УДК 004.032.26

Создание и интеграция простого чат-бота в Unity на C#

Ульянов Егор Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
Целью данной статьи является описание процесса создания и интеграции простого чат-бота в среде разработки «Unity» с использованием языка программирования C# и фреймворка «Syn Bot Framework». Итогом исследования стал интегрированный в игру рабочий чат-бот.
Ключевые слова: Unity 3D, чат-бот, фреймворк

Creating and integrating a simple chat-bot in Unity in C#

Ulianov Egor Andreevich
Sholom-Aleichem Priamursky State University
Student

Abstract
The purpose of this article is to describe the process of creating and integrating a simple chatbot in the Unity development environment using the C# programming language and the Syn Bot Framework. The result of the study was a working chat-bot integrated into the game.
Keywords: Unity 3D, chatbot, framework

1 Введение

1.1 Актуальность исследования

Актуальность данного исследования заключается в растущей популярности чат-ботов в различных областях, в том числе в игровой индустрии. С развитием технологий машинного обучения и искусственного интеллекта, использование чат-ботов в играх становится все более распространенным. Создание и интеграция чат-бота в игровую среду позволяет улучшить взаимодействие игроков с игрой, добавить элементы искусственного интеллекта, создать интересные игровые ситуации и повысить уровень вовлеченности игроков. Таким образом, исследование по созданию и интеграции чат-бота в «Unity» на языке C# имеет практическую значимость для разработчиков игр и индустрии развлечений.

1.2 Обзор исследований

С.С. Лунин в своей статье описал использование Unity Bolt при разработке модуля, реализующего систему управления игровыми объектами для интегрированной среды разработки и движка Unity3D [1]. С. А. Суродин в

своей статье представил сценарий углубленного изучения одного из лучших движков, существующих на данный момент, для создания красивых 2D и 3D игр [2]. В своей работе Р. Ф. Гайнуллин, В. А. Захаров, Е. А. Аксенова изучили инструмент для разработки двух- и трёхмерных игр – Unity 3D [3]. И. А. Савин, О. В. Батенькина рассмотрели процесс написания скриптовых сценариев при разработке виртуального тренажера [4]. В своей работе Э. Р. Гараева, И. И. Бикмуллина, И. А. Барков описали возможности Unity 3D на предмет создания 3D-моделей [5]. А. Ю. Субботина, Н. И. Хохлов в результате их работы была предложена трехмерная версия игры Конвея «Жизнь» [6].

1.3 Цель исследования

Цель исследования – применяя фреймворк «Syn Bot Framework» и игровой движок «Unity 3D», создать и интегрировать простой чат-бот в тестовую игру.

2. Материалы и методы

Для создания нейронной сети будем использовать программное обеспечение Syn Bot Framework [7]. Syn Bot Framework представляет собой инструмент для создания и развертывания чат-ботов, основанный на искусственном интеллекте. Этот фреймворк предоставляет разработчикам гибкую платформу для создания различных видов чат-ботов, включая текстовые, голосовые и мультимедийные боты. Syn Bot Framework предлагает множество функций, таких как распознавание и понимание естественного языка (NLU), интеграция с различными каналами коммуникации (например Slack, Skype и др.), управление диалогами, а также множество инструментов для аналитики и мониторинга производительности бота.

Этот фреймворк также обладает набором предварительно обученных моделей, которые могут быть настроены и доработаны в соответствии с потребностями и требованиями конкретного проекта. Он предоставляет легкую интеграцию с существующими системами и позволяет разработчикам создавать гибкие и мощные чат-боты для различных платформ и сфер применения. Syn Bot Framework предлагает развитую экосистему инструментов и ресурсов, что делает его популярным выбором для разработки чат-ботов с использованием искусственного интеллекта.

3 Результаты и дискуссия

Запускаем Unity и создаем новый 3D-проект, называем проект ChatBot (рис. 1).

