УДК 004.921

АЛГОРИТМ ТА ПРОГРАМА РЕАЛІЗАЦІЯ 3-D ГРИ З ВИКОНАННЯМ ДІЙ ЗА ВИЗНАЧЕНИЙ ЧАС

В. О. Данченков
студент 1 курсу, група АКІТ–11, навчально-науковий інститут автоматики, кібернетики та обчислювальної техніки
Науковий керівник – к.т.н., доцент Я. В. Данченков

Національний університет водного господарства та природокористування, м. Рівне, Україна

У статті наведено алгоритм створення 3-D гри, де потрібно виконувати дії за певний час у конкретні інтервали, які задані фоновою музикою. Зазначено програмні засоби, які можна використовувати для її реалізації. Даний алгоритм базується на методах розробки кросплатформенних 3-D ігор.
Ключові слова: середовище Unity 3D, Unity Store, таймер відліку, музичний супровід.

В статье приведен алгоритм создания 3-D игры, где нужно выполнять действия за определенное время в конкретные интервалы, заданные фоновой музыкой. Указаны программные средства, которые можно применить для её реализации. Данный алгоритм базируется на методах разработки кроссплатформенных 3-D игр.
Ключевые слова: среда Unity 3D, Unity Store, таймер отсчета, музыкальное сопровождение.

The article presents an algorithm for creating a 3-D game where you need to perform actions for a certain time at specific intervals set by background music. The software tools that can be used for its implementation are indicated. This algorithm given is based on methods of three-dimensional crossplatform games development.
Keywords: Unity 3D environment, Unity Store, countdown timer, musical accompaniment.

Ідеєю проєкту є розробка гри, де потрібно виконувати дії за певний час у конкретні інтервали. Початок, протяжність і кінець інтервалу задані фоновою музикою. Аналогічного продукту на локальному ринку не було знайдено. На глобальному ринку конкуренцію становитиме популярна гра «Beat saber», яку було розроблено для ігрових станцій типу «Playstation» та «XBox», але продукти не мають однозначної схожості і мають різницю в інтерфейсі та реалізації, а також розраховані для різних моделей комплектацій, що суттєво відрізняються у ціні експ. [1; 2].

Новизна роботи полягає в тім, що для написання гри було використано ігровий движок Unity 3D, що дозволяє запустити розробку на більшості відомих платформ. Через його популярність можливості його використання для розробника-початківця є максимальними, а наявність великої кількості бібліотек для C# скорочує час написання коду і розробку нових функцій застосунку [3].

Проєкт має бути доступний для запуску і коректної роботи на операційній системі Windows XP, Mac OS X 10.9, Android 4.1, iOS 7, Ubuntu 12.04, SteamOS+. Підтримуватиметься відеоадаптерами із драйверами DirectX 9 і OpenGL ES 2.

