

Разработка и внедрение системы агрегации новостей сообществ ВКонтакте посредством VK API в Unity

Малик Александр Игоревич

*Приамурский государственный университет имени Шолом-Алейхема
студент*

Аннотация

В статье рассматривается процесс разработки системы агрегации новостей сообществ социальной сети ВКонтакте для интеграции в приложение на платформе Unity. Целью исследования является создание интерактивного приложения, которое позволяет пользователям просматривать актуальные новости из различных сообществ ВКонтакте в удобной форме. В статье описаны используемые технологии, методы интеграции с VK API в Unity и подходы к разработке пользовательского интерфейса. Проведен анализ существующих аналогов и представлены результаты внедрения, включая техническую реализацию системы и ее возможности.

Keywords: система агрегации новостей, ВКонтакте, VK API, Unity, социальные сети.

Development and implementation of the VKontakte community news aggregation system via the VK API in Unity

Malik Aleksandr Igorevich

*Sholom-Aleichem Priamursky State University
Student*

Abstract

The article discusses the process of developing a community news aggregation system on the VKontakte social network for integration into an application on the Unity platform. The purpose of the research is to create an interactive application that allows users to view current news from various VKontakte communities in a convenient form. The article describes the technologies used, methods of integration with the VK API in Unity, and approaches to user interface development. The analysis of existing analogues is carried out and the results of implementation are presented, including the technical implementation of the system and its capabilities.

Keywords: news aggregation system, VKontakte, VK API, Unity, social networks.

1 Введение

1.1 Актуальность

С ростом популярности социальных сетей и увеличением объема информации, распространяемой в них, возникает необходимость в инструментах, обеспечивающих эффективный доступ к актуальным данным.

ВКонтакте, одна из крупнейших социальных сетей в России, предоставляет API для взаимодействия с данными пользователей и сообществ. Unity, как универсальная платформа для разработки приложений, предоставляет возможности для создания интерактивных интерфейсов и интеграции с внешними API. Разработка системы агрегации новостей для ВКонтакте в среде Unity открывает новые перспективы для создания удобных приложений, которые сочетают в себе функциональность и визуальную привлекательность.

1.2 Обзор исследований

В своей бакалаврской работе Ж. Мусашен из Университета Твенте разработал систему агрегирования и валидации новостей, исследуя существующие работы и литературу по этому вопросу. Основное внимание было уделено проектированию пользовательского интерфейса и методам проверки достоверности новостей. После разработки концепции и реализации системы, проведена оценка через создание трех сценариев, опрос и тестирование удобства использования. Результаты показали, что система в целом положительно воспринимается, хотя обратная связь указывает на возможность улучшения пользовательского опыта через более детализированный подход [1]. В статье Д. А. Кирьянова, опубликованной в журнале "Программные системы и вычислительные методы", рассматриваются методы построения систем агрегации контента, включая их архитектуру, масштабируемость и производительность. Исследование сосредоточено на разработке высокоуровневой архитектуры для распределенных высоконагруженных систем, автоматизирующих сбор информации из различных источников. В статье подробно анализируются принципы работы агрегаторов контента, такие как веб-кроулинг, обнаружение нечетких дубликатов, суммаризация данных и политика обновления. Автор предлагает рекомендации по выбору архитектурных стилей и программного обеспечения, таких как системы управления распределенными базами данных и брокеры сообщений. Основной вывод исследования — необходимость подхода к построению систем агрегации контента с учетом принципов распределенных систем, разделяя их на три логические части: агрегация, обработка данных и представление информации [2].

1.3 Цель исследования

Целью данного исследования является создание модуля агрегации и отображения новостей из социальных сетей (на примере ВКонтакте) в игровом приложении, разработанном на платформе Unity. Основной задачей является интеграция функционала VK API для получения, обработки и визуализации новостного контента, а также обеспечение интерактивного взаимодействия пользователей с контентом внутри игрового приложения.

2 Материалы и методы

В рамках исследования использовался комплексный подход, включающий изучение теоретических основ, анализ существующих решений и разработку программного обеспечения.