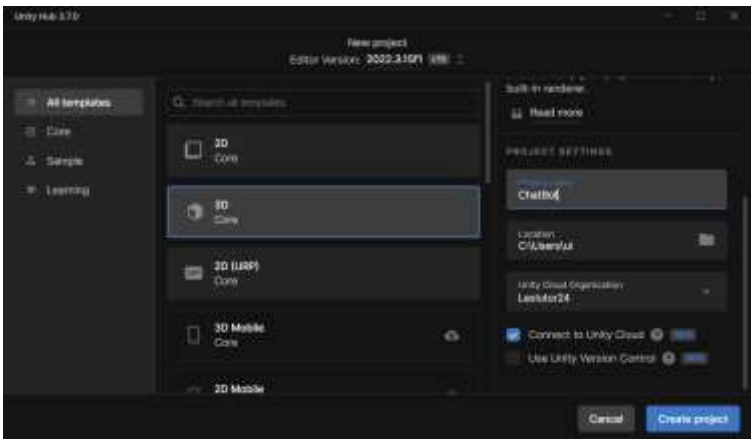

Рисунок 1. Создание проекта

Для создания интерфейса чат-бота в Unity понадобятся три важных элемента пользовательского интерфейса.

1. Панель отображения - где будут храниться все чаты / сообщения
2. Поле ввода - где пользователь будет вводить данные
3. Кнопка «Send» - кнопка, для отправки своего сообщения

Разрешение панели отображения выставляем 500 на 300 (рис.2).

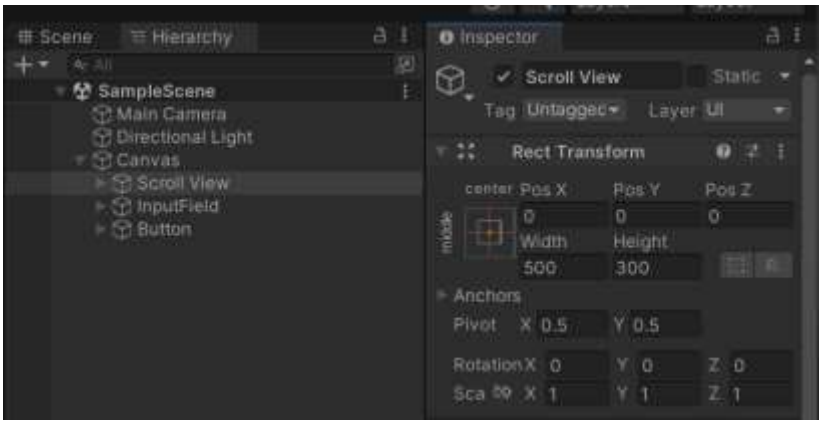

Рисунок 2. Добавление необходимых элементов

Как только сообщение пользователя или бота будет сгенерировано, нужно будет добавить это сообщение в панель отображения. Для этого создадим префаб элемента пользовательского интерфейса Text и будем использовать всякий раз, когда сообщение будет добавлено в панель отображения. Перетащите элемент в Assets, чтобы преобразовать его в prefab, и удалите элемент в окне Иерархии (рис.3-4).

