Привет! Чтобы заменить стандартный InputField на TextMeshPro в Unity, выполни следующие шаги:

**1. Убедись, что у тебя установлен TextMeshPro**

* Если еще не установлен, открой **Window → Package Manager**.
* В списке пакетов найди **TextMeshPro** и установи его.

**2. Замена стандартного InputField на TMP\_InputField**

**Вариант 1: Через UI-меню**

1. **Удали** старый InputField (если он уже есть на сцене).
2. Кликни правой кнопкой в **Hierarchy** → **UI** → **TextMeshPro - Input Field**.
3. Настрой его так же, как обычный InputField (шрифт, размер, плейсхолдер и т. д.).

**Вариант 2: Через код (если нужно программно)**

Если у тебя есть скрипт, который использует InputField, замени:

csharp

Copy

Download

using UnityEngine.UI; *// Стандартный InputField*

public InputField myInputField;

на:

csharp

Copy

Download

using TMPro; *// TextMeshPro*

public TMP\_InputField myInputField;

**3. Настройка TMP\_InputField**

* **Placeholder**: Используй TextMeshPro Text вместо обычного Text.
* **Font**: Выбери TMP-шрифт (например, LiberationSans SDF).
* **Цвета, отступы и другие параметры** настраиваются так же, как в стандартном InputField.

**4. Важно!**

* Если у тебя в проекте были ссылки на InputField в скриптах, их нужно вручную поменять на TMP\_InputField.
* TMP-элементы обычно работают быстрее и выглядят лучше, но требуют настройки шрифтов (Font Asset).

Если что-то не получается — уточни, в какой именно части проблемы, помогу разобраться! 😊

using UnityEngine; using UnityEngine.UI; using TMPro; using System.Collections.Generic; public class DevConsole : MonoBehaviour { public GameObject consolePanel; public TMP\_InputField inputField; public Text outputText; private bool isConsoleVisible = false; private List<string> history = new List<string>(); void Update() { if (Input.GetKeyDown(KeyCode.BackQuote)) // "~" на англ. раскладке { isConsoleVisible = !isConsoleVisible; consolePanel.SetActive(isConsoleVisible); if (isConsoleVisible) { inputField.ActivateInputField(); } } if (isConsoleVisible && Input.GetKeyDown(KeyCode.Return)) { string command = inputField.text; ExecuteCommand(command); inputField.text = ""; inputField.ActivateInputField(); } } void ExecuteCommand(string command) { history.Add(command); AppendMessage("> " + command); string[] parts = command.ToLower().Split(' '); switch (parts[0]) { case "help": AppendMessage("Available commands: help, god, killall, teleport x y z"); break; case "god": AppendMessage("God mode enabled."); // TODO: Enable god mode break; case "killall": var enemies = GameObject.FindGameObjectsWithTag("Enemy"); foreach (var enemy in enemies) { Destroy(enemy); } AppendMessage("All enemies destroyed."); break; case "teleport": if (parts.Length == 4 && float.TryParse(parts[1], out float x) && float.TryParse(parts[2], out float y) && float.TryParse(parts[3], out float z)) { Transform player = GameObject.FindGameObjectWithTag("Player").transform; player.position = new Vector3(x, y, z); AppendMessage($"Teleported to {x}, {y}, {z}"); } else { AppendMessage("Usage: teleport x y z"); } break; default: AppendMessage("Unknown command. Type 'help'."); break; } } void AppendMessage(string message) { outputText.text += message + "\n"; } } - не могу найти что переместить в поле Output Text из ScrollView

Проблема в том, что ты используешь обычный Text (UnityEngine.UI.Text) для вывода сообщений, но хочешь заменить его на **TextMeshPro (TMP)**.

**🔧 Как исправить:**

1. **Замени outputText на TMP-версию:**  
   Вместо:

csharp

Copy

Download

public Text outputText; *// Стандартный Text*

Используй:

csharp

Copy

Download

public TMPro.TextMeshProUGUI outputText; *// TextMeshPro*

1. **Настройка ScrollView в Unity Editor:**
   * Удали старый Text (если он был внутри ScrollView).
   * Добавь **TextMeshPro Text** в ScrollView:
     1. Кликни ПКМ на ScrollView/Viewport/Content → **UI → TextMeshPro - Text**.
     2. Назови его, например, OutputText\_TMP.
     3. Перетащи его в поле outputText в инспекторе скрипта DevConsole.
2. **Обнови метод AppendMessage (опционально):**  
   Если текст не прокручивается автоматически, добавь это:

csharp

Copy

Download

void AppendMessage(string message)

{

outputText.text += message + "\n";

*// Автопрокрутка вниз*

Canvas.ForceUpdateCanvases();

outputText.transform.parent.GetComponent<ScrollRect>().verticalNormalizedPosition = 0f;

}

**🔍 Почему так?**

* **TextMeshPro** (TextMeshProUGUI) — это отдельный компонент, несовместимый напрямую со стандартным Text.
* ScrollView должен содержать **TMP-Text** в Content, чтобы корректно работать.