Инструменты разработки

- Unity: выбрана в качестве основной платформы для разработки приложения, что позволяет объединить игровую механику и интерфейс новостного агрегатора.
- VK API: использовался для получения данных о новостях из сообществ ВКонтакте. Были изучены методы API, включая авторизацию, запросы к методам получения постов и обработку ответов.
- C#: основной язык программирования для реализации взаимодействия между Unity и VK API.
- JSON: использовался для обработки данных, возвращаемых VK API.

Методы разработки

- Интеграция с VK API: разработан скрипт для авторизации пользователя через VK API и получения списка новостей с использованием токена доступа.
- Обработка данных: реализованы функции для фильтрации и форматирования новостных данных, а также отображения их в Scroll View Unity.
- Интерфейс пользователя: использованы встроенные возможности Unity UI, включая компоненты Canvas, Scroll View и TextMeshPro, для создания интуитивно понятного интерфейса.

3 Результаты и обсуждения

3.1 Создание проекта

Первый этап работы начинается с создания нового проекта в Unity. Для этого требуется запустить Unity Hub нажать на New Project (Рис. 1). Следом за этим, выбрать шаблон, в данном случае - 3D. Выбор шаблона зависит от нужд. По нажатию Create Project – проект будет создан в указанной папке с указанным названием (Рис. 2).

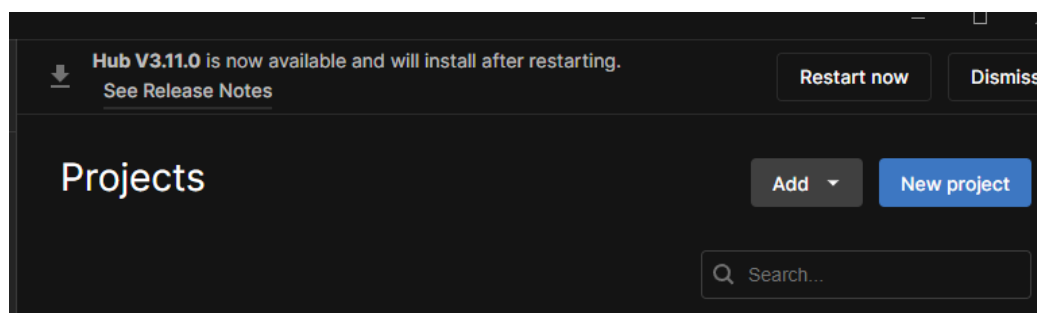


Рисунок 1. Окно создания нового проекта в UnityHub

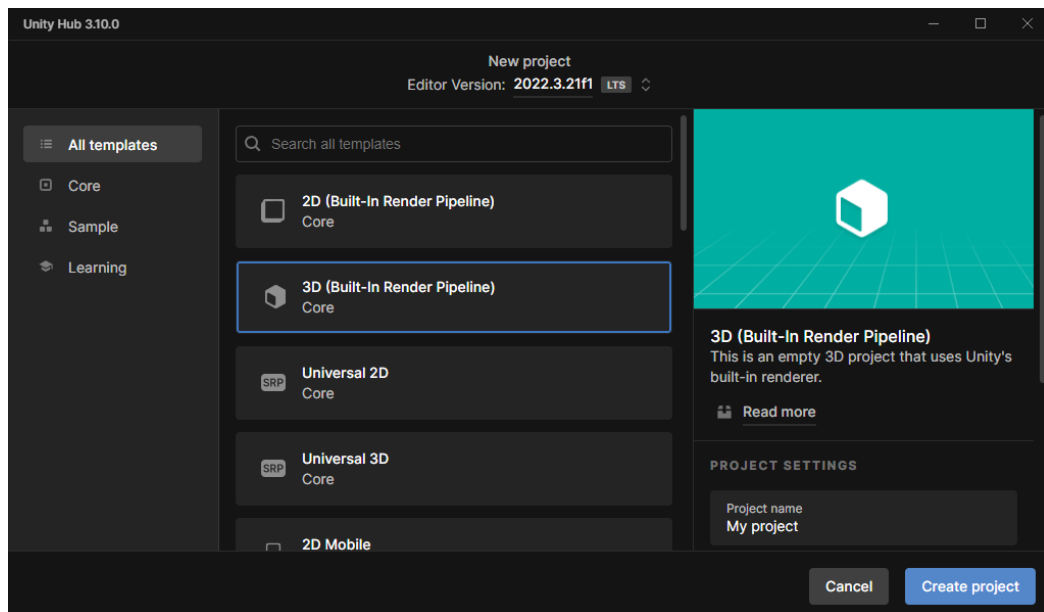


Рисунок 2. Выбор стартового шаблона проекта в UnityHub

После создания проекта можно наблюдать сам проект, в котором пока что нет ничего. (Рис. 3).

