19CAPtcha84

0.1.0

Создано системой Doxygen 1.13.2

Глава 1

Иерархический список классов

1.1 Иерархия классов

Иерархия классов.

Глава 2

Алфавитный указатель классов

2.1 Классы

Классы с их кратким описанием.

Глава 3

Список файлов

3.1 Файлы

Полный список документированных файлов.

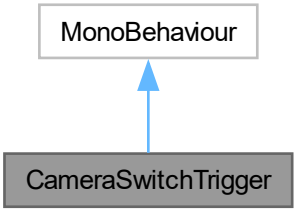
Глава 4

Классы

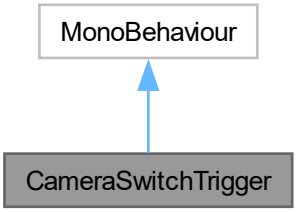
4.1 Класс CameraSwitchTrigger

Manages switching between different cameras and control systems.

Граф наследования:CameraSwitchTrigger:



Граф связей класса CameraSwitchTrigger:



Открытые атрибуты

- GameObject playerObject
- GameObject mainCamera
- GameObject alternateCamera
- MonoBehaviour mainPlayerController
- MonoBehaviour alternateController
- KeyCode switchKey = KeyCode.F
- bool hidePlayer = true

4.1.1 Подробное описание

Manages switching between different cameras and control systems.

This component allows toggling between a main camera/controller and an alternate camera/controller when the player enters a trigger zone and presses a designated key

См. определение в файле CameraSwitchController.cs строка 10

4.1.2 Данные класса

4.1.2.1 alternateCamera

GameObject CameraSwitchTrigger.alternateCamera

См. определение в файле CameraSwitchController.cs строка 15

4.1.2.2 alternateController

MonoBehaviour CameraSwitchTrigger.alternateController

См. определение в файле CameraSwitchController.cs строка 17

4.1.2.3 hidePlayer

bool CameraSwitchTrigger.hidePlayer = true

См. определение в файле CameraSwitchController.cs строка 21

4.1.2.4 mainCamera

GameObject CameraSwitchTrigger.mainCamera

См. определение в файле CameraSwitchController.cs строка 14

4.1.2.5 mainPlayerController

MonoBehaviour CameraSwitchTrigger.mainPlayerController

См. определение в файле CameraSwitchController.cs строка 16

4.1.2.6 playerObject

GameObject CameraSwitchTrigger.playerObject

См. определение в файле CameraSwitchController.cs строка 13

4.1.2.7 switchKey

KeyCode CameraSwitchTrigger.switchKey = KeyCode.F

См. определение в файле CameraSwitchController.cs строка 20

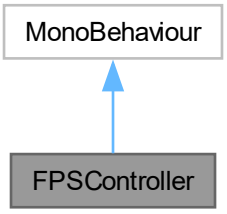
Объявления и описания членов класса находятся в файле:

- Assets/Scripts/CameraSwitchController.cs

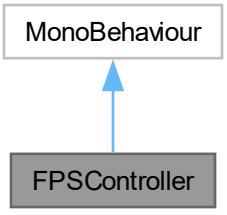
4.2 Класс FPSController

First Person Shooter character controller.

Граф наследования:FPSController:



Граф связей класса FPSController:



Открытые атрибуты

- float moveSpeed = 5f
- float mouseSensitivity = 2f
- Transform cameraTransform

4.2.1 Подробное описание

First Person Shooter character controller.

This class handles movement and camera control for a first-person character

См. определение в файле FPSController.cs строка 8

4.2.2 Данные класса

4.2.2.1 cameraTransform

Transform FPSController.cameraTransform

См. определение в файле FPSController.cs строка 12

4.2.2.2 mouseSensitivity

float FPSController.mouseSensitivity = 2f

См. определение в файле FPSController.cs строка 11

4.2.2.3 moveSpeed

float FPSController.moveSpeed = 5f

См. определение в файле FPSController.cs строка 10

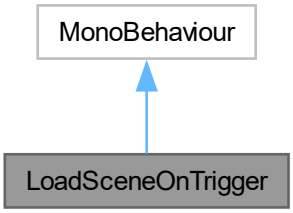
Объявления и описания членов класса находятся в файле:

- Assets/Scripts/FPSController.cs

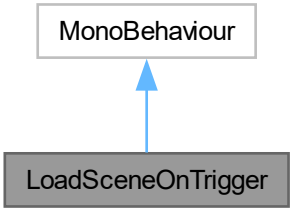
4.3 Класс LoadSceneOnTrigger

Loads a new scene when triggered.