Если после замены текст не отображается, проверь:

* Назначен ли **TMP Font Asset** (например, LiberationSans SDF).
* Не перекрывает ли OutputText\_TMP другие элементы (попробуй увеличить его размер).

Если нужна помощь с настройкой ScrollRect или шрифтов — спрашивай! 😊

Нет, в Scroll View у меня отображается обычный Text

Понял! Если в твоем ScrollView используется обычный Text (из UnityEngine.UI), а не TextMeshPro, то тебе нужно **либо** заменить его на TextMeshPro, **либо** оставить как есть и просто правильно подключить в скрипте.

**🔹 Вариант 1: Оставить обычный Text (если не хочешь переходить на TMP)**

Просто убедись, что в скрипте DevConsole поле outputText имеет тип UnityEngine.UI.Text (как у тебя сейчас), и в инспекторе оно привязано к Text, который находится внутри ScrollView.

**Как проверить:**

1. Разверни ScrollView в иерархии (Hierarchy).
2. Найди внутри Viewport → Content → Text (это стандартный UnityEngine.UI.Text).
3. Перетащи его в поле outputText в инспекторе DevConsole.

**🔹 Вариант 2: Заменить Text на TextMeshPro (рекомендуется)**

Если хочешь использовать TextMeshPro для вывода текста, сделай так:

**1. Удали старый Text:**

* Найди Text внутри ScrollView/Viewport/Content.
* Удали его (правой кнопкой → **Delete** или Delete на клавиатуре).

**2. Добавь TextMeshPro - Text:**

* Кликни ПКМ на Content → **UI → TextMeshPro - Text**.
* Назови его, например, ConsoleOutput\_TMP.

**3. Настрой TextMeshPro:**

* Выбери шрифт (например, LiberationSans SDF).
* Настрой Font Size, Color и другие параметры.

**4. Обнови скрипт:**

Замени:

csharp

Copy

Download

public Text outputText; *// Устаревший вариант (если остаешься на UI.Text)*

на:

csharp

Copy

Download

public TMPro.TextMeshProUGUI outputText; *// Новый вариант для TMP*

**5. Перетащи ConsoleOutput\_TMP в поле outputText в инспекторе.**

**🔹 Почему может не работать?**

* ❌ **Поле outputText не привязано** — убедись, что в инспекторе DevConsole в поле outputText перетащен нужный Text или TextMeshProUGUI.
* ❌ **Шрифт не назначен в TMP** — если используешь TextMeshPro, проверь, что у него выбран Font Asset (SDF-шрифт).
* ❌ **Текст скрыт** — проверь Viewport и Mask, чтобы текст не обрезался.

Если хочешь, могу помочь с настройкой ScrollRect или автоматической прокруткой вниз при новом сообщении. 😊

Какой вариант тебе ближе — оставить Text или перейти на TextMeshPro?

добавить: Историю команд (стрелки ↑↓). Автодополнение. Настройку прав администратора. Консоль в стиле Quake или Half-Life.

Отличные идеи! Давай улучшим твою консоль, добавив историю команд, автодополнение, админ-права и стиль Quake/Half-Life. Вот обновлённый код с комментариями:

**🔧 Улучшенный DevConsole.cs**

csharp

Copy

Download

using UnityEngine;

using TMPro;

using System.Collections.Generic;

using System.Linq;

public class DevConsole : MonoBehaviour

{

[Header("UI References")]

public GameObject consolePanel;

public TMP\_InputField inputField;

public TextMeshProUGUI outputText;

public KeyCode toggleKey = KeyCode.BackQuote;

[Header("Settings")]

public bool adminModeRequired = true;

public string adminPassword = "12345";

public float fadeSpeed = 1f;

private bool isConsoleVisible = false;

private bool isAdmin = false;

private List<string> commandHistory = new List<string>();

private int historyIndex = -1;

private CanvasGroup canvasGroup;

private float targetAlpha = 0f;

private Dictionary<string, string> commands = new Dictionary<string, string>()

{

{"help", "Show all commands"},

{"god", "Toggle god mode"},

{"killall", "Destroy all enemies"},

{"teleport", "Teleport to coordinates x y z"},

{"admin", "Enter admin mode"}

};

void Start()

