування.

2.4 Реалізація ігрового процесу в контексті розпізнання жестів

Ігровий процес, або геймплей (англ. Gameplay), - компонент гри, який відповідає за інтерактивне взаємодія гри і гравця. Геймплей описує, як гравець взаємодіє з ігровим світом, як ігровий світ реагує на дії гравця і як визначається набір дій, який пропонує гравцеві гра. Термін частіше вживається в контексті комп'ютерних ігор.

Геймплей безумовно не відноситься до таких компонентів гри, як графіка і звуковий супровід. Він являє собою патерн взаємодії гравця з грою на підставі її правил, визначає зв'язок між гравцем і грою, пропонований ігровий виклик і способи його подолання, сюжет як участь в ньому гравця. Геймплей являє собою компонент, який робить цю форму творчості унікальної - без нього гра перестає бути грою.

Розберемо зіставні частини ігрового процесу та їх взаємодію в умовах імплементації розпізнання жестів.

2.4.1 Визначення ігрової механіки

Ігрова механіка (англ. Game mechanics) - набір правил і способів, який реалізує певним чином деяку частину інтерактивної взаємодії гравця і гри. Всі безліч ігрових механік гри формують конкретну реалізацію її ігрового процесу.

Точного визначення терміна немає, але в той же час думки сходяться в тому, що ігровий процес (геймплей) описує гру в загальному випадку, а ігрові механіки дозволяють конкретно реалізувати те, як гра функціонує. Подання ігрового процесу в термінах ігрових механік дозволяє не тільки успішніше вирішувати питання комунікації в команді, а й формалізувати, тестувати і покращувати ігровий процес під час його розробки.

Ряд підходів до визначення ігрових механік розділяє такі поняття, як правила гри (як моделюється ігровий світ) і дії, доступні гравцеві (способи взаємодії гравця з цим світом). У цих випадках під ігровою механікою розуміється останнє - безліч дій і способів, доступних гравцеві. У той же час є визначення, що використовують обидва ці поняття: «будь-яка частина системи правил гри, яка покриває один і тільки один спосіб інтерактивної взаємодії, що відбувається під час гри; може бути як загальної, так і специфічної ... все механіки в сукупності задають правила гри »[33].

Одним з підходів є опис в термінах об'єктів і їх методів, або агентів. Тут ігрова механіка є дією, яке відбувається над системою в якомусь стані. Тобто ігрова механіка визначається як «методи, які викликаються агентами» або «методи, спроектовані для інтерактивної взаємодії зі станом гри» [34]. В цьому випадку ігрові механіки описуються дієсловами: бігти, стрибати, стріляти, їхати верхи, перезарядити зброю, віддати наказ, поміняти зброю, поглянути на деякий місце, і так далі [32]. Узагальнюючи, ігрові механіки - це правила, які зачіпають гравців, аватари, ігрові частки, ігрове стан, поле огляду і описують всі способи зміни ігрового стану.

2.4.2 Різновиди ігрових механік

Аналізуючи спеціалізовану літературу з гейм дизайну а також різновиди популярних ігор можна виокремити такі різновиди ігрових механік:

— досягнення (Achievement), віртуальне або фізичне втілення будь-якого звершення. Зазвичай їх розглядають як нагороди (до прикладу значок, рівень, нагорода, бали, по суті все, що можна прийняти за нагороду може бути ачівкі);

— розпоряджувальна динаміка (Appointment Dynamic), динаміка для досягнення успіху в якій, необхідне повернення в заздалегідь певний момент часу для вжиття будь-які дії. Розпоряджувальна динаміки часто об'єднує міцна зв'язок з режимами інтервальних винагород або виборчих (до прикладу при поверненні в гру в певний час, користувач отримує який буст);

— механіка уникнення (Avoidance), уникнення - акт, що спонукає гравця до діяльності, по засобів не винагороди, а уникнення покарання. Викликає ряд послідовних дій узгоджених за часом з розкладом (до прикладу тварини натискають важіль кожні 30 секунд, щоб не отримати удар струмом);

— механіка поведінкового контрасту (Behavioral Contrast), поведінковий контраст – це теорія, яка визначає те, як сильно може змінюватися поведінка відповідно до змін очікувань (до прикладу за виконання квесту гравцеві дають 100 досвіду в перший раз. Надалі - по 5000. При виконанні цього ж квесту, скажімо, в десятий раз - йому дають 100 досвіду. Як наслідок - швидше за все гравець перестане виконувати цей квест).

— поведінковий імпульс (Behavioral Momentum), поведінковий імпульс – це тенденція в поведінці гравців до продовження виконання тих дій, якими вони займалися раніше (прикладом може стати фарм і гринд в різних іграх, тобто постійне виконання одного і того ж дії, в надії отримати бажаний результат / нагороду);

— вдячна продуктивність (Blissful Productivity), ідея про те, що в процесі гри ви відчуваєте себе щасливіше важко працюючи, ніж відпочиваючи і розслабляючись;

— теорія каскадної подачі інформації (Cascading Information Theory), теорія говорить про те, що для досягнення належного рівня розуміння в кожному описовому моменті гри інформація повинна подаватися мінімально можливими фрагментами;

— ланцюги подій (Chain Schedules), практика зв'язування винагород з серіями непередбачених ситуацій. Гравці схильні приймати це як прості індивідуальні події. Відкриття одного кроку в послідовності часто розглядається гравцем як окреме нагородження;

— відлік (Countdown), механіка, в якій гравець наділений обмеженою кількістю часу, для здійснення яких-небудь дій. Створює графік активності, який примушує первісну активність гравця драматично збільшуватися, поки не скінчиться час, і не відбудеться примусове завершення;