Граф наследования:LoadSceneOnTrigger:



Граф связей класса LoadSceneOnTrigger:



4.3.1 Подробное описание

Loads a new scene when triggered.

This component loads a specified scene when a ball enters its trigger collider

См. определение в файле LoadSceneOnTrigger.cs строка 9

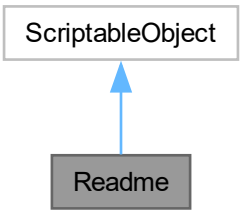
Объявления и описания членов класса находятся в файле:

- Assets/Scripts/LoadSceneOnTrigger.cs

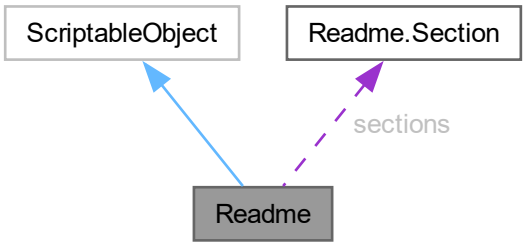
4.4 Класс Readme

ScriptableObject for displaying readme information in the Unity Editor.

Граф наследования:Readme:



Граф связей класса Readme:



Классы

- class Section

  Represents a section of content in the readme.

Открытые атрибуты

- Texture2D icon
- string title
- Section[] sections
- bool loadedLayout

4.4.1 Подробное описание

ScriptableObject for displaying readme information in the Unity Editor.

This class provides a structured way to create and display tutorial or documentation information within the Unity Editor interface

См. определение в файле Readme.cs строка 10

4.4.2 Данные класса

4.4.2.1 icon

Texture2D Readme.icon

См. определение в файле Readme.cs строка 12

4.4.2.2 loadedLayout

bool Readme.loadedLayout

См. определение в файле Readme.cs строка 15

4.4.2.3 sections

Section [] Readme.sections

См. определение в файле Readme.cs строка 14

4.4.2.4 title

string Readme.title

См. определение в файле Readme.cs строка 13

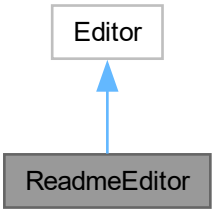
Объявления и описания членов класса находятся в файле:

- Assets/TutorialInfo/Scripts/Readme.cs

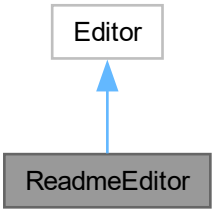
4.5 Класс ReadmeEditor

Custom editor for the Readme ScriptableObject.

Граф наследования:ReadmeEditor:



Граф связей класса ReadmeEditor:



Открытые члены

- override void OnInspectorGUI ()
  Renders the main inspector GUI.

Защищенные члены

- override void OnHeaderGUI ()
  Customizes the header section of the inspector.

4.5.1 Подробное описание

Custom editor for the Readme ScriptableObject.

This editor automatically displays the readme when the project is opened and provides a custom inspector UI for displaying the readme content

См. определение в файле ReadmeEditor.cs строка 17

4.5.2 Методы

4.5.2.1 OnHeaderGUI()

override void ReadmeEditor.OnHeaderGUI () [protected]

Customizes the header section of the inspector.

Displays the readme icon and title

См. определение в файле ReadmeEditor.cs строка 132

4.5.2.2 OnInspectorGUI()

override void ReadmeEditor.OnInspectorGUI ()

Renders the main inspector GUI.

Displays all sections of the readme with their headings, text, and links

См. определение в файле ReadmeEditor.cs строка 165

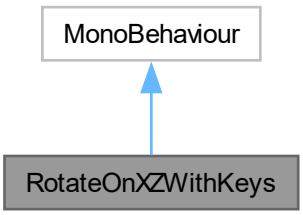
Объявления и описания членов класса находятся в файле:

- Assets/TutorialInfo/Scripts/Editor/ReadmeEditor.cs

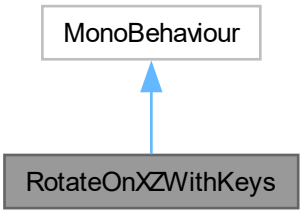
4.6 Класс RotateOnXZWithKeys

Enables keyboard-controlled rotation of an object.

