буде меншою для схожих граней і більшою для різних. Для кожного зображення обчислюємо Евклідову відстань між вхідним зображенням і кожного із зображень, представлених в наборі даних. Яке з них має найменшу Евклідову відстань, може розглядатися як зображення тієї ж людини, що й у вхідному зображенні.

Отже, після проведення експериментів та тестування програми слід зауважити, що для досягнення позитивних результатів роботи необхідно хоча б 10 фотографій особи для навчання класифікатора. З такою кількістю фото програма вірно ідентифікує особи при різних поворотах і положеннях голови, будь-якому освітленні, при цьому достатньо лише трохи світла, аби алгоритм зміг визначити краї обличчя.

Список літературних джерел.

1. OpenCV: Home. [Електронний ресурс] – Режим доступу: https://opencv.org/.
2. Mark Nixon. Feature Extraction and Image Processing for Computer Vision 3rd Edition. Academic Press, 2012. - 632 pages
3. OpenFace - GitHub Pages. [Електронний ресурс] – Режим доступу: https://cmusatyalab.github.io/openface/

УДК 004.4(043.2)

Зінченко Б.В., студент 3 курсу спеціальності 122 «Комп'ютерні науки»
Січко Т.В., к.т.н., доцент кафедри комп'ютерних наук та інформаційних технологій

ВИКОРИСТАННЯ ДАНИХ ДЛЯ ОПТИМІЗАЦІЇ ПРОЕКТІВ ТА КОДУ .NET НА БАЗІ ІГРОВОГО РУШІЯ UNITY

Донецький національний університет імені Василя Стуса, м. Вінниця

На сьогодні в умовах великої конкуренції у сфері ігор важливим фактором є швидкодія та розмір програмного коду. Проекти оцінюються не тільки в залежності від ідеї та імені компанії, що розробила гру, але й від реалізації. Реалізація може вважатися успішною при умові, що розробнику вдалося втілити усі ідеї проекту та оминути проблеми з помилками у коді.

Першим і найважливішим видом оптимізації програмного продукту є робота з пам'яттю. Скрипти, які пишуться у міжплатформенному середовищі розробки ігор Unity використовують автоматичне керування пам'яттю. Мови програмування, такі як C і C ++, навпаки, використовують ручне управління пам'яттю - розробник може безпосередньо зчитувати і записувати дані за вказаними адресами пам'яті і він відповідальний за видалення будь-якого створюваного ним об'єкта. Наприклад, якщо створюються об'єкти в C++, то після

закінчення роботи, необхідно вручну звільнити виділену для них пам'ять. У скриптовій же мові досить написати objectReference = null;

Якщо створюється об'єкт невідомий Unity, наприклад, екземпляр класу, який ні від чого не успадковується (більшість класів або "скриптових компонентів" успадковуються від MonoBehaviour). Встановивши змінну з посиланням значення null, об'єкт буде втрачено для скрипту і Unity, вони не зможуть отримати до нього доступ і ніколи знову його не побачать, але при цьому він залишиться в пам'яті. Потім, через якийсь час відпрацює “збирач сміття” і при цьому видалить з пам'яті все, на що немає посилань. Він може це зробити, тому що в надрах збирача ведеться облік кількості посилань на кожен блок пам'яті. Це одна з причин низької швидкодії скриптових мов.

Для різних ігор можна використати різни методи оптимізації пам’яті. Для онлайн гри, важливою є швидкодія та кількість кадрів в секунду, тому слід групувати непотрібні об’єкти для їх спільного видалення. Приклад на Рисунок1.

```
if (Time.frameCount % 30 == 0)
{
   System.GC.Collect();
}
```

Рисунок 1 - Мала кількість “сміття” з швидким його видаленням

Інший метод оптимізації пам’яті підходить для ігор, де виділення пам'яті відносно рідкісне. Їх видалення можна провести під час пауз в ігровому процесі. Таким чином кількість непотрібних даних можна зробити максимально великою, не перевантажуючи пам'ять системи, що може викликати закриття програми.

```
//C# script example
using UnityEngine;
using System.Collections;

public class ExampleScript : MonoBehaviour {
    void Start() {
        var tmp = new System.Object[1024];

        // make allocations in smaller blocks to avoid them to be treated in a special way, which is designed for large blocks
        for (int i = 0; i < 1024; i++)
            tmp[i] = new byte[1024];

        // release reference
        tmp = null;
    }
}
```

Рисунок 2 - Велика кількість “сміття” з повільним його видаленням

Іншим способом отримання бажаної швидкодії є зменшення операцій на великих наборах даних.