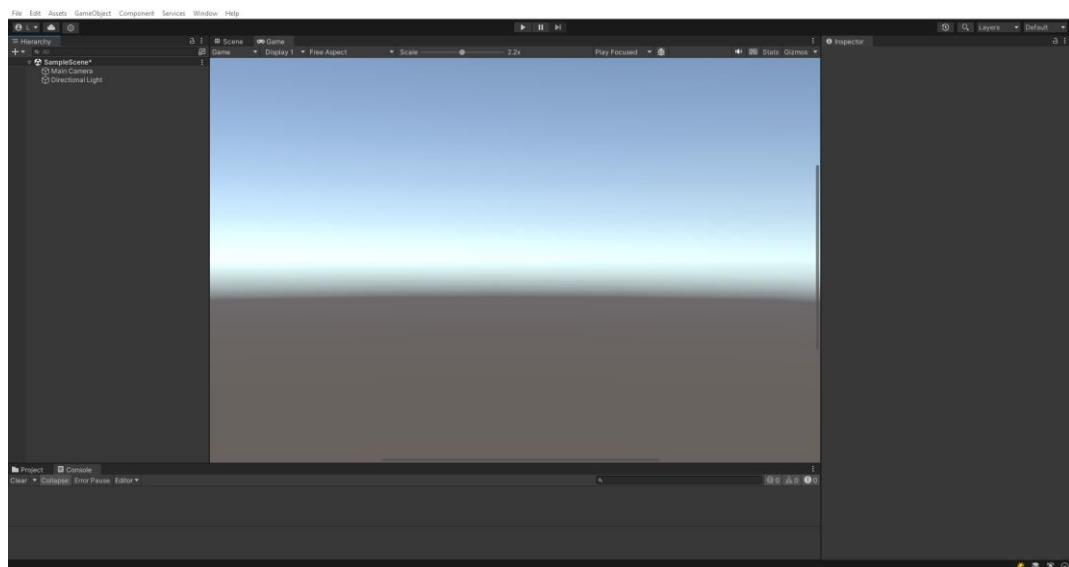


Рисунок 3. Окно пустого проекта в Unity

3.2 Настройка VK API

Далее необходимо осуществить подготовку к взаимодействию с VK API. Для этого нужно иметь действующий токен, который предоставляет доступ к необходимым методам API.

Получение токена **VK API**:

1. Переход на страницу создания приложений VK.
2. Нажатие на "Создать приложение" или "Создать сообщество", в зависимости от потребностей.

3. Ввод названия приложения и выбор типа (standalone для личного использования, сайт, сторонние или Android/iOS).

4. После создания приложения скопировать его ID из адресной строки браузера.

Получение **access_token**:

1. Необходимо вернуться на страницу созданного приложения в Мои приложения.

2. Перейти во вкладку "Настройки" и найти раздел "Настройки". Там можно увидеть "Сервисный ключ доступа". Далее необходимо скопировать его.

3. Теперь есть **access_token**, который можно использовать в своем приложении для взаимодействия с VK API.

3.3 Создание агрегатора

Далее, необходимо создать основной скрипт агрегатора, для этого, подключим необходимые встроенные библиотеки и создадим класс агрегатора (Рис. 4).

```
using UnityEngine;
using UnityEngine.Networking;
using TMPro;
using UnityEngine.EventSystems;
using System.Collections;
using System.Text.RegularExpressions;

public class VKNewsDisplay : MonoBehaviour, IPointerClickHandler
{
}
```

Рисунок 4. Подключение библиотек и объявление класса агрегатора

3.3 Создание окна с постами

Создадим объект интерфейса с вложенными объектами Text и Scroll View. Для этого необходимо добавить полотно (Canvas), сделать это можно нажатием ПКМ по иерархии проекта и в появившемся окне выбрать UI – Canvas. Создадим два объекта Text и Scroll View пользовательского интерфейса и поместим их в родительский объект иерархии Canvas. В свойстве text объекта Text напомним «Новости» и расположим в верхней части интерфейса. Scroll View расположим ниже объекта Text (Рис. 5).