Граф наследования:RotateOnXZWithKeys:



Граф связей класса RotateOnXZWithKeys:



Открытые атрибуты

- float rotationSpeed = 100f

4.6.1 Подробное описание

Enables keyboard-controlled rotation of an object.

This component allows rotating an object around its X, Y, and Z axes using keyboard input

См. определение в файле RotateOnXZWithKeys.cs строка 8

4.6.2 Данные класса

4.6.2.1 rotationSpeed

float RotateOnXZWithKeys.rotationSpeed = 100f

См. определение в файле RotateOnXZWithKeys.cs строка 10

Объявления и описания членов класса находятся в файле:

- Assets/Scripts/RotateOnXZWithKeys.cs

4.7 Класс Readme.Section

Represents a section of content in the readme.

Открытые атрибуты

- string heading
- string text
- string linkText
- string url

4.7.1 Подробное описание

Represents a section of content in the readme.

Each section can contain a heading, text content, and an optional link

См. определение в файле Readme.cs строка 23

4.7.2 Данные класса

4.7.2.1 heading

string Readme.Section.heading

См. определение в файле Readme.cs строка 25

4.7.2.2 linkText

string Readme.Section.linkText

См. определение в файле Readme.cs строка 27

4.7.2.3 text

string Readme.Section.text

См. определение в файле Readme.cs строка 26

4.7.2.4 url

string Readme.Section.url

См. определение в файле Readme.cs строка 28

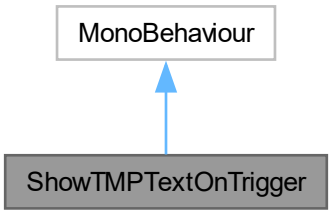
Объявления и описания членов класса находятся в файле:

- Assets/TutorialInfo/Scripts/Readme.cs

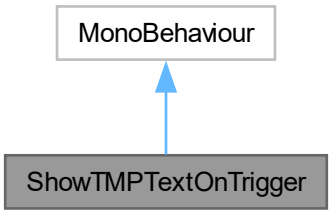
4.8 Класс ShowTMPTextOnTrigger

Controls the visibility of TextMeshPro text based on player position and view angle.

Граф наследования:ShowTMPTextOnTrigger:



Граф связей класса ShowTMPTextOnTrigger:



4.8.1 Подробное описание

Controls the visibility of TextMeshPro text based on player position and view angle.

This component shows/hides a TextMeshPro text element when the player enters a trigger and is looking at the object within a specified angle threshold

См. определение в файле ShowCanvasOnTrigger.cs строка 10

Объявления и описания членов класса находятся в файле:

- Assets/Scripts/ShowCanvasOnTrigger.cs

Глава 5

Файлы

5.1 CameraSwitchController.cs

```
00001 using UnityEngine;
00002
00010 public class CameraSwitchTrigger : MonoBehaviour
00011 {
00012     [Header("References")]
00013     public GameObject playerObject; // Player object (mesh/model)
00014     public GameObject mainCamera; // Main camera
00015     public GameObject alternateCamera; // Alternate camera
00016     public MonoBehaviour mainPlayerController; // Main player controller
00017     public MonoBehaviour alternateController; // Alternate controller
00018
00019     [Header("Settings")]
00020     public KeyCode switchKey = KeyCode.F; // Switch key
00021     public bool hidePlayer = true; // Whether to hide the player when switching
00022
00023     private bool isInsideTrigger = false;
00024     private bool isUsingAlternate = false; // Current state flag
00025
00031     private void Update()
00032     {
00033         if (isInsideTrigger && Input.GetKeyDown(switchKey))
00034         {
00035             ToggleCameraAndController();
00036         }
00037     }
00038
00045     private void ToggleCameraAndController()
00046     {
00047         isUsingAlternate = !isUsingAlternate;
00048
00049         // Switch cameras
00050         mainCamera.SetActive(!isUsingAlternate);
00051         alternateCamera.SetActive(isUsingAlternate);
00052
00053         // Switch controllers
00054         mainPlayerController.enabled = !isUsingAlternate;
00055
00056         if (alternateController != null)
00057         {
00058             alternateController.enabled = isUsingAlternate;
00059         }
00060
00061         // Hide/show player if needed
00062         if (hidePlayer && playerObject != null)
00063         {
00064             playerObject.SetActive(!isUsingAlternate);
00065         }
00066     }
00067
00075     private void OnTriggerEnter(Collider other)
00076     {
00077         if (other.CompareTag("Player"))
00078         {
00079             isInsideTrigger = true;
00080             // Can add UI hint here
00081         }
00082     }
```

```
00083
00092     private void OnTriggerExit(Collider other)
00093     {
00094         if (other.CompareTag("Player"))
00095         {
00096             isInsideTrigger = false;
00097             // Can remove UI hint here
00098
00099             // Automatically return to main control when exiting
00100             if (isUsingAlternate)
00101             {
00102                 ResetToMain();
00103             }
00104         }
00105     }
00106
00113     private void ResetToMain()
00114     {
00115         isUsingAlternate = false;
00116
00117         mainCamera.SetActive(true);
00118         alternateCamera.SetActive(false);
00119
00120         mainPlayerController.enabled = true;
00121
00122         if (alternateController != null)
00123         {
00124             alternateController.enabled = false;
00125         }
00126
00127         if (hidePlayer && playerObject != null)
00128         {
00129             playerObject.SetActive(true);
00130         }
00131     }
00132 }
```

5.2 FPSController.cs

```
00001 using UnityEngine;
00002
00008 public class FPSController : MonoBehaviour
00009 {
00010     public float moveSpeed = 5f;
00011     public float mouseSensitivity = 2f;
00012     public Transform cameraTransform;
00013
00014     private Rigidbody rb;
00015     private Vector3 movement;
00016     private float rotationX = 0f;
00017
00023     void Start()
00024     {
00025         rb = GetComponent<Rigidbody>();
00026         Cursor.lockState = CursorLockMode.Locked;
00027     }
00028
00034     void Update()
00035     {
00036         float mouseX = Input.GetAxis("Mouse X") * mouseSensitivity;
00037         float mouseY = Input.GetAxis("Mouse Y") * mouseSensitivity;
00038
00039         rotationX -= mouseY;
00040         rotationX = Mathf.Clamp(rotationX, -90f, 90f);
00041
00042         cameraTransform.localRotation = Quaternion.Euler(rotationX, 0, 0);
00043         transform.Rotate(Vector3.up * mouseX);
00044
00045         movement.x = Input.GetAxis("Horizontal");
00046         movement.z = Input.GetAxis("Vertical");
00047     }
00048
00055     void FixedUpdate()
00056     {
00057         Vector3 moveDirection = transform.right * movement.x + transform.forward * movement.z;
00058         rb.linearVelocity = new Vector3(moveDirection.x * moveSpeed, rb.linearVelocity.y, moveDirection.z * moveSpeed);
00059     }
00060 }
```

5.3 LoadSceneOnTrigger.cs

```
00001 using UnityEngine;
00002 using UnityEngine.SceneManagement;
00003
00009 public class LoadSceneOnTrigger : MonoBehaviour
00010 {
00011     [SerializeField] private string sceneToLoad = "SceneName"; // Scene name to load
00012     [SerializeField] private string ballTag = "Ball";       // Ball tag to detect
00013
00021     private void OnTriggerEnter(Collider other)
00022     {
00023         if (other.CompareTag(ballTag))
00024         {
00025             SceneManager.LoadScene(sceneToLoad);
00026         }
00027     }
00028 }
```

5.4 RotateOnXZWithKeys.cs

```
00001 using UnityEngine;
00002
00008 public class RotateOnXZWithKeys : MonoBehaviour
00009 {
00010     public float rotationSpeed = 100f; // Rotation speed
00011
00021     private void Update()
00022     {
00023         float rotationX = Input.GetAxis("Vertical") * rotationSpeed * Time.deltaTime;    // W/S
00024         float rotationZ = -Input.GetAxis("Horizontal") * rotationSpeed * Time.deltaTime; // A/D
00025
00026         float rotationY = 0f;
00027         if (Input.GetKey(KeyCode.Q))
00028         {
00029             rotationY = -rotationSpeed * Time.deltaTime;
00030         }
00031         else if (Input.GetKey(KeyCode.E))
00032         {
00033             rotationY = rotationSpeed * Time.deltaTime;
00034         }
00035
00036         transform.Rotate(rotationX, rotationY, rotationZ, Space.World);
00037     }
00038 }
```