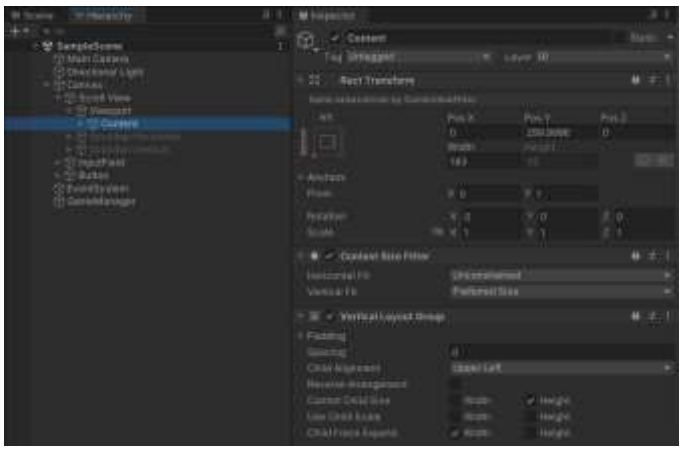

Рисунок 3. Настройки панели отображения

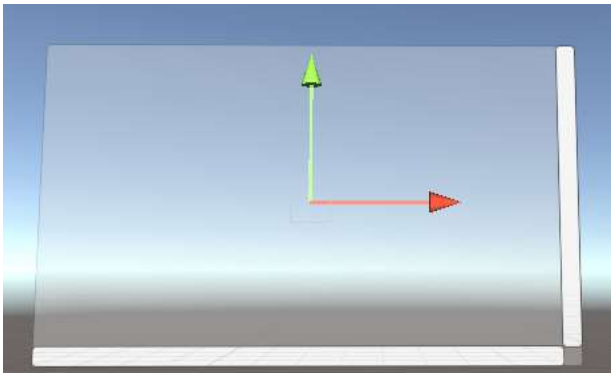

Рисунок 4. Готовый «холст» чата

Создаем поле ввода и кнопку и размещаем их рядом, как показано ниже. Эти два элемента пользовательского интерфейса будут использоваться для отправки сообщений пользователя боту (рис. 5).

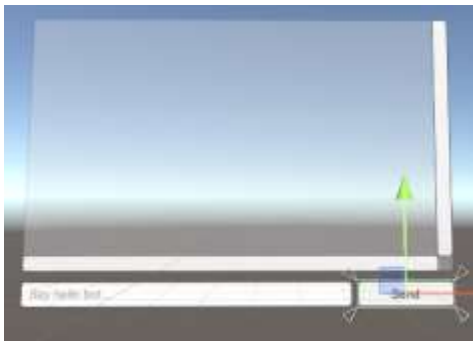

Рисунок 5. Результат проверки ячеек

Далее необходимо импортировать фреймворк «Syn Bot Framework» для работы чата (рис.6).

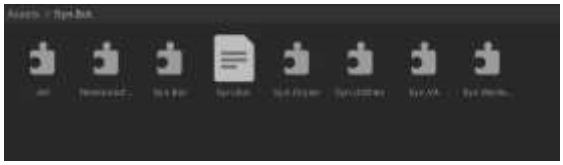

Рисунок 6. Импорт фреймворка

Переходим к написанию кода. Создаем пустой объект с именем GameManager в окне иерархии, добавляем к нему скрипт и называем так же (рис.7).

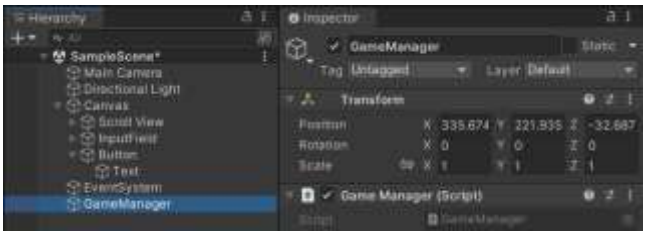

Рисунок 7. Добавление скрипта

Кликаем два раза по скрипту, чтобы перейти в редактор кода. Подключаем необходимые библиотеки, создаем переменные. Пространство имен Assets содержит классы Message, StartDialog и GameManager, а также перечисление MessageType: содержащее два элемента: User и Bot. Класс Message представляет сообщение и содержит поля Text (текст сообщения), TextObject (объект текста) и MessageType (тип сообщения). Класс StartDialog наследуется от класса Dialog и содержит метод Hello, помеченный атрибутом [Expression("Hello Bot")]. Этот метод принимает контекст и результат и отправляет ответ "Hello User!". Класс GameManager наследуется от MonoBehaviour и содержит поле MainBot типа OscovaBot, а также ссылки на объекты chatPanel, textObject и chatBox. Также содержит цвета UserColor и BotColor, а также список Messages типа Message для хранения сообщений (рис.8).

Рисунок 8. Классы Message, StartDialog и GameManager

Метод Start(), который используется для инициализации и настройки бота. Внутри метода происходит следующее:

- Создается новый экземпляр бота OscovaBot.
- Ведется журналирование логов с помощью события LogReceived, где каждая запись выводится в отладочный журнал.