— стримуючі фактори (Disincentives), ігровий елемент, який використовує штраф (або зміну ситуації) для того, щоб викликати поведінкові зміни;

— посилення (Reinforcer), винагорода, що отримується в разі виконання очікуваної дії в тричастинній парадигмі режимів нагородження;

— весело один раз, весело завжди (Fun Once, Fun Always), концепція того, що дія приносить задоволення, скільки б раз воно не повторювалося. В основному це має відношення до найпростіших прикладів активності. Також, дуже часто існує обмежувальний поріг для загального рівня задоволення, принесеного одним дією.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ІГРОВОЇ МЕХАНІКИ НА ОСНОВІ РОЗПІЗНАННЯ ЖЕСТІВ

3.1 Особливості реалізації архітектури середовища розпізнавання жестів

Середовище розпізнавання жестів складається з ядра та модулів (плагінів). Ці складові підключаються у динамічному режимі. Кожен модуль виконує свою вузькоспеціалізовану задачу, звертаючись при необхідності до інших модулів системи. Ядро підтримує інструментарій міжмодульної взаємодії, а також забезпечує початкове завантаження (старт) та конфігурацію системи.

Кожен модуль має унікальний строковий ідентифікатор (ім’я та ID). Він також формує набір інтерфейсів взаємодії з ним для інших компонентів системи. Кожен інтерфейс модуля має ім’я та може бути застосований до інших модулів. Таким чином, для отримання будь-якого інтерфейсу модуля необхідно знати його ім’я та ідентифікатор інтерфейсу взаємодії.

Розвиваючи (2.7) можна вивести рекурентне рівняння (3.1):

$$g_t(\tau) = min \begin{bmatrix} g_{t-1}(\tau-1) + 3d_t(\tau) \\ g_{t-1}(\tau-2) + 2d_t(\tau-1) + d_t(\tau) \\ g_{t-2}(\tau-1) + 3d_t(\tau) + 3d_t(\tau) \end{bmatrix} \quad (3.1)$$

де τ – повторювані шаги обробки;

t – номер кадру зчитування;

d_t – нелінійна дистанція розпізнавання;

g – рекурсивне рівняння динамічного програмування.

Зводячи рівняння класифікатору жестів до розпізнавання вже одного кадру зображення отримаємо (3.2):

$$H_t(x) = sign(\sum_{\tau=1}^{t} \alpha_\tau h_\tau(x)) \quad (3.2)$$

де $H_t(x)$ – результат розпізнавання на t-кадрі;

α_τ – вага окремого класифікатору;

$h_\tau(x)$ – класифікатор фрему на t-кадрі.

У рівнянні (3.2) вагу окремого класифікатору можна виразити через (3.3):

$$\alpha_\tau = \ln(1 - \varepsilon_t) \varepsilon_t^{-1} / 2 \quad (3.3)$$

де α_τ – вага окремого класифікатору;

ε_t - коефіцієнт помилок.

На базовому рівні реалізація алгоритму розпізнавання жестів як хмар з точок у версії PDollarGestureRecognizer складається з наступних класів:

– Point , що реалізує 2D точку, яка виставляє властивості X, Y та StrokeID. StrokeID - індекс обведення, якому належить точка (наприклад, 0, 1, 2, ...), яка заповнюється підрахунком подій вниз / вгору ручки (див. рис 3.1).

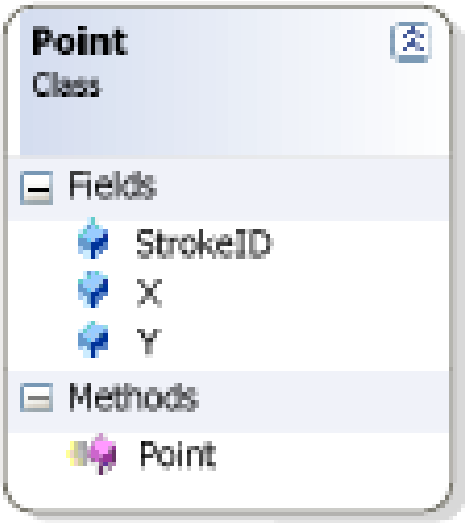

Рисунок 3.1 – Діаграма класу Point

```
[Serializable]
public class Point
{
    public float X, Y;
    public int StrokeID;

    public Point(float x, float y, int strokeId)
    {
        this.X = x;
        this.Y = y;
        this.StrokeID = strokeId;
    }
}
```

Атрибут [Serializable] вказує на те, що даний клас є серіалізуємими даними, тобто такими, що можуть бути збережені та відтворені універсальним ігровим рушієм Unity.

– Gesture. Клас, що реалізує жест, як абстракцію у вигляді хмари точок (тобто не упорядкований набір точок). Жести нормалізуються щодо масштабу, переводяться на походження та перетворюються у фіксовану кількість 32 точок (див. рис 3.2).