5.5 ShowCanvasOnTrigger.cs

```
00001 using UnityEngine;
00002 using TMPro;
00003
00010 public class ShowTMPTextOnTrigger : MonoBehaviour
00011 {
00012     [SerializeField] private TMP_Text tmpText;
00013     [SerializeField] private float fadeDuration = 0.5f;
00014     [SerializeField] private string playerTag = "Player";
00015     [SerializeField] private float viewAngleThreshold = 45f;
00016     [SerializeField] private Camera playerCamera;
00017
00018     private CanvasGroup textCanvasGroup;
00019     private bool isPlayerInside = false;
00020     private bool isTextVisible = false;
00021
00028     private void Start()
00029     {
00030         if (tmpText != null)
00031         {
00032             textCanvasGroup = tmpText.GetComponent<CanvasGroup>();
00033             if (textCanvasGroup == null)
00034             {
00035                 textCanvasGroup = tmpText.gameObject.AddComponent<CanvasGroup>();
00036             }
00037             textCanvasGroup.alpha = 0f;
00038         }
00039         else
00040         {
00041             Debug.LogWarning("TMP_Text not assigned in the inspector!");
00042         }
```

```
00043
00044         if (playerCamera == null)
00045         {
00046             playerCamera = Camera.main;
00047         }
00048     }
00049
00057     private void OnTriggerEnter(Collider other)
00058     {
00059         if (other.CompareTag(playerTag))
00060         {
00061             isPlayerInside = true;
00062         }
00063     }
00064
00072     private void OnTriggerExit(Collider other)
00073     {
00074         if (other.CompareTag(playerTag))
00075         {
00076             isPlayerInside = false;
00077             isTextVisible = false;
00078             StopAllCoroutines();
00079             StartCoroutine(FadeText(0f));
00080         }
00081     }
00082
00089     private void Update()
00090     {
00091         if (isPlayerInside && textCanvasGroup != null)
00092         {
00093             bool looking = IsLookingAtObject();
00094
00095             if (looking && !isTextVisible)
00096             {
00097                 isTextVisible = true;
00098                 StopAllCoroutines();
00099                 StartCoroutine(FadeText(1f));
00100             }
00101             else if (!looking && isTextVisible)
00102             {
00103                 isTextVisible = false;
00104                 StopAllCoroutines();
00105                 StartCoroutine(FadeText(0f));
00106             }
00107         }
00108     }
00109
00116     private bool IsLookingAtObject()
00117     {
00118         Vector3 toObject = transform.position - playerCamera.transform.position;
00119         float angle = Vector3.Angle(playerCamera.transform.forward, toObject);
00120
00121         return angle <= viewAngleThreshold;
00122     }
00123
00131     private System.Collections.IEnumerator FadeText(float targetAlpha)
00132     {
00133         float startAlpha = textCanvasGroup.alpha;
00134         float time = 0f;
00135
00136         while (time < fadeDuration)
00137         {
00138             time += Time.deltaTime;
00139             textCanvasGroup.alpha = Mathf.Lerp(startAlpha, targetAlpha, time / fadeDuration);
00140             yield return null;
00141         }
00142
00143         textCanvasGroup.alpha = targetAlpha;
00144     }
00145 }
```

5.6 ReadmeEditor.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004 using UnityEditor;
00005 using System;
00006 using System.IO;
00007 using System.Reflection;
00008
00015 [CustomEditor(typeof(Readme))]
00016 [InitializeOnLoad]
```