{

canvasGroup = consolePanel.GetComponent<CanvasGroup>();

consolePanel.SetActive(false);

canvasGroup.alpha = 0;

*// Quake-style appearance*

consolePanel.GetComponent<Image>().color = new Color(0.1f, 0.1f, 0.1f, 0.85f);

outputText.color = Color.green;

}

void Update()

{

*// Плавное появление/исчезновение*

canvasGroup.alpha = Mathf.MoveTowards(canvasGroup.alpha, targetAlpha, fadeSpeed \* Time.deltaTime);

if (Input.GetKeyDown(toggleKey))

{

ToggleConsole();

}

if (!isConsoleVisible) return;

HandleInput();

}

void ToggleConsole()

{

isConsoleVisible = !isConsoleVisible;

targetAlpha = isConsoleVisible ? 1f : 0f;

consolePanel.SetActive(true);

if (isConsoleVisible)

{

inputField.ActivateInputField();

}

else

{

historyIndex = -1;

}

}

void HandleInput()

{

*// История команд*

if (Input.GetKeyDown(KeyCode.UpArrow))

{

if (commandHistory.Count == 0) return;

historyIndex = Mathf.Min(historyIndex + 1, commandHistory.Count - 1);

inputField.text = commandHistory[commandHistory.Count - 1 - historyIndex];

inputField.caretPosition = inputField.text.Length;

}

else if (Input.GetKeyDown(KeyCode.DownArrow))

{

if (historyIndex <= 0)

{

historyIndex = -1;

inputField.text = "";

return;

}

historyIndex--;

inputField.text = commandHistory[commandHistory.Count - 1 - historyIndex];

inputField.caretPosition = inputField.text.Length;

}

*// Автодополнение при нажатии Tab*

if (Input.GetKeyDown(KeyCode.Tab) && !string.IsNullOrEmpty(inputField.text))

{

AutoComplete();

}

*// Отправка команды*

if (Input.GetKeyDown(KeyCode.Return))

{

ProcessCommand(inputField.text);

}

}

void AutoComplete()

{

string partial = inputField.text.ToLower();

var matches = commands.Keys.Where(cmd => cmd.StartsWith(partial)).ToList();

if (matches.Count == 1)

{

inputField.text = matches[0];

inputField.caretPosition = inputField.text.Length;

}

else if (matches.Count > 1)

{

AppendMessage($"Possible commands: {string.Join(", ", matches)}");

}

}

void ProcessCommand(string command)

{

if (string.IsNullOrWhiteSpace(command)) return;

commandHistory.Add(command);

historyIndex = -1;

AppendMessage("> " + command);

inputField.text = "";

inputField.ActivateInputField();

string[] parts = command.Split(' ');

string cmd = parts[0].ToLower();

switch (cmd)

{

case "help":

foreach (var kvp in commands)

{

AppendMessage($"{kvp.Key}: {kvp.Value}");

}

break;

case "admin":

if (parts.Length > 1 && parts[1] == adminPassword)

{

isAdmin = true;

AppendMessage("Admin mode activated!");

}

else

{

AppendMessage("Invalid password!");

}

break;

case "god":

if (!adminModeRequired || isAdmin)

{

*// Реализация god mode*

AppendMessage("God mode toggled");

}

else

{

AppendMessage("Admin rights required!");

}

break;

*// ... остальные команды*

}

}

void AppendMessage(string message)

{

outputText.text += message + "\n";

*// Автопрокрутка*

Canvas.ForceUpdateCanvases();

var scrollRect = outputText.GetComponentInParent<ScrollRect>();

scrollRect.verticalNormalizedPosition = 0f;

}

}

**🎨 Визуальные улучшения (Quake-стиль)**

1. Создай UI с такими параметрами:
   * Фон консоли: темный с прозрачностью (~85%)
   * Шрифт: моноширинный (например, Courier New в TMP)
   * Цвет текста: зеленый (как в матрице) или оранжевый (как в Half-Life)
2. Добавь эффекты:

csharp

Copy

Download

*// В Start()*

consolePanel.GetComponent<Image>().color = new Color(0.1f, 0.1f, 0.1f, 0.85f);

outputText.color = new Color(0.2f, 1f, 0.2f); *// Зеленый*

**🔒 Система администрирования**

1. Добавь проверку прав перед выполнением команд:

csharp

Copy

Download

if (!isAdmin && adminModeRequired)

{

AppendMessage("Access denied! Use 'admin [password]'");

return;

}

1. Команда для входа:

Copy

Download

admin 12345

**⚙️ Дополнительные улучшения**

1. **Сохраняйте историю** между сессиями (используя PlayerPrefs)
2. **Логирование** всех команд в файл
3. **Конфигурацию через JSON** (пароль, список команд)
4. **Звуковые эффекты** открытия/закрытия

Хочешь, чтобы я подробнее объяснил какой-то из этих пунктов или добавил новые функции? 😊

MissingComponentException: There is no 'CanvasGroup' attached to the "DevConsolePanel" game object, but a script is trying to access it. You probably need to add a CanvasGroup to the game object "DevConsolePanel". Or your script needs to check if the component is attached before using it. UnityEngine.Object+MarshalledUnityObject.TryThrowEditorNullExceptionObject (UnityEngine.Object unityObj, System.String parameterName) (at <89527d299cb64ba88f87240781b415e5>:0) UnityEngine.Bindings.ThrowHelper.ThrowNullReferenceException (System.Object obj) (at <89527d299cb64ba88f87240781b415e5>:0) UnityEngine.CanvasGroup.set\_alpha (System.Single value) (at <4b670d7243cc40f19dc0db888c7f9998>:0) DevConsole.Start () (at Assets/DarkTree FPS/Scripts/DevConsole/DevConsole.cs:41)

Ошибка возникает потому, что мы пытаемся получить компонент CanvasGroup у consolePanel, но он не был добавлен заранее. Давай исправим это и улучшим обработку ошибок!

**🔧 Исправленная версия кода (с проверками)**

csharp

Copy

Download

using UnityEngine;

using TMPro;

using System.Collections.Generic;

using System.Linq;

public class DevConsole : MonoBehaviour

{

[Header("UI References")]

public GameObject consolePanel;

public TMP\_InputField inputField;

public TextMeshProUGUI outputText;

public KeyCode toggleKey = KeyCode.BackQuote;

[Header("Settings")]

public bool adminModeRequired = true;

public string adminPassword = "12345";

public float fadeSpeed = 1f;

private bool isConsoleVisible = false;

private bool isAdmin = false;

private List<string> commandHistory = new List<string>();

private int historyIndex = -1;

private CanvasGroup canvasGroup;

private float targetAlpha = 0f;

void Start()

{

*// Добавляем CanvasGroup, если его нет*

if (!consolePanel.TryGetComponent(out canvasGroup))

{

canvasGroup = consolePanel.AddComponent<CanvasGroup>();

Debug.Log("CanvasGroup added automatically");

}

*// Настройки по умолчанию*

canvasGroup.alpha = 0;

consolePanel.SetActive(false);

*// Quake-style настройки*

if (consolePanel.TryGetComponent(out UnityEngine.UI.Image panelImage))

{

panelImage.color = new Color(0.1f, 0.1f, 0.1f, 0.85f);

}

if (outputText != null)

{

outputText.color = Color.green;

}

}

void Update()

{

*// Плавное появление только если есть CanvasGroup*

if (canvasGroup != null)

{

canvasGroup.alpha = Mathf.MoveTowards(canvasGroup.alpha, targetAlpha, fadeSpeed \* Time.deltaTime);

}

if (Input.GetKeyDown(toggleKey))

{

ToggleConsole();

}

if (!isConsoleVisible) return;

HandleInput();

}

*// ... остальные методы остаются без изменений ...*

}

**🔨 Как исправить в редакторе Unity:**

1. **Выбери DevConsolePanel** в иерархии
2. **Добавь компонент**:
   * Нажми Add Component в инспекторе
   * Найди и добавь Canvas Group
3. **Настрой параметры**:
   * Alpha: 0
   * Interactable: true
   * Blocks Raycasts: true

**⚠️ Важные проверки:**

1. Убедись что consolePanel не null:

csharp

Copy

Download

if (consolePanel == null)

{

Debug.LogError("Console Panel is not assigned!");

return;

}

1. Для автоматического добавления компонентов можно использовать:

csharp

Copy

Download

[ContextMenu("Setup Console")]

void SetupConsole()

{

if (!consolePanel.TryGetComponent(out canvasGroup))

{

canvasGroup = consolePanel.AddComponent<CanvasGroup>();

}

if (!consolePanel.TryGetComponent(out UnityEngine.UI.Image img))

{

img = consolePanel.AddComponent<UnityEngine.UI.Image>();

img.color = new Color(0.1f, 0.1f, 0.1f, 0.85f);

}

}

(Вызови через правый клик на компоненте в инспекторе)

**💡 Почему это важно?**

* CanvasGroup позволяет управлять прозрачностью всей панели сразу
* Автоматическое добавление компонентов делает скрипт более устойчивым
* Проверки TryGetComponent предотвращают ошибки

