КОМПЛЕКСНОЕ ПРИМЕНЕНИЕ ТЕХНОЛОГИЙ РАЗРАБОТКИ ИГРОВЫХ ПРИЛОЖЕНИЙ НА ПЛАТФОРМЕ UNITY: ОТ ПРОТОТИПИРОВАНИЯ ДО ОПТИМИЗАЦИИ

САВКИНА Анастасия Васильевна
кандидат технических наук, доцент
КОТЬКИН Егор Александрович
студент
Мордовский государственный университет им. Н. П. Огарева
г. Саранск, Республика Мордовия, Россия

В статье рассматривается процесс комплексного применения информационных технологий Unity для создания мультимедийного приложения. Реализованы отслеживание и сохранение времени прохождения уровней, партиклы, шаблоны паттернов, интерактивное меню, лазеры, саунд менеджер, куб хроноса и босс. Многосторонний подход демонстрирует возможности технологий для создания функционального и увлекательного игрового приложения.
Ключевые слова: Unity, партиклы, паттерны, меню, лазер, куб хроноса, босс, таймер.

Для реализации полнофункционального мультимедийного приложения большое внимание уделено информационным технологиям, которые позволяют добавить реалистичности, красочности, умению быстро ориентироваться в сложных ситуациях, принимать ответственные решения. Среди них оригинальный таймер, шаблоны для спавна, меню, созданы различные препятствия при переходе из одной башни в другую, пауза, куб хроноса и босс [2].

Таймер. В случае реализации таймера учитывается тот факт, что пользователь, который понимает, что его время при переходе из одной башни в другую строго ограничено, начинает совершать ошибки, поэтому для усложнения была добавлена опция для плавного изменения цвета ползунка таймера от timerColor к timerColorEnd, при этом скорость изменения зависит от квадрата elapsedTime и квадрата (timeConst — 3), что обеспечивает нелинейную (ускоряющуюся) интерполяцию. Для этого была использована функция Lerp, у которой первый и второй параметры, это состояние между которыми идет изменение, а третий параметр должен находиться в диапазоне от 0 до 1, таким образом таймер по параболе медленно переходит от желтоватого к красному, создавая при этом дополнительное напряжение для игрока.

```
timer.color = Color.Lerp(timerColor, timerColorEnd, (elapsedTime * elapsedTime) /
((timeConst - 3) * (timeConst - 3)));
```

Для прохода всего игрового пути реализовано 8 башен, причем чем выше по уровню пользователь поднимается, время прохождения сокращается, а именно, если на первой башне на прохождение этажа отводится 30 секунд, то на последней – всего 12 секунд, то есть каждый раз время прохождения уменьшается на 2,5 секунды.

```
Timer timer = GameObject.FindGameObjectWithTag("MainCamera").GetComponent<Timer>();
timer.timeConst = Mathf.Round(30 - 2.5f * (curScInd - 1));
if (curScInd == 8)
    timer.timeConst = 12;
```

Паттерны. Для реализации шаблонов для спавна было решено хранить данные в виде json файла. В каждом из этих файлов записывались координаты позиций в последовательности: 1 – позиция куба игрока, со 2 по 6-ую – позиции остальных кубов. При этом в каждой башне были доступны разные паттерны, например, в первой доступно всего 3 паттерна, в последней – все 15.

```
sl.positionsSaved.Add(new Vector3(-192, 0, 0));
```

```
sl.positionsSaved.Add(new Vector3(0, 0, 0));
…
jsonSaveList = JsonUtility.ToJson(sl);
File.WriteAllText(Application.persistentDataPath + "/savedListData" +
PNum.ToString() + ".json", jsonSaveList);
```