```
00017 public class ReadmeEditor : Editor
00018 {
00019     static string s_ShowedReadmeSessionStateName = "ReadmeEditor.showedReadme";
00020
00021     static string s_ReadmeSourceDirectory = "Assets/TutorialInfo";
00022
00023     const float k_Space = 16f;
00024
00030     static ReadmeEditor()
00031     {
00032         EditorApplication.delayCall += SelectReadmeAutomatically;
00033     }
00034
00040     static void RemoveTutorial()
00041     {
00042         if (EditorUtility.DisplayDialog("Remove Readme Assets",
00043
00044             $"All contents under {s_ReadmeSourceDirectory} will be removed, are you sure you want to proceed?",
00045             "Proceed",
00046             "Cancel"))
00047         {
00048             if (Directory.Exists(s_ReadmeSourceDirectory))
00049             {
00050                 FileUtil.DeleteFileOrDirectory(s_ReadmeSourceDirectory);
00051                 FileUtil.DeleteFileOrDirectory(s_ReadmeSourceDirectory + ".meta");
00052             }
00053             else
00054             {
00055                 Debug.Log($"Could not find the Readme folder at {s_ReadmeSourceDirectory}");
00056             }
00057
00058             var readmeAsset = SelectReadme();
00059             if (readmeAsset != null)
00060             {
00061                 var path = AssetDatabase.GetAssetPath(readmeAsset);
00062                 FileUtil.DeleteFileOrDirectory(path + ".meta");
00063                 FileUtil.DeleteFileOrDirectory(path);
00064             }
00065
00066             AssetDatabase.Refresh();
00067         }
00068     }
00069
00076     static void SelectReadmeAutomatically()
00077     {
00078         if (!SessionState.GetBool(s_ShowedReadmeSessionStateName, false))
00079         {
00080             var readme = SelectReadme();
00081             SessionState.SetBool(s_ShowedReadmeSessionStateName, true);
00082
00083             if (readme && !readme.loadedLayout)
00084             {
00085                 LoadLayout();
00086                 readme.loadedLayout = true;
00087             }
00088         }
00089     }
00090
00096     static void LoadLayout()
00097     {
00098         var assembly = typeof(EditorApplication).Assembly;
00099         var windowLayoutType = assembly.GetType("UnityEditor.WindowLayout", true);
00100         var method = windowLayoutType.GetMethod("LoadWindowLayout", BindingFlags.Public | BindingFlags.Static);
00101         method.Invoke(null, new object[] { Path.Combine(Application.dataPath, "TutorialInfo/Layout.wlt"), false });
00102     }
00103
00109     static Readme SelectReadme()
00110     {
00111         var ids = AssetDatabase.FindAssets("Readme t:Readme");
00112         if (ids.Length == 1)
00113         {
00114             var readmeObject = AssetDatabase.LoadMainAssetAtPath(AssetDatabase.GUIDToAssetPath(ids[0]));
00115
00116             Selection.objects = new UnityEngine.Object[] { readmeObject };
00117
00118             return (Readme)readmeObject;
00119         }
00120         else
00121         {
00122             Debug.Log("Couldn't find a readme");
00123             return null;
00124         }
00125     }
00126
00132     protected override void OnHeaderGUI()
00133     {
00134         var readme = (Readme)target;
```

```
00135         Init();
00136
00137         var iconWidth = Mathf.Min(EditorGUIUtility.currentViewWidth / 3f - 20f, 128f);
00138
00139         GUILayout.BeginHorizontal("In BigTitle");
00140         {
00141             if (readme.icon != null)
00142             {
00143                 GUILayout.Space(k_Space);
00144                 GUILayout.Label(readme.icon, GUILayout.Width(iconWidth), GUILayout.Height(iconWidth));
00145             }
00146             GUILayout.Space(k_Space);
00147             GUILayout.BeginVertical();
00148             {
00149
00150                 GUILayout.FlexibleSpace();
00151                 GUILayout.Label(readme.title, TitleStyle);
00152                 GUILayout.FlexibleSpace();
00153             }
00154             GUILayout.EndVertical();
00155             GUILayout.FlexibleSpace();
00156         }
00157         GUILayout.EndHorizontal();
00158     }
00159
00165     public override void OnInspectorGUI()
00166     {
00167         var readme = (Readme)target;
00168         Init();
00169
00170         foreach (var section in readme.sections)
00171         {
00172             if (!string.IsNullOrEmpty(section.heading))
00173             {
00174                 GUILayout.Label(section.heading, HeadingStyle);
00175             }
00176
00177             if (!string.IsNullOrEmpty(section.text))
00178             {
00179                 GUILayout.Label(section.text, BodyStyle);
00180             }
00181
00182             if (!string.IsNullOrEmpty(section.linkText))
00183             {
00184                 if (LinkLabel(new GUIContent(section.linkText)))
00185                 {
00186                     Application.OpenURL(section.url);
00187                 }
00188             }
00189
00190             GUILayout.Space(k_Space);
00191         }
00192
00193         if (GUILayout.Button("Remove Readme Assets", ButtonStyle))
00194         {
00195             RemoveTutorial();
00196         }
00197     }
00198
00199     bool m_Initialized;
00200
00201     GUIStyle LinkStyle
00202     {
00203         get { return m_LinkStyle; }
00204     }
00205
00206     [SerializeField]
00207     GUIStyle m_LinkStyle;
00208
00209     GUIStyle TitleStyle
00210     {
00211         get { return m_TitleStyle; }
00212     }
00213
00214     [SerializeField]
00215     GUIStyle m_TitleStyle;
00216
00217     GUIStyle HeadingStyle
00218     {
00219         get { return m_HeadingStyle; }
00220     }
00221
00222     [SerializeField]
00223     GUIStyle m_HeadingStyle;
00224
00225     GUIStyle BodyStyle
00226     {
```