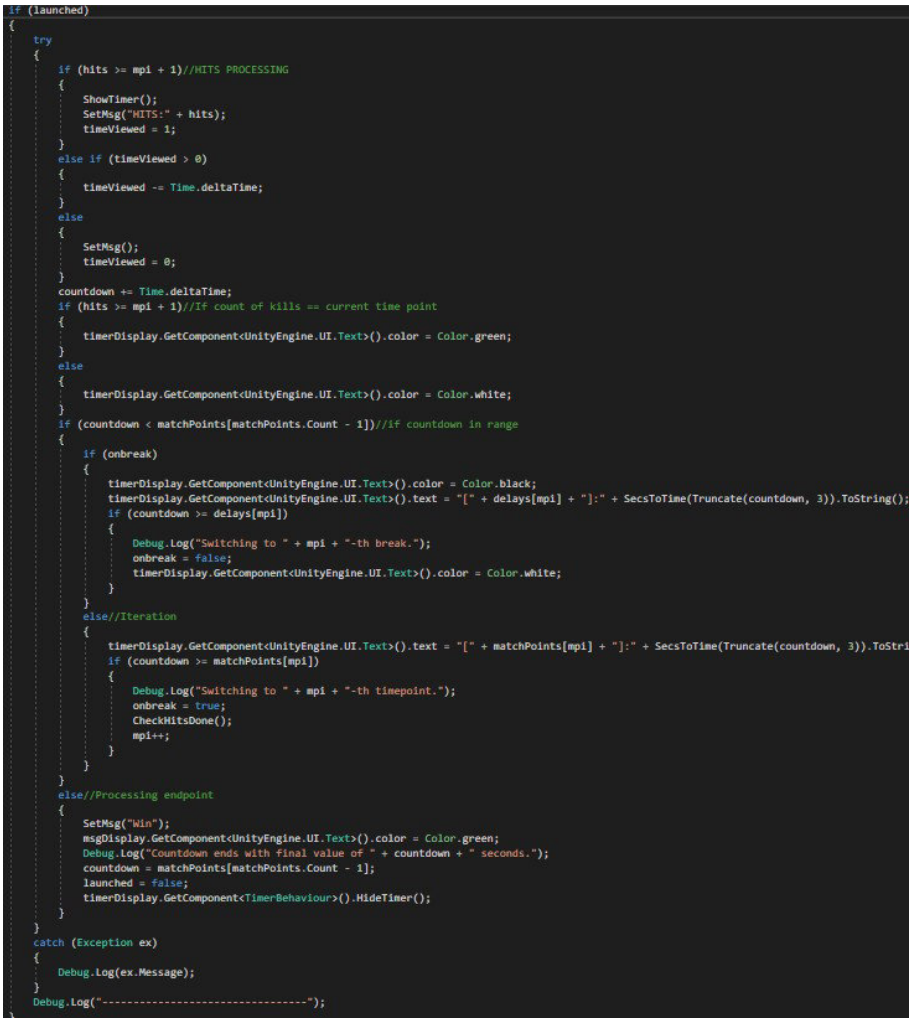
Реалізація 3D моделей потребує навичок у 3D моделюванні з використанням таких програм, як Blender, 3D Max, Maya 3D тощо. Зазвичай це трудомісткий процес, що потребує багато часу і навичок роботи із переліченими програмними продуктами, із якими не усі мають досвід роботи. Для цього у Unity Store опубліковано набір безкоштовних моделей, які можна імпортувати до будь-якого 3D проєкту Unity. Далі наведено алгоритм імпорту сторонніх пакетів зовнішніх 3D моделей до Unity 3D з Unity Store:

1. відкриваємо потрібний проєкт;
2. шукаємо потрібний пакет ассетів;
3. на сторінці доповнення натискаємо на «Add To My Assets»;
4. підтверджуємо умови отримання;
5. натискаємо «Open in Unity»;
6. переходимо до сторінки отриманих ассетів;
7. відкриється менеджер пакетів проєкту;
8. завантажуємо обраний та імпортуємо його;
9. перетягуємо із внутрішнього файлового провідника потрібні моделі до ігрового простору.

Пошук анімацій також відбувається у Unity Store. Відповідно до алгоритму, що розміщено вище, ми можемо відшукати анімацію, яка є найбільш підходящою для обраної моделі персонажу.

Для підключення даної анімації достатньо застосувати її до властивості моделі у конструкторі Unity 3D, але цього не завжди достатньо. У більшості випадків у сучасних іграх використовують переходи між анімаціями. Для застосування ситуативних анімацій потрібно спочатку додати відповідності значень у вікні Animator (що у редакторі Unity), перенести до вікна анімацій потрібні і налаштувати можливості і умови (за отриманими значеннями прив’язаних до аніматора змінних) переходу між анімаціями (їх маршрутизацію), написати MonoBehaviour скрипт, що описуватиме умови переходів із імпортованим компонентом Animator встановленням нових значень, прив’язаних до конструктора змінних. Цей двошаровий спосіб забезпечує спрощення візуального сприйняття вмісту діаграми. З використанням метода Animator SetInteger можна налаштовувати будь-яку кількість станів персонажа на поточний момент.