- Диалоги бота обучаются на простом диалоге с помощью класса StartDialog.
- Импортируется база знаний бота из файла проекта Workspace. Этот файл содержит информацию о вопросах и ответах для обучения бота.
- Запускается процесс обучения бота с помощью метода Trainer.StartTraining().
- После того как бот генерирует ответ, он отображается в пользовательском интерфейсе через событие ResponseReceived.

В случае возникновения ошибки в любом из этих шагов, исключение будет обработано и выведено в отладочный журнал (рис.9).

```
void Start()
{
    try
    {

        MainBot = new OscovaBot();
        OscovaBot.Logger.LogReceived += (s, o) =>
        {
            Debug.Log($"OscovaBot: {o.Log}");
        };


        MainBot.Dialogs.Add(new StartDialog());


        MainBot.MainUser.ResponseReceived += (sender, evt) =>
        {
            AddMessage($"Bot: {evt.Response.Text}", MessageType.Bot);
        };
    }
    catch (Exception ex)
    {
        Debug.LogError(ex);
    }
}
```

Рисунок 9. Метод Start

Следующий код определяет метод AddMessage с двумя параметрами: messageText (текст сообщения) и messageType (тип сообщения). Внутри метода выполняется проверка, если количество сообщений в списке Messages больше или равно 25, то выполняются следующие действия: старое сообщение удаляется, а затем удаляется из списка Messages. Затем создается новое сообщение, которое представлено экземпляром класса Message. Создается новый объект текста newText путем клонирования объекта textObject и добавляется к панели чата chatPanel. Свойству TextObject объекта newMessage присваивается ссылка на созданный объект текста newText, затем устанавливается текст сообщения и цвет в зависимости от типа сообщения (пользовательское или от бота). Новое сообщение добавляется в список Messages.

Метод SendMessageToBot(), вызывается при отправке сообщения пользователем. Сначала код извлекает текст сообщения пользователя из chatBox. Затем проверяется, что сообщение не пустое. Если сообщение не пустое, то выводится отладочная информация в консоль (Debug.Log), добавляется сообщение пользователя в UI (AddMessage) и создается запрос,

который будет обработан ботом. Затем созданный запрос оценивается (Evaluate) с помощью NLU (Natural Language Understanding), и результат оценки вызывает лучший предложенный замысел, что приводит к генерации ответа от бота.

Наконец, очищается текстовое поле chatBox и устанавливается фокус для дальнейшего взаимодействия с пользователем.

В целом, этот метод передает сообщение пользователя боту для обработки и получает ответ от него (рис.10).

```
public void AddMessage(string messageText, MessageType messageType)
{
    if (Messages.Count >= 25)
    {

        Destroy(Messages[0].TextObject.gameObject);
        Messages.Remove(Messages[0]);
    }

    var newMessage = new Message { Text = messageText };

    var newText = Instantiate(textObject, chatPanel.transform);

    newMessage.TextObject = newText.GetComponent<Text>();
    newMessage.TextObject.text = messageText;
    newMessage.TextObject.color = messageType == MessageType.User ? UserColor : BotColor;

    Messages.Add(newMessage);
}

public void SendMessageToBot()
{
    var userMessage = chatBox.text;

    if (!string.IsNullOrEmpty(userMessage))
    {
        Debug.Log($"OscovaBot:[USER] {userMessage}");
        AddMessage($"User: {userMessage}", MessageType.User);


        var request = MainBot.MainUser.CreateRequest(userMessage);


        var evaluationResult = MainBot.Evaluate(request);


        evaluationResult.Invoke();

        chatBox.Select();
        chatBox.text = "";
    }
}
```

Рисунок 10. Методы AddMessage и SendMessageToBot

Метод Update(), вызывается каждый раз при обновлении кадра в игре. В данном случае, код проверяет, была ли нажата клавиша Enter (KeyCode.Return). Если клавиша Enter была нажата, то вызывается метод SendMessageToBot(), который отправляет сообщение пользователя боту для обработки.