где sl – это экземпляр класса, который можно сериализовать через JsonUtility, и у него есть поле positionsSaved (например, List<Vector3>). Пример содержания такого файла с реализацией функции, отвечающей за выбор паттерна на этаже.

```
public List<Vector3> LoadSpawnerList(List<Vector3> SpawnerListV, int LPattern, int
RPattern)
{
    Spawner spawner = GameObject.FindGameObjectWithTag("MainCamera").GetCompo-
nent<Spawner>();
…
    {
        PNum = Random.Range(LPattern, RPattern + 1);
    }
    spawner.PNum = PNum;
    string jsonSaveList = File.ReadAllText(Application.persistentDataPath +
"/savedListData" + PNum.ToString() + ".json");
    …
    {
        SpawnerListV.Add(sl.positionsSaved[i]);
    }
```

Интерактивное меню. Меню реализовано с помощью скрипта, отвечающего за отображение лучшего времени и последовательного открытия уровней [1, 3].

```
        GlobalTImers[i].text = 0.ToString() + ":" +
Mathf.Round(savedData.TVsaved[i]).ToString("00");
        if (savedData.TVsaved[i] >= 60)
        {
            GlobalTImers[i].text = (Mathf.Round(savedData.TVsaved[i] /
60)).ToString("00") + ":" + (savedData.TVsaved[i] % 60).ToString("00");
        }
        else
…
for (int i = 0; i < Unlocker; i++)
{
    UnlockedTowers[i].interactable = true;
}
```

Динамические лазеры. Для усложнения прохождения по этажам башен в игру были добавлены препятствия в виде динамических лазеров. Кроме функции Lerp, были добавлены синусоидальные функции и линейная интерполяция.

```
private Quaternion RotationCurveType(float current, int type)
{
    switch (type)
    {
        case 0:
            return Quaternion.Lerp(Quaternion.Euler(RotationStart), Quaternion.Eu-
ler(RotationGoal), Mathf.Pow(Mathf.Cos(Mathf.PI * current / 2), 2));
        case 1:
            return Quaternion.Lerp(Quaternion.Euler(RotationStart), Quaternion.Eu-
ler(RotationGoal), Mathf.Pow(Mathf.Cos(Mathf.PI * current / 4), 2));
        default:
            return transform.rotation;
    }
}
```

Здесь интерполяция выполняется от RotationStart к RotationGoal с коэффициентом $\cos^2($ * current / 2), что дает плавное замедление в начале и конце.

Саунд менеджер. Одной из задач было создание саунд менеджера или системы управления звуком, который обеспечил реализацию звуков для разрушения кубов и для касания лазеров. Шаблон метода для воспроизведения звука описан четырьмя параметрами: звуковой файл – audioClip, место, где создать источник звука – spawnTransform, громкость – volume и тип звука – kindOfSound.

```
public void PlaySFX(AudioClip audioClip, Transform spawnTransform, float volume, int
kindOfSound)
```

Такой подход позволил разделить внутри функции проигрывания звуков параметры определенной звуковой дорожки. Внутри нее добавлена конструкция switch. По переменной kindOfSound определяются начальные данные звука.

Пауза и куб хроноса. Пауза – классический элемент одиночных игр, который в данном приложении можно использовать только один раз. Куб хроноса – элемент игры, который появляется на этажах с вероятностью 20% и заменяет один из кубов, который находится в последовательности 3 или 4, обладает синим цветом кристалла и свечения, дает 2-4 секунды при разрушении, при этом появляется текст с количеством бонусных секунд [4].

Босс уровни. Для того чтобы уровни были более насыщенными и имели логический конец, был добавлен объект – босс. Это объект, который появляется в количестве одного на весь уровень игры. В качестве механики его прохождения было смешано две механики в виде двух разных фаз босса.