Реалізація таймера завдань (зупинки гри) програмується наступним чином. Спочатку спрацьовує таймер відліку. У гравця є 3 секунди на підготовку до гри. Коли час закінчується – змінюється логічна змінна launched і рахунок починається у основному таймінговому скрипті. У скрипті таймера основними полями є matchPoints, delays і countdown, що містять точки на прямій часу і поточний час. Із переходом через точку delays програма переходить до поточної matchPoints, а при поверненні – mpi (абревіатура до matchPointsIterator) збільшується на 1, що означає поставлення нової задачі гравцю. При перевиконанні чи недовиконанні поставленої задачі до ітерації mpi гра закінчується і гравець переходить до головного меню, інакше – продовжує гру. По закінченню задач гравцю відображається повідомлення про перемогу.



```
if (launched)
{
    try
    {
        if (hits >= mpi + 1)//HITS PROCESSING
        {
            ShowTimer();
            SetMsg("HITS:" + hits);
            timeViewed = 1;
        }
        else if (timeViewed > 0)
        {
            timeViewed -= Time.deltaTime;
        }
        else
        {
            SetMsg();
            timeViewed = 0;
        }
        countdown += Time.deltaTime;
        if (hits >= mpi + 1)//If count of kills == current time point
        {
            timerDisplay.GetComponent<UnityEngine.UI.Text>().color = Color.green;
        }
        else
        {
            timerDisplay.GetComponent<UnityEngine.UI.Text>().color = Color.white;
        }
        if (countdown < matchPoints[matchPoints.Count - 1])//if countdown in range
        {
            if (onbreak)
            {
                timerDisplay.GetComponent<UnityEngine.UI.Text>().color = Color.black;
                timerDisplay.GetComponent<UnityEngine.UI.Text>().text = "[" + delays[mpi] + "]:" + SecsToTime(Truncate(countdown, 3)).ToString();
                if (countdown >= delays[mpi])
                {
                    Debug.Log("Switching to " + mpi + "-th break.");
                    onbreak = false;
                    timerDisplay.GetComponent<UnityEngine.UI.Text>().color = Color.white;
                }
            }
            else//Iteration
            {
                timerDisplay.GetComponent<UnityEngine.UI.Text>().text = "[" + matchPoints[mpi] + "]:" + SecsToTime(Truncate(countdown, 3)).ToStri
                if (countdown >= matchPoints[mpi])
                {
                    Debug.Log("Switching to " + mpi + "-th timepoint.");
                    onbreak = true;
                    CheckHitsDone();
                    mpi++;
                }
            }
        }
        else//Processing endpoint
        {
            SetMsg("Win");
            msgDisplay.GetComponent<UnityEngine.UI.Text>().color = Color.green;
            Debug.Log("Countdown ends with final value of " + countdown + " seconds.");
            countdown = matchPoints[matchPoints.Count - 1];
            launched = false;
            timerDisplay.GetComponent<TimerBehaviour>().HideTimer();
        }
    }
    catch (Exception ex)
    {
        Debug.Log(ex.Message);
    }
    Debug.Log("---------------------------------");
}
```

Рис. 1. Скріншот програми реалізації таймера

Програмування реалізації сили тяжіння і стрибка відбувається наступним чином. Через можливість повного контролю над тим, що відбувається у грі та використання нестандартної моделі стрибка було обрано описання власної системи гравітації. Перевірка на вертикальний рух персонажа має такий вигляд :

if (IsGrounded(PLANE_THICKNESS))

Тут IsGrounded має реалізацію за допомогою Physics.Raycast, що перевіряє, чи відбувається зіштовхування тіла із будь-яким зовнішнім об'єктом (приймає положення поточного об'єкта, напрям руху, додаткову товщину поверхні). Аналіз гравітації проходить разом із аналізом стрибку (у одному логічному операторі if), бо вони не існуватимуть одночасно [4].

Також в грі здійснюється реалізація горизонтального руху. Для відповідності стандартній моделі обраного ігрового жанру рух персонажа буде змінюваним відносно поточного оберту камери навколо персонажа (а точніше дочірнього елементу юніта, що включає камеру). Керування персонажем на ПК відбувається натисканням клавіш W, A, S, D та їх комбінацій. Усього гра здатна обробити 8 комбінацій клавіш W, A, S, D для задання відповідного напряму. Наведений код здійснює контроль над рухом і його спрямуванням відносно поточного напряму зору камери.