Теперь консоль должна работать без ошибок! Если нужно что-то уточнить — спрашивай. 😊

При включении.выключении консоли в поле ввода добавляются `````

Проблема в том, что при нажатии клавиши **BackQuote** (`) срабатывает и ввод символа, и переключение консоли. Вот как это исправить:

**🔧 Исправленный код (фокус на InputField)**

csharp

Copy

Download

void Update()

{

*// Плавное появление/исчезновение*

if (canvasGroup != null)

{

canvasGroup.alpha = Mathf.MoveTowards(canvasGroup.alpha, targetAlpha, fadeSpeed \* Time.deltaTime);

}

*// Перехватываем ввод, если консоль видима*

if (isConsoleVisible)

{

*// Блокируем ввод ` при открытой консоли*

if (Input.GetKeyDown(toggleKey))

{

ToggleConsole();

return; *// Прерываем дальнейшую обработку*

}

HandleInput();

}

else if (Input.GetKeyDown(toggleKey))

{

ToggleConsole();

}

}

void ToggleConsole()

{

isConsoleVisible = !isConsoleVisible;

targetAlpha = isConsoleVisible ? 1f : 0f;

if (isConsoleVisible)

{

consolePanel.SetActive(true);

inputField.text = ""; *// Очищаем поле при открытии*

inputField.ActivateInputField();

}

else

{

historyIndex = -1;

*// Не деактивируем сразу, чтобы анимация завершилась*

if (fadeSpeed <= 0) consolePanel.SetActive(false);

}

}

**🔥 Альтернативное решение (полная блокировка символа)**

Добавь этот код в метод HandleInput():

csharp

Copy

Download

void HandleInput()

{

*// Блокируем символ ` в любом случае*

if (inputField.text.Contains("`"))

{

inputField.text = inputField.text.Replace("`", "");

inputField.caretPosition = inputField.text.Length;

}

*// Остальная логика обработки ввода...*

}

**🛡️ Дополнительная защита**

1. **В инспекторе Unity**:
   * Убедись, что в InputField → On Validate Input нет обработки символа `
2. **Через настройки Input**:

csharp

Copy

Download

void Start()

{

inputField.onValidateInput = (text, charIndex, addedChar) =>

{

return addedChar == '`' ? '\0' : addedChar;

};

}

**Почему это работает:**

1. Мы **прерываем обработку ввода**, если консоль открыта и нажата `
2. **Очищаем поле** при каждом открытии
3. Полностью **блокируем ввод** этого символа через систему валидации

Выбери любой способ или используй их комбинацию. Теперь символ ` не будет мешать работе консоли! 😊

Проблема. Если при вводе в Input Field раскладка клавиатуры была русская, то набранный текст остается при закрытии консоли и выводится при открытии. вообще нужно сделать чтобы не сохранялись сторки которые not appropriate

Чтобы решить проблему с сохранением нежелательного текста (включая текст на русской раскладке) при закрытии/открытии консоли, предлагаю следующие улучшения:

**🔧 Модифицированный код (с очисткой и фильтрацией)**

csharp

Copy

Download

using System.Text.RegularExpressions;

private string lastValidInput = ""; *// Хранит последний валидный ввод*

void ToggleConsole()

{

isConsoleVisible = !isConsoleVisible;

targetAlpha = isConsoleVisible ? 1f : 0f;

if (isConsoleVisible)

{

consolePanel.SetActive(true);

inputField.text = lastValidInput; *// Восстанавливаем ТОЛЬКО валидный текст*

inputField.ActivateInputField();

}

else

{

*// Сохраняем только если текст соответствует критериям*

if (IsInputValid(inputField.text))

{

lastValidInput = inputField.text;

}

else

{

lastValidInput = "";

}

inputField.text = ""; *// Всегда очищаем при закрытии*

historyIndex = -1;

}

}

bool IsInputValid(string input)

{

if (string.IsNullOrWhiteSpace(input)) return false;

*// Регулярное выражение: только латиница, цифры и основные символы*

var regex = new Regex(@"^[a-zA-Z0-9\_\-\+\=\.,!@#\$%\^\&\\*\(\)\s]+$");

return regex.IsMatch(input);

}

void HandleInput()

{

*// Автоматическая очистка невалидных символов*

if (!IsInputValid(inputField.text))

{

inputField.text = Regex.Replace(inputField.text, @"[^a-zA-Z0-9\_\-\+\=\.,!@#\$%\^\&\\*\(\)\s]", "");

inputField.caretPosition = inputField.text.Length;

}

*// Остальная логика обработки ввода...*

}

**🛡️ Дополнительные улучшения:**

1. **Фильтрация в реальном времени**:

csharp

Copy

Download

inputField.onValueChanged.AddListener((text) =>

{

if (!IsInputValid(text))