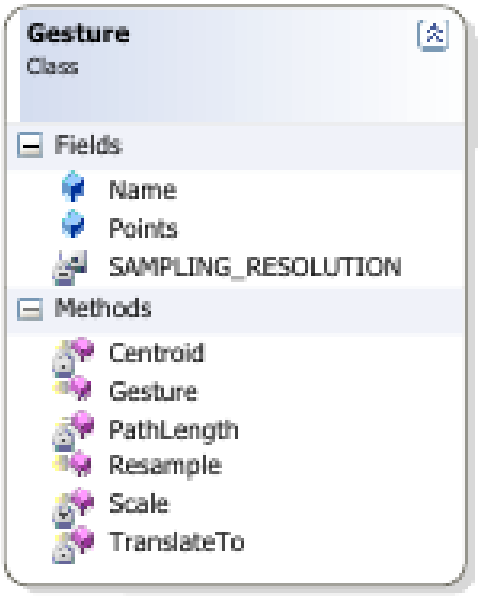

Рисунок 3.2 – Діаграма класу Gesture

Серед методів, що являють собою основу для виконання можемо навести метод Scale, що здійснює нормалізацію масштабу зі збереженням форми в [0..1] x [0..1] :

```
private Point[] Scale(Point[] points)
{
     float minx = float.MaxValue, miny = float.MaxValue, maxx =
     float.MinValue, maxy = float.MinValue;
      for (int i = 0; i < points.Length; i++)
      {
          if (minx > points[i].X) minx = points[i].X;
          if (miny > points[i].Y) miny = points[i].Y;
          if (maxx < points[i].X) maxx = points[i].X;
          if (maxy < points[i].Y) maxy = points[i].Y;
      }

      Point[] newPoints = new Point[points.Length];
      float scale = Math.Max(maxx - minx, maxy - miny);
      for (int i = 0; i < points.Length; i++)
          newPoints[i] = new Point((points[i].X - minx) /
           scale,(points[i].Y-miny)/scale,points[i].StrokeID);

      return newPoints;
  }
```

Ще одним методом є TranslateTo, що переміщує масив точок на величину p:

```
     private Point[] TranslateTo(Point[] points, Point p)
        {
            Point[] newPoints = new Point[points.Length];
            for (int i = 0; i < points.Length; i++)
                newPoints[i] = new Point(points[i].X - p.X,
points[i].Y - p.Y, points[i].StrokeID);
            return newPoints;
        }
```

Наступним класом, що потрібен для реалізації алгоритму є PointCloudRecognizer, клас що імлементує $P 2-D розпізнавач жестів-рухів

розроблений для швидкого прототипування користувальницьких інтерфейсів на основі жестів. $ P - це перший збірник хмарних точок у $ - сімействі розпізнавачів (див. рис 3.3).

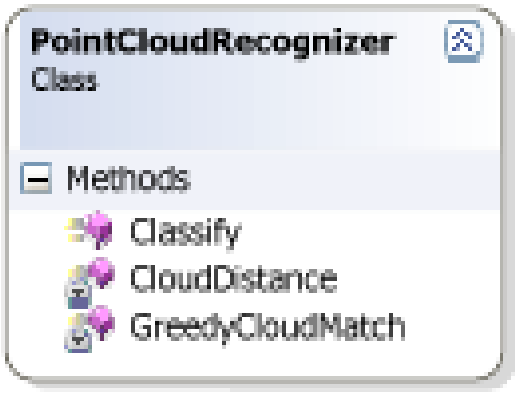

Рисунок 3.3 – Діаграма класу PointCloudRecognizer

Основна функція розпізнавача $ P реалізується Classify. Класифікує жест кандидата проти набору навчальних зразків. Повертає клас найближчого сусіда в навчальному наборі. Приведемо програмну реалізацією цієї функції на язику програмування C# :

```
public static string Classify(Gesture candidate, Gesture[]
trainingSet, out float conformity)
   {
       float minDistance = float.MaxValue;
       string gestureClass = "";
       foreach (Gesture template in trainingSet)
       {
         float dist = GreedyCloudMatch(candidate.Points,
template.Points);
           if (dist < minDistance)
           {
               minDistance = dist;
               gestureClass = template.Name;
           }
       }

       conformity = minDistance;
```

```
        if(minDistance < 0.05f)
          gestureClass = "Nothing";


        return gestureClass;
    }
```

Її основною задачею є отримання найближчої дистанції обраного масиву точок до тих базових векторів з точок, що були введені на етапі навчання. Функцією приймається рішення, чи в достатній кількості відповідає отриманий результат мінімально приємливим.

Для наглядності приведемо код обрахування «подібності» масиву певного ввідного масиву точок щодо масиву початково введеного масиву символів:

```
private static float GreedyCloudMatch(Point[] points1, Point[] points2)
        {
             int n = points1.Length; // the two clouds should have the
    same number of points by now
             float eps = 0.5f;       // controls the number of greedy
             search trials (eps is in [0..1])
            int step = (int)Math.Floor(Math.Pow(n, 1.0f - eps));
            float minDistance = float.MaxValue;
            for (int i = 0; i < n; i += step)
            {
                              float dist1 = CloudDistance(points1,
                points2, i);   // match points1 --> points2 starting with
                index point i
                 float dist2 = CloudDistance(points2, points1, i); //
                match points2 --> points1 starting with index point i
                 minDistance = Math.Min(minDistance, Math.Min(dist1,
                 dist2));
            }
            return minDistance;
        }
```