Музичний супровід задається двома об’єктами: плеєром і аудіо кліпом. Музичний плеєр завантажується з об’єкта, до якого його приєднано, а кліп – з папки ресурсів, що має бути обов’язково присутня. Опційно можна налаштувати рівень звуку через параметр AudioSource.volume [5]. Почати програвання звуку можна командою AudioSource – PlayOneShot.

Встановлення і запуск гри відбувається запуском пакету exe. Меню гри має доволі простий інтерфейс. Start починає гру, Exit – здійснює вихід.

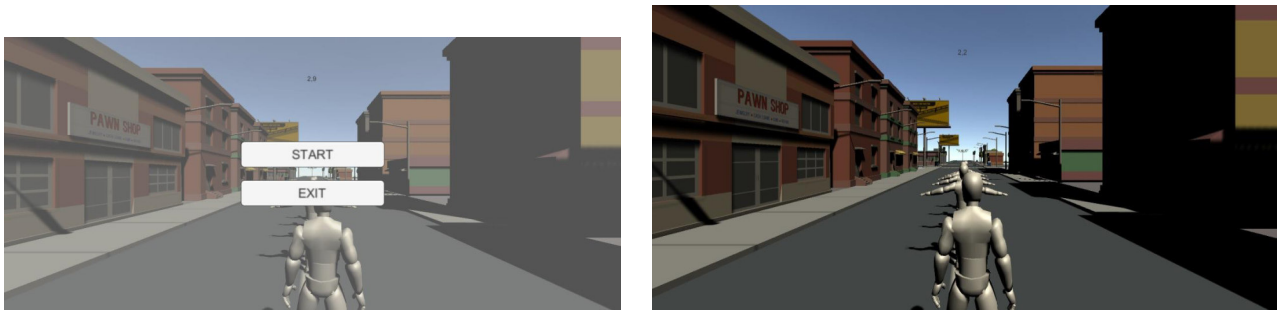


Рис. 2. Скріншоти реалізації 3-D гри

Із запуском гри починається початковий відлік для підготовки. З досягненням нуля дозволяється рух персонажу. Удари здійснюються за допомогою ЛКМ. Коли таймер чорного кольору – гравець не повинен здійснювати удари, це потягне за собою повернення до головного меню. Як тільки таймер білого кольору – потрібно здійснити один удар по цілі (не більше – ні менше). Коли таймер зелений – поточне завдання виконано, більше ударів не потрібно. Здійснення усіх ударів вчасно означає перемогу.

При написанні гри на основі 3D моделі було здійснено аналіз подібних проєктів та сервісів на ринку або їх окремих модулів. На основі цього було сформовано вимоги до функціональних можливостей системи. Було проведено проєктування поведінкових класів за поточними вимогами до можливостей гри. Продукт є кросплатформенним додатком, що розроблений за технологією Unity 3D мовою C#.

1. ТОП-10 лучших игр для PS VR–MOYO. URL: https://www.moyo.ua/news/10-i-1-luchshaya-igra-dlya-ps-vr.html (дата звернення: 14.05.2021). **2.** Box Collider–Unity Manual. URL: https://docs.unity3d.com/ru/2019.4/Manual/class-BoxCollider.html (дата звернення: 14.05.2021). **3.** Unity C# уроки / #5Instantiate (Создание объектов). URL: https://www.youtube.com/watch?v=2CeedUmODOI&ab_channel=%D0%93%D0%BE%D1%88%D0%B0%D0%94%D1%83%D0%B4%D0%B0%D1%80%D1%8C (дата звернення: 14.05.2021). **4.** Физика в Unity-9. Raycast. URL: https://www.youtube.com/watch?v=jNvmp4SZj9c (дата звернення: 14.05.2021) **5**. AudioSource.PlayOneShot увеличение громкости и отсечение. URL: https://coderoad.ru/53491658/AudioSource-PlayOneShot (дата звернення: 14.05.2021).