{

inputField.text = lastValidInput;

}

else

{

lastValidInput = text;

}

});

1. **Настройка через Inspector**:

csharp

Copy

Download

[Header("Input Validation")]

[Tooltip("Регулярное выражение для валидации ввода")]

public string inputValidationPattern = @"^[a-zA-Z0-9\_\-\+\=\.,!@#\$%\^\&\\*\(\)\s]+$";

1. **Автоматическая транслитерация** (опционально):

csharp

Copy

Download

string Transliterate(string input)

{

Dictionary<string, string> translit = new Dictionary<string, string>()

{

{"й","j"}, {"ц","c"}, {"у","u"}, */\*...полная таблица...\*/*

};

return translit.Aggregate(input, (current, pair) =>

current.Replace(pair.Key, pair.Value));

}

**Почему это работает:**

1. Всегда очищаем поле при закрытии (inputField.text = "")
2. Сохраняем только явно разрешенные символы через регулярные выражения
3. Восстанавливаем только предварительно проверенный текст
4. Реагируем на изменения в реальном времени

Дополнительно можно:

* Добавить звуковой сигнал при невалидном вводе
* Показывать всплывающую подсказку о разрешенных символах
* Реализовать "мягкую" очистку (постепенное исчезновение нежелательного текста)

Какой вариант вам больше подходит? Можно настроить строгость валидации под конкретные нужды. 😊

Проблема. Сейчас после ввода 'god' в текстовом поле по центру следующее: New Text> god Admin rights Полный текст скрипта сейчас: using UnityEngine; using TMPro; using System.Collections.Generic; using System.Linq; using UnityEngine.UI; using System.Text.RegularExpressions; public class DevConsole : MonoBehaviour { [Header("UI References")] public GameObject consolePanel; public TMP\_InputField inputField; public TextMeshProUGUI outputText; public KeyCode toggleKey = KeyCode.BackQuote; [Header("Settings")] public bool adminModeRequired = true; public string adminPassword = "12345"; public float fadeSpeed = 1f; private bool isConsoleVisible = false; private bool isAdmin = false; private List<string> commandHistory = new List<string>(); private int historyIndex = -1; private CanvasGroup canvasGroup; private float targetAlpha = 0f; private string lastValidInput = ""; // Хранит последний валидный ввод private Dictionary<string, string> commands = new Dictionary<string, string>() { {"help", "Show all commands"}, {"god", "Toggle god mode"}, {"killall", "Destroy all enemies"}, {"teleport", "Teleport to coordinates x y z"}, {"admin", "Enter admin mode"} }; void Start() { // Добавляем CanvasGroup, если его нет if (!consolePanel.TryGetComponent(out canvasGroup)) { canvasGroup = consolePanel.AddComponent<CanvasGroup>(); Debug.Log("CanvasGroup added automatically"); } canvasGroup = consolePanel.GetComponent<CanvasGroup>(); consolePanel.SetActive(false); canvasGroup.alpha = 0; // Quake-style appearance consolePanel.GetComponent<Image>().color = new Color(0.1f, 0.1f, 0.1f, 0.85f); outputText.color = Color.green; // удаеляем ` если остался с прошлого раза inputField.onValidateInput = (text, charIndex, addedChar) => { return addedChar == '`' ? '\0' : addedChar; }; } void Update() { // Плавное появление только если есть CanvasGroup if (canvasGroup != null) { // Плавное появление/исчезновение canvasGroup.alpha = Mathf.MoveTowards(canvasGroup.alpha, targetAlpha, fadeSpeed \* Time.deltaTime); } if (Input.GetKeyDown(toggleKey)) { ToggleConsole(); } if (!isConsoleVisible) return; HandleInput(); } void ToggleConsole() { isConsoleVisible = !isConsoleVisible; targetAlpha = isConsoleVisible ? 1f : 0f; if (isConsoleVisible) { consolePanel.SetActive(true); inputField.text = lastValidInput; // Восстанавливаем ТОЛЬКО валидный текст inputField.ActivateInputField(); } else { // Сохраняем только если текст соответствует критериям if (IsInputValid(inputField.text)) { lastValidInput = inputField.text; } else { lastValidInput = ""; } inputField.text = ""; // Всегда очищаем при закрытии historyIndex = -1; } } bool IsInputValid(string input) { if (string.IsNullOrWhiteSpace(input)) return false; // Регулярное выражение: только латиница, цифры и основные символы var regex = new Regex(@"^[a-zA-Z0-9\_\-\+\=\.,!@#\$%\^\&\\*\(\)\s]+$"); return regex.IsMatch(input); } void HandleInput() { // Автоматическая очистка невалидных символов if (!IsInputValid(inputField.text)) { inputField.text = Regex.Replace(inputField.text, @"[^a-zA-Z0-9\_\-\+\=\.,!@#\$%\^\&\\*\(\)\s]", ""); inputField.caretPosition = inputField.text.Length; } // История команд if (Input.GetKeyDown(KeyCode.UpArrow)) { if (commandHistory.Count == 0) return; historyIndex = Mathf.Min(historyIndex + 1, commandHistory.Count - 1); inputField.text = commandHistory[commandHistory.Count - 1 - historyIndex]; inputField.caretPosition = inputField.text.Length; } else if (Input.GetKeyDown(KeyCode.DownArrow)) { if (historyIndex <= 0) { historyIndex = -1; inputField.text = ""; return; } historyIndex--; inputField.text = commandHistory[commandHistory.Count - 1 - historyIndex]; inputField.caretPosition = inputField.text.Length; } // Автодополнение при нажатии Tab if (Input.GetKeyDown(KeyCode.Tab) && !string.IsNullOrEmpty(inputField.text)) { AutoComplete(); } // Отправка команды if (Input.GetKeyDown(KeyCode.Return)) { ProcessCommand(inputField.text); } } void AutoComplete() { string partial = inputField.text.ToLower(); var matches = commands.Keys.Where(cmd => cmd.StartsWith(partial)).ToList(); if (matches.Count == 1) { inputField.text = matches[0]; inputField.caretPosition = inputField.text.Length; } else if (matches.Count > 1) { AppendMessage($"Possible commands: {string.Join(", ", matches)}"); } } void ProcessCommand(string command) { if (string.IsNullOrWhiteSpace(command)) return; commandHistory.Add(command); historyIndex = -1; AppendMessage("> " + command); inputField.text = ""; inputField.ActivateInputField(); string[] parts = command.Split(' '); string cmd = parts[0].ToLower(); switch (cmd) { case "help": foreach (var kvp in commands) { AppendMessage($"{kvp.Key}: {kvp.Value}"); } break; case "admin": if (parts.Length > 1 && parts[1] == adminPassword) { isAdmin = true; AppendMessage("Admin mode activated!"); } else { AppendMessage("Invalid password!"); } break; case "god": if (!adminModeRequired || isAdmin) { // Реализация god mode AppendMessage("God mode toggled"); } else { AppendMessage("Admin rights required!"); } break; // ... остальные команды } } void AppendMessage(string message) { outputText.text += message + "\n"; // Автопрокрутка Canvas.ForceUpdateCanvases(); var scrollRect = outputText.GetComponentInParent<ScrollRect>(); scrollRect.verticalNormalizedPosition = 0f; } } -Надо чтобы команды и описание действий слева отображались