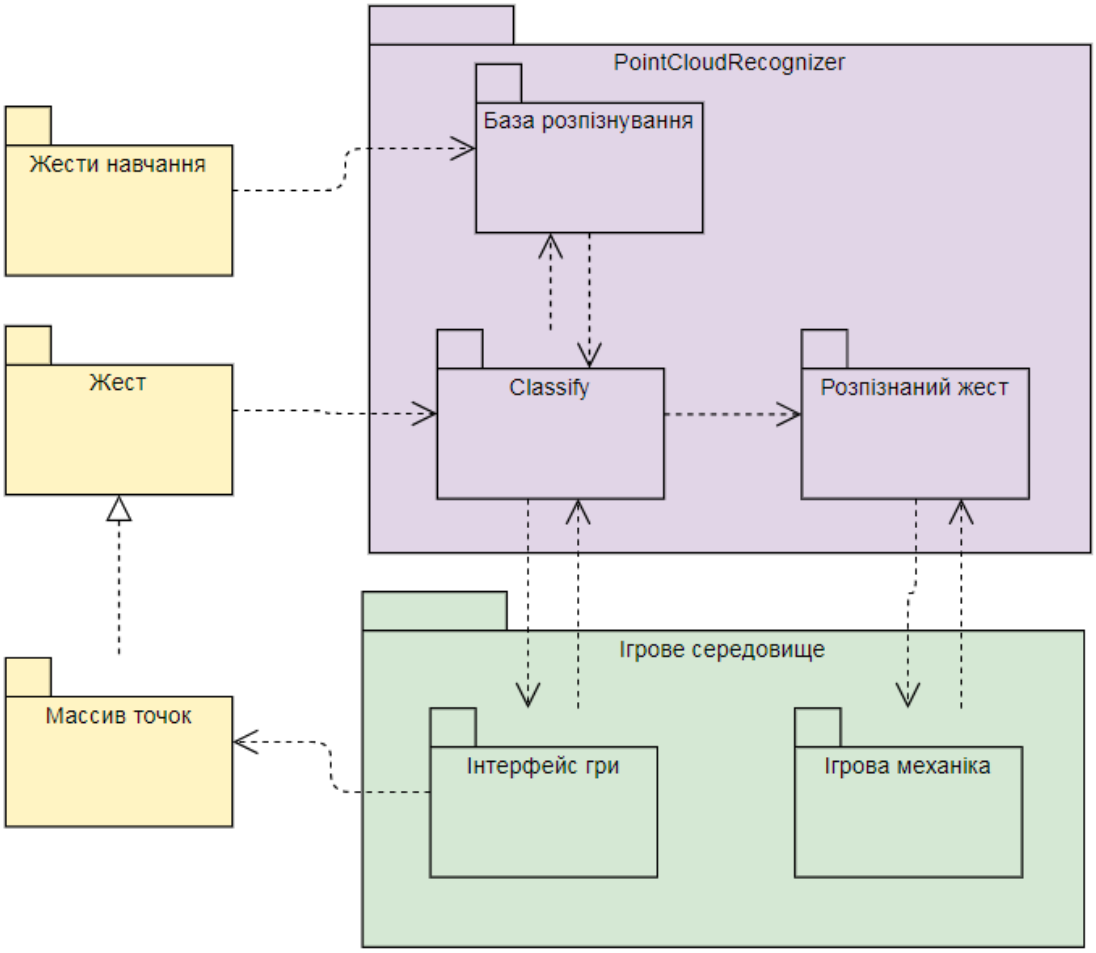

Рисунок 3.4 – Потік обробки розпізнання жестів.

Підсумований потік розпізнання жестів – від вводу жесту користувача через інтерфейс користувача і наявність бази жестів, що була створена на етапі введення навчальної бази (див. рис 3.4).

Жести навчання повинні бути введені на етапі розробки проекту. Зі створенням бази її потрібно помістити на клієнт чи залишити на стороні серверного оточення.

Інтерфейс користувача скомпонований таким чином, щоб полегшити введення жестів, показ вже введених жестів а також зворотна реакція на ввід користувача.

3.2 Практична реалізаціяя ігрового програмного продукту

Перейдемо до огляду практичної реалізації ігрового програмного продукту на основі механіки зчитування та розпізнавання жестів на основі ігрового рушія Unity3d.

Розглянемо програмну архітектуру (див. рис 3.5).

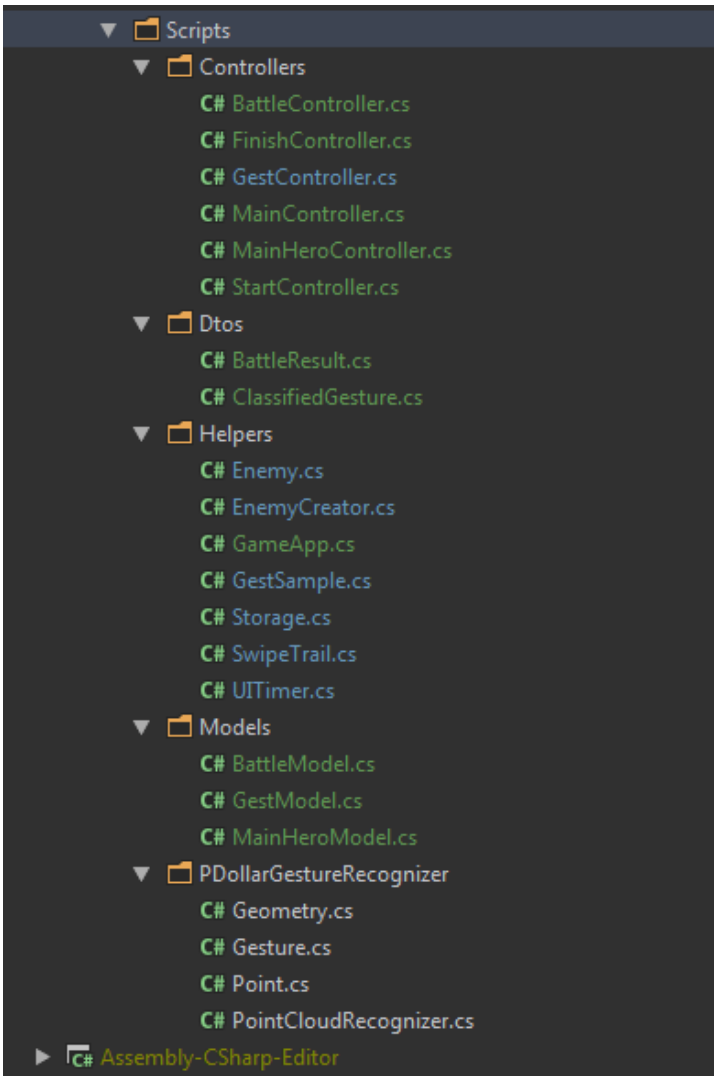

Рисунок 3.5 – Архітектура ігрової програми

Як можна бачити зі схеми, архітектура програми складеться за шаблоном MVC (model-view-controller) - модель-відображення-контролер. Це спосіб організації коду, який передбачає виділення блоків, що відповідають за вирішення