Первая заключается в том, что игроку нужно на время сделать определенное количество нажатий по объекту босса, на котором отображается требуемое число. Для реализации этой механики используется стандартный объект кнопки из движка с модификациями. За изображение босса отвечает компонент Sprite, который в дальнейшем изменяется в зависимости от текущих очков босса. Нажатия по объекту босса считываются с помощью метода OnClick() стандартного компонента Button, за каждое нажатие количество очков снижается на единицу и по достижении 0 начинается вторая

фаза. Этот метод отвечает за проверку условий на наличие у босса очков здоровья (BossPoints > 0) и игра не закончена (!isFailed), то происходит уменьшение здоровья (!isFailed) на 1, после чего текст обновляется. Смена спрайта проходит по 4 стадиям повреждения (80%, 60%, 40%, 20% здоровья).

Вторая фаза отличается тем, что таймер останавливается, появляются проверки на реакцию. Для этого используется такая механика, как Quick time event. Проверка была реализована следующим образом – в случайной из пяти заготовленных точек появляется кружок красного цвета, который со временем меняет свой цвет на оранжевый, затем на зеленый, потом в обратную сторону (снова оранжевый и красный). Особенность заключается в том, что игрок должен нажать на кружок в тот момент, когда он станет зеленым иначе засчитывается провал. Провал означает проигрыш на уровне. Сложность заключается в реализации изменения цвета с помощью Lerp, так как в некоторые моменты цвет может мелькать, возвращаясь в начальный цвет. Время на проверку задавалось случайно в диапазоне от 4.5 до 5.75 секунд, а время, когда появляется зеленый цвет в диапазоне от 2 до 3.25 секунд. Случайные диапазоны добавлены для усложнения процесса проверки. Чтобы пройти босса нужно на второй фазе пройти от 4 до 6 проверок на реакцию.

Параметр, отвечающий за добавление уровней мастерства влияет только на сложность проверки реакции. Работает по тому же принципу, что и процент побед, только в него засчитываются удачные прохождения боссов и количество попыток пройти босса, начальное значение 1/10 = 0.1. Это число округляется до десятых и умножается на 10, получается ранг мастерства. Всего рангов 10.

Заключение и перспективы.

В результате проделанной работы было создано мультимедийное игровое приложение, обладающее многообразным функционалом, интересным содержанием, увлекательными событиями.

Направлениями для дальнейшей разработки определены:

— порталы, меняющие местами кубы с задержкой;

— добавление более сложных функций цветных лазеров;

— реализация лидер бордов;

— введение дополнительных паттернов;

— обособленные переходы между этажами.

СПИСОК ЛИТЕРАТУРЫ

1. Машнин, Т. М. Введение в облачные и распределенные информационные системы / Т. М. Машнин. — Екатеринбург : Ridero, 2020. — 138 с.

2. Савкина, А. В., Котькин, Е. А. Разработка 2D-игры под мобильные устройства: от идеи до прототипа / А. В. Савкина, Е. А. Котькин // Общество. — 2025. — № 3(38). — С. 12–14.

3. Тоттен, К. Левел-дизайн. Архитектурный подход и пространственное проектирование / К. Тоттен. — 2-е изд. — Москва : Эксмо, 2025. — 592 с.

4. Харрисон, Ф. Изучаем C# через разработку игр на Unity / Ф. Харрисон. — 5-е изд. — Санкт-Петербург : Питер, 2022. — 400 с.

COMPREHENSIVE APPLICATION OF GAME APPLICATION DEVELOPMENT TECHNOLOGIES ON THE UNITY PLATFORM: FROM PROTOTYPING TO OPTIMIZATION

SAVKINA Anastasiia Vasilyevna
Candidate of Technical Sciences, Associate Professor
KOTKIN Egor Alexandrovich
Student
Ogarev Mordovia State University
Saransk, Russia

The article examines the comprehensive application of Unity information technologies to create a multimedia application. It implements level time tracking and saving, particles, pattern templates, an interactive menu, lasers, a sound manager, a chronos cube, and a boss. This approach demonstrates how these technologies support a functional and engaging game application.
Keywords: Unity, particles, patterns, menu, laser, chronos cube, boss, timer.