Чтобы выровнять текст в консоли с левой стороны и сделать вывод более структурированным, внесем изменения в метод AppendMessage и обработку команд. Вот модифицированный код:

**🔧 Исправленный вывод (левое выравнивание)**

csharp

Copy

Download

void AppendMessage(string message)

{

*// Добавляем отступ слева и перенос строки*

outputText.text += " " + message + "\n";

*// Автопрокрутка вниз*

Canvas.ForceUpdateCanvases();

var scrollRect = outputText.GetComponentInParent<ScrollRect>();

scrollRect.verticalNormalizedPosition = 0f;

}

void ProcessCommand(string command)

{

if (string.IsNullOrWhiteSpace(command)) return;

commandHistory.Add(command);

historyIndex = -1;

*// Вывод команды с левым отступом*

AppendMessage("> " + command.Trim());

inputField.text = "";

inputField.ActivateInputField();

string[] parts = command.Split(' ');

string cmd = parts[0].ToLower();

switch (cmd)

{

case "help":

*// Красивый вывод списка команд*

AppendMessage("<b>Available commands:</b>");

foreach (var kvp in commands)

{

AppendMessage($" <b>{kvp.Key.PadRight(10)}</b> - {kvp.Value}");

}

break;

case "admin":

if (parts.Length > 1 && parts[1] == adminPassword)

{

isAdmin = true;

AppendMessage("<color=green>Admin mode activated!</color>");

}

else

{

AppendMessage("<color=red>Invalid password!</color>");

}

break;

case "god":

if (!adminModeRequired || isAdmin)

{

AppendMessage("<color=yellow>God mode toggled</color>");

}

else

{

AppendMessage("<color=red>Admin rights required!</color>");

}

break;