різних завдань. Один блок відповідає за дані додатки, інший відповідає за зовнішній вигляд, а третій контролює роботу додатка. Компоненти MVC:

— модель - це компонент відповідає за дані, а також визначає структуру програми. В нашій програмі модельні класи зосереджені у директорії Models та представлені наступними класами – BattleModel (безпосередньо відповідає за ініціацію, перетворення та взаємодіє даних під час проведення 1 битви користувачем додатку). GestModel – абстрагує данні пов’язані безпосередньо з розпізнанням та класифікацією введених жестів. MainHeroModel – відповідає за зберігання, трансформацію данних, що пов’язані з ігровим аватаром головного героя, та його взаємодію з іншими компонентами гри;

— відображення - це компонент, що відповідає за взаємодію з користувачем. Тобто його основна задача - визначення зовнішнього вигляду програми і способи його використання. Оскільки ігровий рушій Unity3d надає потужний інструмент для створення, візуалізації та налаштування вигляду кінцевого продукту, даний компонент реалізується не у вигляді класів, а у вигляді спеціалізованих файлів;

— контролер - цей компонент відповідає за зв'язок між model і view. Код компонента controller визначає, як програмний продукт реагує на дії користувача. По суті, це мозок MVC-додатків. В нашій грі вони представлені найбільшою різноманітністю класів, що підтверджує їх виняткову та спеціалізовану роль у роботі кожного з етапів взаємодії користувача з додатком. Так класи StartController та FinishController відповідають за коректне відображення початого екрану а також екрану завершення битви, оброблюють ввід команд користувача та завантажують наступні екрани згідно потому взаємодії гравця та гри. BattleController поєднує та координує роботу GestController (перетворює візуальних користувача ввід з відповідного компонента View на таких набір даних, що їх зможе сприйняти та обробити цільовий компонент Model) та MainHeroController (що передає данні зміни стану моделі на відповідний компонент View, що представлений аватаром головного героя а також

елементами графічного інтерфейсу, що допомагають користувачеві оперувати отриманою інформацією та виконувати відповідні до мінливих обставин дії).

Найбільш очевидна перевага, яку ми отримуємо від використання концепції MVC - це чіткий поділ логіки подання (інтерфейсу користувача) і логіки програми. Підтримка різних типів користувачів, які використовують різні типи пристроїв є спільною проблемою наших днів. Наданий інтерфейс повинен відрізнятися, якщо запит приходить з персонального комп'ютера або з мобільного телефону. Модель повертає однакові дані, єдина відмінність полягає в тому, що контролер вибирає різні види для виведення даних. Крім ізолювання видів від логіки додатки, концепція MVC істотно зменшує складність великих додатків. Код виходить набагато більш структурованим, і, тим самим, полегшується підтримка, тестування і повторне використання рішень.

Особливістю конкретної реалізації MVC не як суворого патерну, а скоріше методологічно-організаційної методології в конкретних язикових умовах, а також методах, особливостях і можливостях, що їх надає ігровий рушій Unity3d ми хотіли виділити наступні риси:

— компонент Відображення не знає про компоненти Контролерів та Моделі. Його головний обов’язок – візуалізація даних, що їх надаватиме контролер, та виклик певних колбеків контролеру у відповідь на дії користувача;

— компонент Модель не знає про компоненти Відображення та Контролери. Модель інкапсулює в собі всі данні, а також методи роботи з ними, що виведенні до публічного API через імплементацію конкретних інтерфейсів. Клас Моделі має поле івенту (event), на який може підписатися Контролер для отримання зворотної інформації про те, які данні в моделі змінилися;

— компонент Контролер має інформацію як про Модель так і про Відображення. З функціональної точки зору а також потому управління клас конкретного контролеру поділяється на дві половини – обробники колбеків від Відображення (тут проводиться перевірка даних, їх групування, вирішується які методи Моделі повинні бути визванними. Тут відбувається алгоритмічна обробка введеної інформації для її подальшого використання) та обробники івентів від

Моделі (реалізація непрямого способу взаємодії, отримання зворотної інформації від Моделі з урахуванням принципу Найменшого Зв’язування компонентів. В цій частині коду переважно відбувається проста передача даних на Відображення для їх подальшої візуалізації).

3.2.1 Огляд інтерфейсу та візуальної складової

Розглянемо вигляд основного вікна бойової сцени. Він складається з таких частин як Графічний інтерфейс користувача (graphical user interface, GUI - система засобів для взаємодії користувача з комп'ютером, заснована на представленні всіх доступних користувачеві системних об'єктів і функцій у вигляді графічних компонентів екрану), власне ігрової сцени, що є засобом візуалізації та трансформації ігрових подій.

Графічний інтерфейс користувача складається з верхньої та нижньої половин (див. рис 3.6). У нижній частині розміщена інформація про основні жести, що їх може застосовувати гравець. Інформація наведена у вигляді графічних зображень (іконок), що на початку сцени є затемненими і поступово відкриваються. В даній реалізації відкриття здійснюється з плином часу, проте найбільш розповсюдженим прийомом у галузі розробки ігрових додатків є поступове відкриття нових жестів/здібностей з ігровим прогресом гравця (кількістю вбитих ворогів, досягненням певної мети тощо). Іконки відкритих жестів/здібностей потрібні для того, щоб нагадувати користувачеві, які йому в даний час доступні для введення жести, та яку користь він може отримати від використання

Рисунок 3.6 – Вікно ігрової сцени

даного жесту (так жести лівої половини знадобляться для нанесення втрат противникам, жести правої половини поповнюють рівень магічної енергії, що витрачається при застосуванні першої груп навичок).