```
00227         get { return m_BodyStyle; }
00228     }
00229
00230     [SerializeField]
00231     GUIStyle m_BodyStyle;
00232
00233     GUIStyle ButtonStyle
00234     {
00235         get { return m_ButtonStyle; }
00236     }
00237
00238     [SerializeField]
00239     GUIStyle m_ButtonStyle;
00240
00246     void Init()
00247     {
00248         if (m_Initialized)
00249             return;
00250         m_BodyStyle = new GUIStyle(EditorStyles.label);
00251         m_BodyStyle.wordWrap = true;
00252         m_BodyStyle.fontSize = 14;
00253         m_BodyStyle.richText = true;
00254
00255         m_TitleStyle = new GUIStyle(m_BodyStyle);
00256         m_TitleStyle.fontSize = 26;
00257
00258         m_HeadingStyle = new GUIStyle(m_BodyStyle);
00259         m_HeadingStyle.fontStyle = FontStyle.Bold;
00260         m_HeadingStyle.fontSize = 18;
00261
00262         m_LinkStyle = new GUIStyle(m_BodyStyle);
00263         m_LinkStyle.wordWrap = false;
00264
00265         // Match selection color which works nicely for both light and dark skins
00266         m_LinkStyle.normal.textColor = new Color(0x00 / 255f, 0x78 / 255f, 0xDA / 255f, 1f);
00267         m_LinkStyle.stretchWidth = false;
00268
00269         m_ButtonStyle = new GUIStyle(EditorStyles.miniButton);
00270         m_ButtonStyle.fontStyle = FontStyle.Bold;
00271
00272         m_Initialized = true;
00273     }
00274
00282     bool LinkLabel(GUIContent label, params GUILayoutOption[] options)
00283     {
00284         var position = GUILayoutUtility.GetRect(label, LinkStyle, options);
00285
00286         Handles.BeginGUI();
00287         Handles.color = LinkStyle.normal.textColor;
00288         Handles.DrawLine(new Vector3(position.xMin, position.yMax), new Vector3(position.xMax, position.yMax));
00289         Handles.color = Color.white;
00290         Handles.EndGUI();
00291
00292         EditorGUIUtility.AddCursorRect(position, MouseCursor.Link);
00293
00294         return GUI.Button(position, label, LinkStyle);
00295     }
00296 }
```

5.7 Readme.cs

```
00001 using System;
00002 using UnityEngine;
00003
00010 public class Readme : ScriptableObject
00011 {
00012     public Texture2D icon;     // The icon to display with the readme
00013     public string title;       // The title of the readme
00014     public Section[] sections; // The content sections of the readme
00015     public bool loadedLayout;  // Whether the editor layout has been loaded for this readme
00016
00022     [Serializable]
00023     public class Section
00024     {
00025         public string heading;  // Section heading
00026         public string text;     // Main content text
00027         public string linkText; // Text for the hyperlink
00028         public string url;      // URL for the hyperlink
00029     }
00030 }
```

Предметный указатель