Это позволяет пользователю отправлять сообщения боту, нажимая клавишу Enter вместо того, чтобы нажимать кнопку отправки сообщения вручную (рис.11).

```
void Update()
{
    if (Input.GetKeyDown(KeyCode.Return))
    {

        SendMessageToBot();
    }
}
```

Рисунок 11. Метод Update

Переходим в Unity и в настройках скрипта, связываем поля с соответствующими компонентами. Устанавливаем для значений объекта User color и Bot color любой цвет, и устанавливаем альфа-значение 255, чтобы цвет оставался видимым (рис.12).

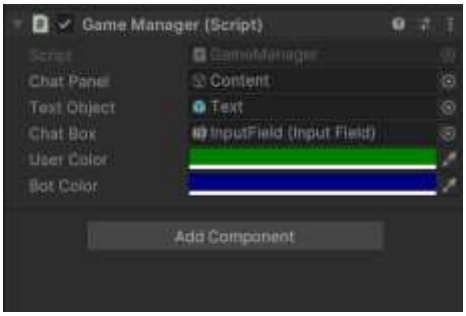

Рисунок 12. Настройки скрипта

Метод SendMessageToBot() необходимо вызывать всякий раз, когда пользователь нажимает кнопку Send, для этого в настройках кнопки в тригере нажатия кнопки добавляем скрипт GameManager (рис.13).

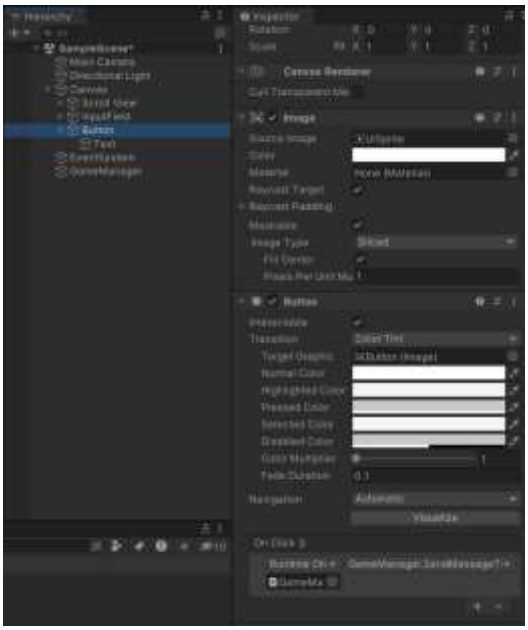

Рисунок 13. Настройка кнопки

Дописав код, тестируем программу (рис.14).

Рисунок 14. Результат программы

4 Выводы

В данной статье был представлен процесс создания и интеграции простого чат-бота в среде разработки Unity с использованием языка программирования C#. В ходе работы были рассмотрены основные принципы создания чат-ботов, такие как обработка естественного языка (NLP), а также способы интеграции бота в игровую среду. Таким образом, статья представляет руководство для разработчиков, желающих интегрировать чат-ботов в свои проекты на Unity, и может служить основой для дальнейших исследований и экспериментов в области создания интерактивных и игровых сценариев.

Библиографический список

1. Лунин С.С. Разработка подключаемого модуля реализации подсистемы управления игровыми персонажами в среде Unity3D // Информационные технологии, системный анализ и управление (ИТСАУ-2019). 2019. №2. С. 371-376.
2. Суродин С. А. Unity 3D разработка сценария проектирования в среде Unity 3D // Информатика и вычислительная техника. 2015. №3. С. 504-511.
3. Гайнуллин Р. Ф., Захаров В. А., Аксенова Е. А. Создание 2d игры на Unity 3D 5.4 // Вестник современных исследований. 2018. №4. С. 78-82.
4. Савин И. А., Батенькина О. В. Написание скриптов для трехмерного графического движка // Визуальная культура: дизайн, реклама, информационные технологии. 2018. № 12-7 (28). С. 7-15.
5. Долженко А. И., Глушенко С. А. Особенности подготовки 3d-объектов, смоделированных в Blender, для импорта в Unity 3D // Прикаспийский журнал: управление и высокие технологии. 2014. №6. С. 92-96.
6. Субботина А. Ю., Хохлов Н. И. Реализация клеточных автоматов "игра жизнь" и клеточного автомата Кохомото-ооно с применением технологии MPI // Компьютерные исследования и моделирование. 2010. №17. С. 319-322.