Верхня половина графічного інтерфейсу користувача має за мету інформувати гравця про основні його данні – рівень життя (червона смужка з червоним сердечком і цифровим відображенням рівня життя – коли воно зменшиться до 0 бій закінчується поразкою) та рівнем магічної енергії – надважливим індикатором здібності наносити вдари по ворогам (за відсутності достатньої кількості магічної енергії ігровий аватар не зможе нанести вдар і завдати шкоди навіть попри правильно введений жест атаки).

Між верхнім та ніжним графічними інтерфейсами користувача розміщується основна ігрова локація. З графічної точки зору вона також поділена

у вертикальному напрямку – там в нижній половині екрану ми бачимо більш темний ландшафт, у верхній частині екрану оточення ігрового полю становиться більш світлим. Це дає користувачеві підказку стосовно того, з якого боку будуть виходити вороги (з нижньої половини екрану) та в який частині розміщений головний аватар користувача.

Все поле в цілому є основою, на якому користувач може дотиком вводити лінії жестів. Це саме «серце» гри – гравець проводить по екрану, графічний інтерфейс підрисовує лінії майбутніх жестів та по команді користувача відправляю записані данні на аналіз та розпізнавання. Задачею даного виду графічного інтерфейсу користувача є необхідність відобразити ввід користувача, щоб дати адекватну оцінку того, що користувач ввів і на яку реакцію систему він повинен очікувати.

Слід також додати, що кожен з варіантів дії, що їх викликають введені користувачем жести, певним чином візуалізується. Це з одного боку урізноманітнює гру, робить її інтерактивною, привабливою, цікавою. А з іншого боку дозволяє певним чином вибирати стратегії – бо кожен жест чи дія відрізняється від іншого дозволяючи будувати певні патерни взаємодії.

3.2.2 Огляд програмної складової

Великий обсяг програмної складової не дозволяє в повній мірі описати програмну реалізацію даного виду програмного продукту. Проти ми намагатимуся зупинитися на найбільш характерних, знакових чи складних елементах реалізації.

```
public class MainController
{
private StartController _start;
private BattleController _battle;
private FinishController _finish;
```

```
public MainController()
{
_start = GetSceneRootComponent<StartController>();

_start.BattleStart += OnBattleStarted;
_start.BattleStart += CleanStartController;
}

private void CleanStartController()
{
_start.BattleStart -= OnBattleStarted;
_start.BattleStart -= CleanStartController;
_start = null;
}…
}
```

Частина програмної реалізації коду головного контролеру нашого електронного додатку. Можна звернути увагу на те, що даний контролер має в свою чергу посилання на три інші контролери – стартового екрану, власне битви, а також екрану завершення битви. Це є власне 3 стейти ігрового процесу що циклічно змінюють один одного. Ініціалізація стейту починається з пошуку та отримання вказівника на конкретний елемент контролеру в сцені (спеціально підготовлений файл у межах функціональності рушія Unity3d). Потім використовуючи патерн «спостерігач» ми підписуємося на певні події, що їх у собі інкапсулює певний контролер. Самому контролеру в цей час передається повне управління процесом – він сам ініціалізує потрібні йому данні а також, відстежуючи логіку завершення своїх функцій деініціалізує власні змінні та підготовлює в межах своєї відповідальності перехід до іншого стейту.

Можна узагальнити, що в даній реалізації головний контролер нагадує патерн «стан», і являє собою в певній мірі реалізацію стейт-машини. Власне всі обов’язки головного контролеру зводяться до «перемикання» одного стану на інший, підписування на деякі події та обов’язкове відписування від цих подій з деактивацією самого стейту.

Наступним представником є метод CreateGestureSample класу GestController:

```
private Gesture CreateGestureSample(string gestureName = "")
    {
      var touches = new List<Point>();
      var index = 0;

      foreach(var swipe in _swipes)
      {
        foreach(var touch in swipe.GetTouches())
        {
          if(swipe.GetTouches().Count < 5)
            continue;

          var pos = touch + swipe.transform.position;
          touches.Add(new Point(pos.x, pos.y, index));
        }

        index++;

        swipe.DestroyObj();
      }

      _swipes = new List<SwipeTrail>();

      if (touches.Count == 0)
        return null;

      return new Gesture(touches.ToArray(), gestureName);
    }
```

Тут ми маємо аналіз свайпів (безперервних ліній), їх об’єднання з урахуванням відносної відстані один від одного, перетворенням на масив об’єктів Vector3. Данні згруповані у такому форматі є зрозумілими для класу моделі аналізу даних розпізнавання жестів. В даному класі контролер перетворює данні візуального компонента (View) на данні, що будуть передані на обробку.

ВИСНОВКИ

Поява нових технологічних рішень у галузі розробки ігрових систем, перед усім пов’язаних з розвитком і поширенням мобільних комп’ютерних пристроїв, зумовили якісний і кількісний ріст і розвиток різноманітних напрямів ігрового виробництва. Зростаюча конкуренція на ринку, зумовлена величезним зростанням обсягів доходів, призвело до нагальної потребі у створенні нових ігрових механік, комбінації з уже існуючими підходами а також імплементації у ігровий простір таких технологій, що початкове розвивалися в інших галузях програмування або сферах розробки прикладного програмного забезпечення.

Однією з таких технологій, що має певні перспективи до впровадження у ігровій механіці є практичні опрацювання у сфері розпізнання та класифікації жестів. Проте аналіз ринку ігрових програмних продуктів показав відсутність як широкого різноманіття таких продуктів, так і повноцінних бібліотек, що б могли перебрати на себе відповідальність за введені жести.