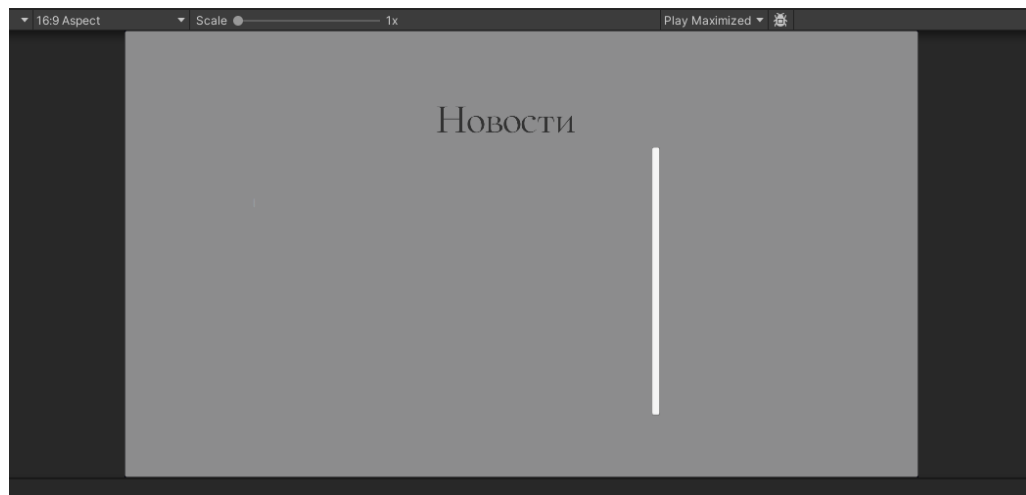


Рисунок 5. Вид панели новостей

Зададим параметры Scroll View в инспекторе объекта (Рис. 6).

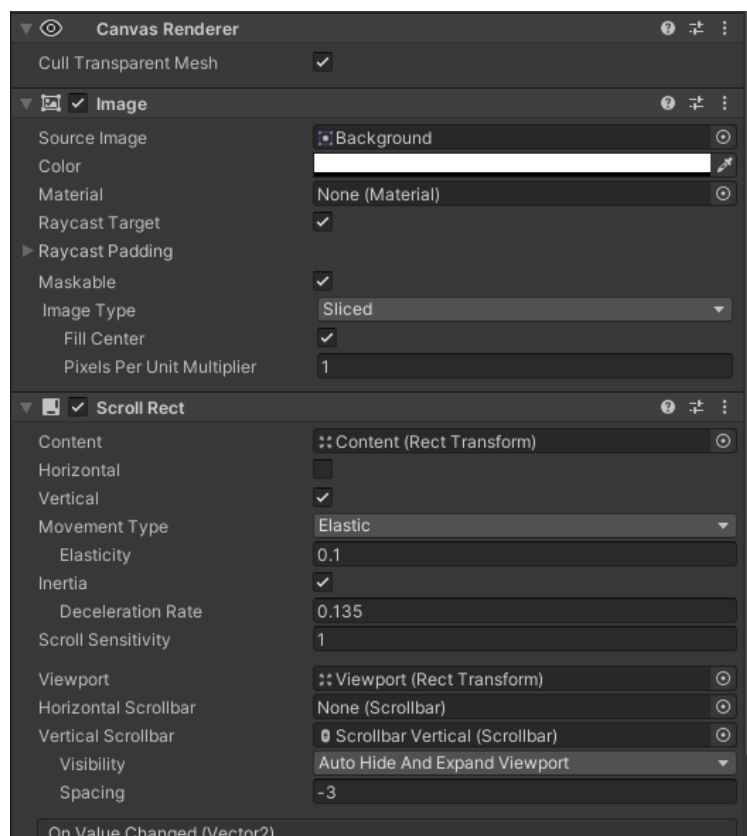


Рисунок 6. Свойства компонентов целевого объекта Scroll View

3.4 Создание префаба поста

Для вывода большого кол-ва постов воспользуемся методом создания копий объекта одного поста. Для этого создадим внутри Scroll View – Viewport объект с именем Content, этот объект и будет дублироваться и содержать в себе информацию о разных постах (Рис. 7).