Прикладом може бути алгоритм сортування зображений на рис 3.

```
void sort(int[] arr) {
   int i, j, newValue;
   for (i = 1; i < arr.Length; i++) {
       // record
       newValue = arr[i];
       //shift everything that is larger to the right
       j = i;
       while (j > 0 && arr[j - 1] > newValue) {
           arr[j] = arr[j - 1];
           j--;
       }
       // place recorded value to the left of large values
       arr[j] = newValue;
   }
}
```

Рисунок **3** - Метод керування великою кількістю об’єктів

Слід слідкувати за кількістю операцій у циклі, адже при неправильному алгоритмі може з’явитись надмірне виконання циклу. Тому слід змінювати будову циклу, працюючи з групами об’єктів, а не навпаки.

Розглянемо ігровий приклад, в якому присутні 100 ворогів, де пересування кожного ворога враховує пересування всіх інших ворогів. Для підвищення швидкодії слід розбити ігрове поле на осередки, записати пересування кожного ворога в сусідню клітинку і потім змусити кожного з ворогів перевіряти кілька найближчих осередків. Тоді це можна зробити в меншою кількістю операцій.

Для операцій, які використовують багато ресурсів комп’ютера чи телефона, слід застосовувати їх рідкі виклики та кешування результату.
На Рисунок 4 наведено фрагмент коду, який контролює політ кулі.

```
var speed = 5.0;

function FixedUpdate () {
   var distanceThisFrame = speed * Time.fixedDeltaTime;
   var hit : RaycastHit;

   // every frame, we cast a ray forward from where we are to where we will be next frame
   if(Physics.Raycast(transform.position, transform.forward, hit, distanceThisFrame)) {
       // Do hit
   } else {
       transform.position += transform.forward * distanceThisFrame;
   }
}
```

Рисунок **4** - Заміна методу Raycast

Отже, оптимізація є важливою, оскільки від швидкодії продукту залежить його успіх та реалізація. Слід досліджувати і знаходити нові методи оптимізації проекту.

Список літературних джерел.

1. *Документація по ігровому рушію Unity. Unity Docmentation: URL: https://docs.unity3d.com/ru/current/Manual/index.html (дата звернення: 02, 12,2020)*
2. *Unity(Рушій_гри)URL: https://uk.wikipedia.org/wiki/Unity_(%D1%80%D1%83%D1%88%D1%96%D0%B9_%D0%B3%D1%80%D0%B8)*
3. *Головний сайт рушія Unity: URL: https://unity.com/ru*

УДК 004.4(043.2)

Кузьмін О. В., студент 4 курсу спеціальності 122 «Комп'ютерні науки»
Січко Т.В., к.т.н., доцент, доцент кафедри комп'ютерних наук та інформаційних технологій

ВИКОРИСТАННЯ СУЧАСНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ У МЕДИЦИНІ

Донецький національний університет імені Василя Стуса, Україна

На сьогоднішній день майже в усій галузі охорони здоров'я запроваджені інформаційні технології. Завдячуючи цьому медицина набула на даний момент абсолютно нових Рисунок Цей процес супроводжується активними змінами в медичній теорії та практиці, пов'язаними з внесенням коректив до підготовки медичних працівників. Інформаційні технології допомагають лікарям проводити об'єктивну діагностику захворювань, накопичувати та ефективно використовувати отриману інформацію на всіх стадіях лікувального процесу і, що найбільш важливе для медичної науки, є неоціненними у науковому пізнанні.

При використанні комп'ютерів в лабораторних медичних дослідженнях в програму закладають певний алгоритм діагностики. Створюється база захворювань, де кожному захворюванню відповідають певні симптоми чи синдроми. У процесі тестування, використовуючи алгоритм, людині задаються питання. На підставі її відповідей підбираються симптоми, максимально відповідної групи захворювань. Наприкінці тесту виводиться ця група захворювань з позначенням у відсотках - наскільки це захворювання ймовірно у даного тестування. Чим вище відсотки, тим вища ймовірність цього захворювання. Зараз робляться спроби створити такий алгоритм, який би визначав не кілька, а один діагноз. Але все це поки що на стадії розробки і тестування. Взагалі, на сьогоднішній день у світі створено понад 200 комп'ютерних експертних систем [1].

Експертні системи є одним з найпоширеніших типів систем штучного інтелекту. Вони розроблялися як науково-дослідні інструментальні засоби і розглядалися як штучний інтелект спеціального типу, призначений для