В результаті виконання атестаційної роботи отримані наукові та практичні результати.

1. Розглянуто види жестів, їх класифікацію, пригідність тих чи інших видів жестів до імплементації у ігровому середовищі.

2. Дан огляд ринку мобільних ігрових систем, його історія, розвиток, сучасний стан та виявленні тенденції, що будуть рухати цю сферу у найближчі роки.

3. Проведено аналіз методів розпізнання та класифікації введених жестів, їхню можливість до застосування на не надто потужних мобільних комп’ютерах.

4. Запропоновано принципову схему інтеграції технології розпізнавання жестів у динамічний ігровий процес, що дозволяє урізноманітнити ігровий досвід та зберегти динамічний темп зміни подій.

5. Розроблений програмний засіб, що реалізує запропоновану схему. Проведена апробація та узгодження компонентів системи. Проведенні

оптимізаційні прийоми, що дозволяються ігровому програмному продукти бути розгорнутим на мобільному пристрої. Виконана робота по естетичному оформленню та балансуванню легкості/складності в процесі проходження рівнів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Абакумов В. Г. Интерпретация движений рук расширяет возможности интерактивного управления в интеллектуальных системах/В.Г. Абакумов, Е.Ю. Ломакина//Природные и интеллектуальные ресурсы Сибири. 2009. С. 199-202.

2. Statista. Number of available gaming apps at Google Play from 1st quarter 2015 to 1st quarter 2020. URL: https://www.statista.com/statistics/780229/number-of-available-gaming-apps-in-the-google-play-store-quarter/ (дата звернення: 13.05.2020).

3. Statista. Number of available apps in the Apple App Store from 2008 to 2019. URL: https://www.statista.com/statistics/268251/number-of-apps-in-the-itunes-app-store-since-2008/ (дата звернення: 06.02.2020).

4. App Annie. The State of Mobile in 2019 – The Most Important Trends to Know. URL: https://www.appannie.com/en/insights/market-data/the-state-of-mobile-2019/ (дата звернення: 10.02.2020).

5. Statista. Number of active users of Pokémon Go worldwide from 2016 to 2020. URL: https://www.statista.com/statistics/665640/pokemon-go-global-android-apple-users/ (дата звернення: 14.02.2020).

6. Video Games Stats. Pokemon Go Statistics and Facts. URL: https://videogamesstats.com/pokemon-go-statistics-facts/ (дата звернення: 14.02.2020).

7. Unity. Main Page. URL: https://unity.com/ (дата звернення: 15.02.2020).

8. Java. Что такое J2ME или JavaME. URL: https://www.java.com/ru/download/faq/whatis_j2me.xml (дата звернення: 17.02.2020).

9. Android Developers Blog. Android Market: Now available for users. URL: https://android-developers.googleblog.com/2008/10/android-market-now-available-for-users.html (дата звернення: 21.02.2020).

10. Newzoo. Global Games Market Report 2019 | Light Version. URL: https://newzoo.com/insights/trend-reports/newzoo-global-games-market-report-2019-light-version/ (дата звернення: 25.02.2020).

11. Business of Apps. App Download and Usage Statistics (2019). URL: https://www.businessofapps.com/data/app-statistics/ (дата звернення: 10.03.2020).

12. Абакумов В. Г. Ломакіна Є. Ю. Автоматичне розпізнавання жестів в інтелектуальних системах//Штучний інтелект. 2010. № 3. С. 269-273.

13. Авдеев Д. А., Кулишова Н. Е. Сучасні методи вирішення проблем розпізнавання жестів людини у реальному часі.//Біоніка інтелекту: наук.-техн. журнал. 2016. №1(86). С. 102-107.

14. Бондарев В. Н., Аде Ф. Г. Искусственный интеллект. – Севастополь: СевНТУ, 2002. – 613 с.

15. Гонсалес Р., Вудс Р. Цифровая обработка изображений. - Москва: Техносфера, 2005. - 1072 с.

16. Зайцева Г. Л. Жестовая речь. Дактилология: учеб. для студ. Высш. учеб. заведений. - Москва: Гуманит. Изд. центр ВЛАДОС, 2004. - 192 с.

17. Форсайт Д., Понс Ж. Компьютерное зрение. Современный подход. – Москва: Вильямс, 2004. - 926 с.

18. Местецкий Л. М. Непрерывная морфология бинарных изображений: фигуры, скелеты, циркуляры. – Москва: ФИЗМАТЛИТ, 2009. – 288 с.

19. Мурлін А. Г. Алгоритм і методи виявлення і розпізнавання жестів руки на відео в режимі реального часу//Научный журнал КубГАУ. 2014. №97(03). С. 162.

20. Хуанг Т. С., Эклунд Дж-О., Нуссбаумер Г. Дж. Быстрые алгоритмы в цифровой обработке изображений. – Москва: Радио и связь, 1984. – 224 с.

21. Шапиро Л., Стокман Д. Компьютерное зрение/Пер. с англ. - Москва: БИНОМ. Лаборатория знаний, 2006. - 752 с.

22. Єрохін А. Л., Ледньов С. Н. Методи розпізнавання жестів на основі даних трьохосьових акселерометрів Android пристроїв.//Вісник Національного технічного університету "ХПІ". 2017. № 21 (1243). С. 54.

23. Gesture Recognition. URL: https://en.wikipedia.org/wiki/Gesture_recognition (дата звернення: 22.03.2020).

24. Feng Hong, Shujuan You, Meiyu Wei, Yongtuo Zhang, and Zhongwen Guo MGRA: Motion Gesture Recognition via Accelerometer//Sensors. 2016. Vol. 16 (4). P. 67.

25. Motion Sensors and Android Developers. 2017. Accessed: 12 March 2017. URL: http://developer.android.com/intl/ru/guide/topics/sensors/sensors_motion.html (дата звернення: 26.03.2020).