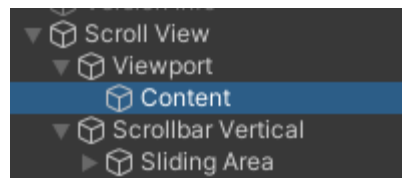


Рисунок 7. Структура окна агрегатора

Для того, чтобы задать получившемуся пустому объекту необходимый вид и функционал – добавим 3 компонента: Content Size Fitter, Vertical Layout Group, Scroll Rect. Выставим необходимые настройки и ссылки на нужные объекты в компонентах (Рис. 8).

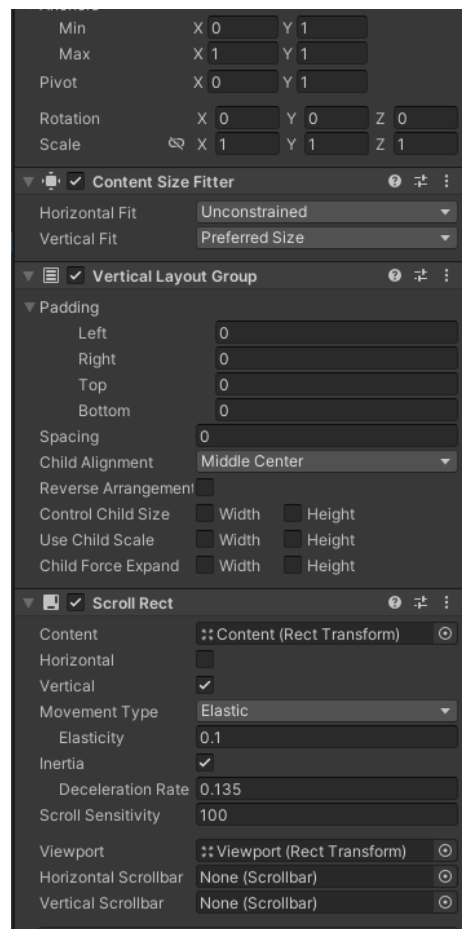


Рисунок 8. Компоненты шаблонного объекта

3.5 Создание структурированных классов, описывающих объекты для парсинга новостей

Добавим классы внутрь класса агрегатора VKNewsDisplay, для более четкого разделения данных и работы с ними: VKPost, VKItems, VKResponse, VKGroups, VKGroups, VKGroupResponse

```

[System.Serializable]
public class VKPost
{
    public int id;
    public int from_id;
    public int date;
    public string text;
}

[System.Serializable]
public class VKItems
{
    public VKPost[] items;
}

[System.Serializable]
public class VKResponse
{
    public VKItems response;
}

[System.Serializable]
public class VKGroup
{
    public string id;
    public string name;
    public string screen_name;
}

[System.Serializable]
public class VKGroups
{
    public VKGroup[] groups;
}

[System.Serializable]
public class VKGroupResponse
{
    public VKGroups response;
}

```

Рисунок 9. Сериализованная структура объектов

3.6 Добавление собственных параметров объекту

Для взаимодействия с группой VK необходимо выполнить обращение используя токен доступа, идентификатор группы и версию API VK. Для этого, объявим в главном классе заголовок с переменными, используя декоратор Header визуального разделения параметров в инспекторе объектов. К списку переменных добавим данные о группе и данные о внешнем виде агрегатора (Рис. 10).

```

[Header("Settings")]
public string accessToken = "9519c91f9519c91f9519c91f1a96099fdb995199519c91ff7e35f112aa0ef734c45f146"; // Вставьте свой токен
public string groupId = "63311869"; // Вставьте ID вашей группы
public string apiVersion = "5.199"; // Версия API VK
public GameObject postPrefab; // Префаб для постов
public Transform content; // Содержимое ScrollView
private string groupName;