}

}

**🎨 Дополнительные улучшения:**

1. **Настройка TextMeshPro**:
   * Убедитесь, что у outputText в инспекторе:
     + Alignment установлен в **Left** (по левому краю)
     + Extra Padding → Left = 10 (для отступа)
2. **Обновление Start()**:

csharp

Copy

Download

void Start()

{

*// ... предыдущий код ...*

*// Настройка выравнивания текста*

if (outputText != null)

{

outputText.alignment = TextAlignmentOptions.Left;

outputText.margin = new Vector4(10, 0, 0, 0); *// Левый отступ 10px*

}

}

1. **Форматирование текста**:
   * Используйте TMP-теги для стилизации:
     + <b>текст</b> - жирный
     + <i>текст</i> - курсив
     + <color=green>текст</color> - цветной текст

**🔄 Почему это работает:**

1. **PadRight(10)** создает фиксированную ширину для команд
2. **Левое выравнивание** в настройках TMP гарантирует одинаковый отступ
3. **HTML-теги** TextMeshPro позволяют стилизовать вывод без сложного кода

Теперь вывод будет выглядеть примерно так:

Copy

Download

> god

God mode toggled

> help

Available commands:

help - Show all commands

god - Toggle god mode

killall - Destroy all enemies

Для дополнительного улучшения можно добавить:

* Разделительные линии (AppendMessage("----------------"))
* Цветовые коды для разных типов сообщений
* Временные метки команд

Хотите, чтобы я добавил какие-то из этих функций? 😊

Проблема. Scroll View сейчас перекрывается полем Input Field. Они дочерние DevConsolePanel. Никаких скриптов на панели нет. как исправить?

Чтобы исправить перекрытие **Scroll View** полем **Input Field**, нужно правильно настроить **RectTransform** и иерархию UI-элементов. Вот пошаговое решение:

**🔧 1. Проверьте иерархию UI-элементов**

Должно быть так:

Copy

Download

DevConsolePanel (Panel)

├── ScrollView (ScrollRect)

│ ├── Viewport (Mask)

│ │ └── Content (внутри лежит TextMeshProUGUI для вывода)

│ └── Scrollbar (если нужен)

└── InputField (TMP\_InputField)

Если InputField находится **внутри** ScrollView – вынесите его наружу, сделав **соседним элементом**.

**🛠 2. Настройте RectTransform**

**Для ScrollView:**

1. **Anchor (якоря)**:  
   Установите **нижний край** (Bottom) так, чтобы оставить место для InputField (например, Bottom = 50).
   * **Min**: (0, 0)
   * **Max**: (1, 0)
   * **Pos Y**: 0
   * **Height**: -50 (или другое значение, в зависимости от высоты InputField).
2. **Pivot (точка привязки)**:  
   (0.5, 1) (верхний центр), чтобы контент начинался сверху.

**Для InputField:**

1. **Anchor (якоря)**:  
   Привяжите к нижней части панели:
   * **Min**: (0, 0)
   * **Max**: (1, 0)
   * **Pos Y**: 25 (половина его высоты)
   * **Height**: 50 (или другое удобное значение).

**🎨 3. Дополнительные настройки**

1. **Проверьте Viewport и Content в ScrollView**:
   * У Viewport должен быть включен **Mask**, чтобы текст не выходил за границы.
   * Content должен иметь **Vertical Layout Group** (если нужен автоматический размер) или **Content Size Fitter** (если контент динамический).
2. **Прокрутка (ScrollRect)**:
   * В компоненте ScrollRect укажите:
     + **Content** — ваш Content с текстом.
     + **Vertical Scrollbar** (если есть скроллбар).

**📐 Пример правильных настроек RectTransform**

| **Элемент** | **Anchor Min** | **Anchor Max** | **Pos Y** | **Height** |
| --- | --- | --- | --- | --- |
| **ScrollView** | (0, 0) | (1, 0) | 0 | -50 |
| **InputField** | (0, 0) | (1, 0) | 25 | 50 |

**💡 Если текст не прокручивается**

Добавьте в метод AppendMessage():

csharp

Copy

Download

*// После добавления текста:*

Canvas.ForceUpdateCanvases();

ScrollRect scrollRect = outputText.GetComponentInParent<ScrollRect>();

scrollRect.verticalNormalizedPosition = 0; *// Прокрутка вниз*

**🔄 Итог**

После этих настроек:

* **ScrollView** будет занимать всю панель, кроме нижней части (где InputField).
* **InputField** останется фиксированным внизу и не будет перекрывать вывод.

Если нужно больше свободного места для текста — просто увеличьте Height у ScrollView (например, -70) и сдвиньте InputField ниже (Pos Y = 35).