26. Sezgin T. M., Davis R. HMM-based efficient sketch recognition// Proceedings of the 10th international conference on Intelligent user interfaces. 2005. IUI '05. P. 281–283.

27. Rubine D. Specifying gestures by example//ACM SIGGRAPH Computer Graphics. 1991. P. 329–337.

28. Kara L. B., Stahovich T. F. Hierarchical parsing and recognition of hand-sketched diagrams//UIST. 2004. P. 13–22.

29. Hammond T., Paulson B. Recognizing sketched multistroke primitives//ACM TIIS 1. 2011. P. 34.

30. Senin P. Dynamic time warping algorithm review. – Honolulu: Information and Computer Science Department. University of Hawaii at Manoa, 2008. - 23 p.

31. QA Testlab. Ігрові механіки: види і особливості тестування. URL: https://training.qatestlab.com/blog/technical-articles/game-mechanics/ (дата звернення: 02.04.2020).

32. Rouse, Richard. Game Design: Theory & Practice. USA: Wordware Publishing, 2004. — 698 p.

33. 3D News. История оригинальных игровых механик. Свежие решения и нестандартный подход. URL: https://3dnews.ru/981901/istoriya-originalnih-igrovih-mehanik-svegie-resheniya-i-nestandartniy-podhod (дата звернення: 04.04.2020).

34. For Brain. Ігрові механіки. URL: https://4brain.ru/gamification/igrovye-mehaniki.php (дата звернення: 04.04.2020).

35. Google Developers. MediaPipe on the Web. URL: https://developers.googleblog.com/2020/01/mediapipe-on-web.html (дата звернення: 05.04.2020).

36. Google AI Blog. On-Device, Real-Time Hand Tracking with MediaPipe. URL: https://ai.googleblog.com/2019/08/on-device-real-time-hand-tracking-with.html (дата звернення: 07.04.2020).

37. Github. Mediapipe. URL: https://github.com/google/mediapipe (дата звернення: 07.04.2020).

38. Elena Sanchez-Nielsen, Luis Antyn-Canalos, Mario Tejera Hand Getsure recognition for Human Machine Intercation//In Proc. 12th International Conference on Computer Graphics, Visualization and Computer Vision: WSCG. 2004. P. 1379–1384.

39. Листровой С.В., Гуль А.Ю. Метод решения задачи о минимальном покрытии на основе рангового подхода//Электронное моделирование. 1999. №1 С. 58–70.

40. Хемди А. Таха Введение в исследование операций, 7-е издание: Пер. с англ. - Москва: Издательский дом «Вильямс», 2005. - 912 с.

41. Бычков И. В., Опарин Г. А., Феоктистов А. Г., Корсуков А. С. Распределение заданий в интегрированной кластерной системе на основе их классификации//Вычислительные технологии. 2013. №2. Том 18, С. 25–32.

42. Красавина А. К. Исследование методов и алгоритмов распределения задач между исполнителями в системах управления проектами//Качество. Инновации. Образование. 2013. №11. С. 70–73.

43. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. Москва: МЦНМО, 1999. - 960 с.

44. Neumann K. Stochastic project networks. Temporal analysis, scheduling and cost minimization. - Berlin: Springer-Verlag, 1990. - 237 p.

45. HTCondor: Hight Throughput Computing. URL: https://research.cs.wisc.edu/htcondor/ (дата звернення: 10.04.2020).

46. HTCondor Version 8.0.0 Manual. University of Wisconsin-Madison: Center for High Throughput Computing, 2013. - 1053 p.

47. Amar A., Bolze R., Boix E. DIET 2.8 user's manual, 2011. URL: https://graal.ens-lyon.fr/~diet/download/doc/UsersManualDiet2.8.1.pdf (дата звернення: 10.04.2020).

48. Abdelkader A., Caniou Y., Caron E. et al. DIET 2.8. User's manual. Inria, ENS-Lyon, UCBL, SysFera, 2011. URL: https://graal.ens-lyon.fr/~diet/UsersManual-DIET2.8/UsersManual.html/ (дата звернення: 12.04.2020).

49. PBS Works 10 Enabling On-Demand Comp. URL: http://www.pbsgrid-works.com/images/solutions-en-US/PBS_10.2Highlights_letr_REVISE-WEB.pdf (дата звернення: 14.04.2020).

50. TORQUE Resource Manager Guide. URL: https://www.clusterresources. com/products/torque-resource-manager.php (дата звернення: 16.04.2020).

51. Maui Administrator's Guide//Adaptive Computing Enterprises. 2011. v.3.2. P. 287.

52. Moab Workload Manager Administrator Guide//Adaptive Computing Enterprises. 2013. v.7.2.4. P. 1136.

53. 12. Brian H. Cloud computing.//Communications of the ACM. 2008. Vol. 51(7). P. 9–11.

54. Peter Mell, Timothy Grance The NIST Definition of Cloud Computing. Recommendations of the National Institute of Standards and Technology.//National Institute of Standards and Technology Special Publication. 2011. P.7. P. 2-3.

55. Михайлов П. А., Радченко Г. И. Методы моделирования и оценки производительности облачных систем//Вестн. ЮУрГУ. Серия: Вычислительная математика и информатика. 2014. №3. Том 3. С. 109–123.

56. Батура Т. В., Мурзин Ф. А., Семич Д. Ф. Облачные технологии: основные понятия, задачи и тенденции развития//Программные продукты и системы и алгоритмы. 2014. №1. С. 64-72.

57. Willems D., Niels R., Gerven M., Vuurpijl L. Iconic and multi-stroke gesture recognition//Pattern Recognition 42. 2009. №12. P. 3303–3312.