```

Рисунок 10. Основные переменные, принимающие значения

3.7 Обращение к VK API и обработка полученных данных

Следующий шаг включает в себя создания Coroutine для асинхронного парсинга данных. Для этого объявим две основные функции GetGroupName() и GetGroupPosts(). В первом случае выполняем запрос: https://api.vk.com/method/groups.getById?group_id={groupId}&access_token={accessToken}&v={apiVersion} подставляя соответствующие данные из введенных переменных и извлекаем из полученного JSON-объекта – имя

группы, во втором: получаем содержимое последних 10-ти постов запросом https://api.vk.com/method/wall.get?owner_id=-{groupId}&count=10&access_token={accessToken}&v={apiVersion}, в данном случае любой текст, который содержат посты (Рис. 11-12).

```
private IEnumerator GetGroupName()
{
    string url = $"https://api.vk.com/method/groups.getById?group_id={groupId}&access_token={accessToken}&v={apiVersion}";
    using (UnityWebRequest request = UnityWebRequest.Get(url))
    {
        yield return request.SendWebRequest();

        if (request.result != UnityWebRequest.Result.Success)
        {
            Debug.LogError($"Ошибка запроса имени группы: {request.error}");
        }
        else
        {
            VKGroupResponse response = JsonUtility.FromJson<VKGroupResponse>(request.downloadHandler.text);

            if (response.response != null && response.response.groups.Length > 0)
            {
                groupName = response.response.groups[0].name;
            }
        }
    }
}

private IEnumerator GetGroupPosts()
{
    string url = $"https://api.vk.com/method/wall.get?owner_id=-{groupId}&count=10&access_token={accessToken}&v={apiVersion}";
    using (UnityWebRequest request = UnityWebRequest.Get(url))
    {
        yield return request.SendWebRequest();

        if (request.result != UnityWebRequest.Result.Success)
        {
            Debug.LogError($"Ошибка запроса: {request.error}");
        }
        else
        {
            VKResponse response = JsonUtility.FromJson<VKResponse>(request.downloadHandler.text);

            if (response.response != null && response.response.items != null)
            {
                foreach (VKPost post in response.response.items)
                {
                    if (post.text != "") CreatePostUI(post);
                }
            }
        }
    }
}
```

Рисунок 11. Функции парсинга данных через VK API

```
private void CreatePostUI(VKPost post)
{
    GameObject postObject = Instantiate(postPrefab, content);

    // Получаем ссылки на компоненты TextMeshProUGUI
    TextMeshProUGUI textComponent = postObject.transform.Find("Text").GetComponent<TextMeshProUGUI>();
    TextMeshProUGUI dateComponent = postObject.transform.Find("Date").GetComponent<TextMeshProUGUI>();
    TextMeshProUGUI authorComponent = postObject.transform.Find("Author").GetComponent<TextMeshProUGUI>();

    // Устанавливаем значения
    textComponent.text = RemoveUnsupportedCharacters(FormatTextWithLinks(post.text));
    dateComponent.text = UnixTimeToDateTime(post.date).ToString("yyyy-MM-dd HH:mm");
    authorComponent.text = $"{groupName}";

    // Добавляем компонент для обработки кликов
    LinkHandler linkHandler = postObject.AddComponent<LinkHandler>();
    linkHandler.textMeshPro = textComponent;
}
```

Рисунок 12. Функция обработки данных и формирования поста

Основная работа закончена, остается сохранить скрипт и навесить его на объект агрегатора, после чего указать необходимые параметры для начала работы агрегатора (Рис. 13).

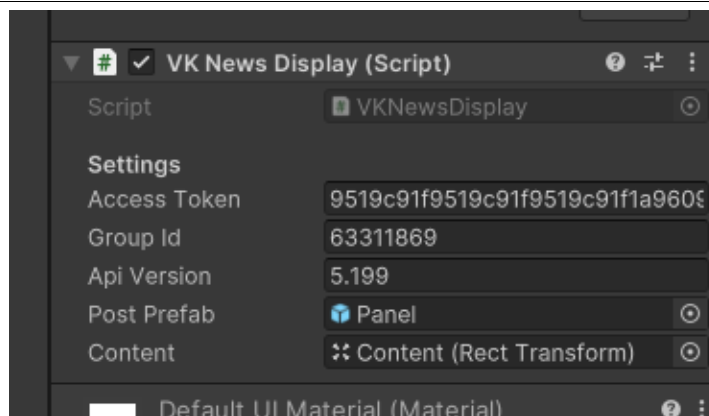


Рисунок 13. Готовые параметры скрипта парсинга новостей

3.9 Результат парсинга

Результатом проделанной работы является полноценный блок с актуальной информацией из указанной группы VK (Рис. 14).

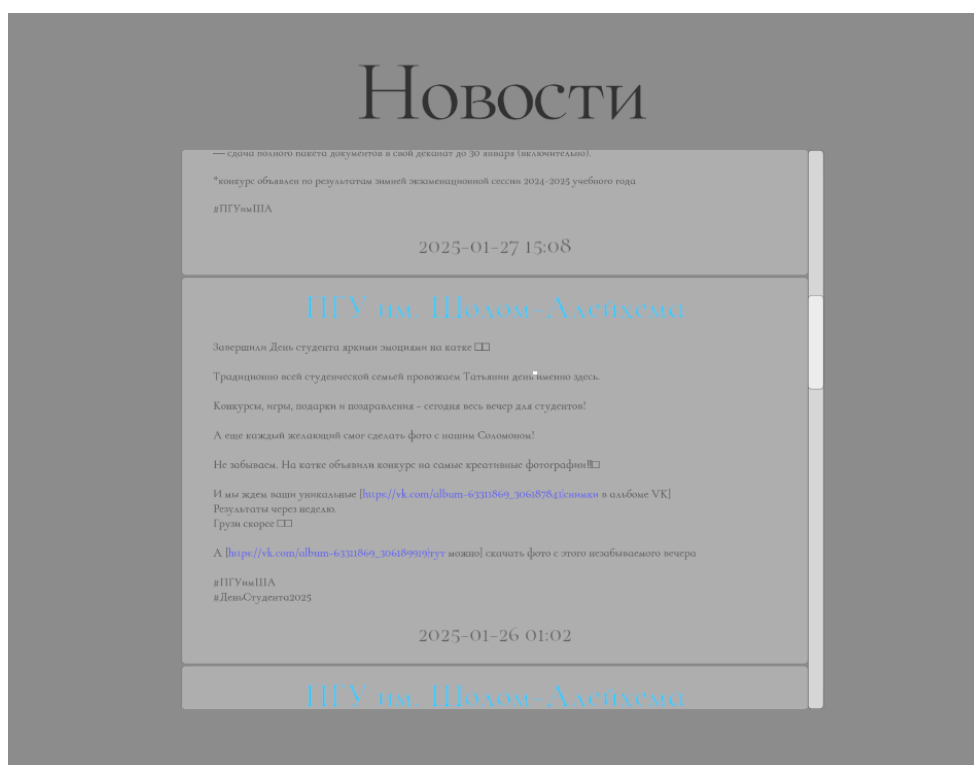


Рисунок 14. Готовый рабочий блок с новостями группы

3.10 Заключение

В результате проведенного исследования была достигнута цель — разработан модуль для агрегации и отображения новостей из социальной сети ВКонтакте в игровом приложении на платформе Unity. Разработанный модуль позволяет интегрировать функционал VK API, обеспечивая авторизацию, получение, обработку и отображение данных. Успешно реализован пользовательский интерфейс на базе Unity UI, включающий интерактивные элементы, такие как Scroll View и префабы постов.

Использование структурированных классов и методов позволило создать удобный и масштабируемый модуль, который можно адаптировать под различные проекты. Применение асинхронных запросов к VK API обеспечивает эффективную работу с данными, минимизируя нагрузку на приложение.

4 Выводы

Разработанный модуль продемонстрировал возможность успешного объединения игровых механик и новостного агрегатора, что открывает перспективы для интеграции информационных сервисов в игровые приложения. Решение отличается гибкостью, простотой внедрения и возможностью дальнейшего масштабирования. В будущем данная разработка может быть дополнена функционалом интерактивного взаимодействия пользователей с новостным контентом, а также расширена на другие социальные сети.

Библиографический список

1. Moucachen J. Developing a News Aggregation and Validation System. University of Twente. 2017. С. 1-100.
2. Кирьянов Д.А. Исследование методов построения систем агрегации контента // Программные системы и вычислительные методы. 2022. № 1. DOI: 10.7256/2454-0714.2022.1.37341.