TEXTURE WEB VIEW

User Manual

Version 1.0.0

Contents

USER SUPPORT

If you need support or have any questions or suggestions, please contact with me:

- easywayasset@gmail.com

INTRODUCTION

Texture Web View (TWV) is a Android web view plugin for Unity that gives ability to render websites on “Texture” component with possibility to use interact functionality that based on native web component.

Possibilities of current version:

- Fast and flexible web view rendering that uses native hardware canvas: support only **OpenGLES2/3** graphics API);
- Supported armeabi-v7a, x86, arm64-v8a-bit Android processors (for arm64 you need enable libraries directly in Unity Engine);
- Supported possibility to render web pages on 2D or 3D objects;
- Supported possibility to render components that need a hardware rendering support (only for API level 23+);
- Supported possibility to asynchronously evaluates JavaScript in the context of the currently displayed page;
- Supported possibility to render multiple web views instances;
- Supported possibility to get pixels of current web page frame (but it's will be ignore using hardware features and can work not so smooth);
- Supports main interaction features, like: click, scroll, text selection, edit;
- Experimental Virtual Reality Support (Cardboard, Oculus).

IMPORTANT

- Doesn't work in Unity Editor mode;
- Requires minimum API level 15 (Android 4.0.3) for hardware rendering API level 23+;
- For correct work switch in Player Settings - Internet Access: Require.

INSTALLATION GUIDELINE

Import the TWV package from the Asset Store. You should now have a folders named "StreamingAssets" and "TextureWebView" in your project with the following structure:

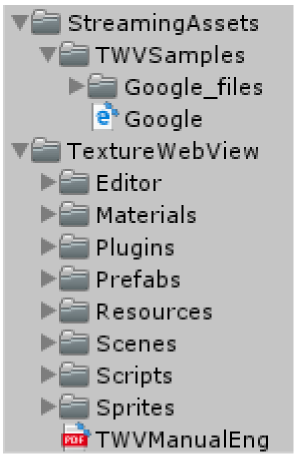

```
▼ StreamingAssets
    ▼ TWVSamples
        ▶ Google_files
          Google
▼ TextureWebView
    ▶ Editor
    ▶ Materials
    ▶ Plugins
    ▶ Prefabs
    ▶ Resources
    ▶ Scenes
    ▶ Scripts
    ▶ Sprites
      TWVManualEng
```

Figure 3.1: Structure of TWV asset

- **TWVSamples**: Include samples that will be used in "StreamingAssetsExample" scene;
- **Editor**: Custom Editor script for TWV instances;
- **Materials**: Materials that uses for correctly render web view output texture on Unity 3D objects;
- **Plugins**: All the native dlls and shared libraries (in this folder you can directly enable/disable of using libraries for each Android processors);
- **Prefabs**: Special web view prefab that can be used to easily setup your project with TWV asset;
- **Resources**: Include asset default icon;
- **Scenes**: Example scenes that's show all components in a ready setup and how work with some additional features of TWV asset;
- **Scripts**: C# classes that show how to work with TWV;
- **Sprites**: Additional images that used in demo scenes.

USAGE GUIDELINE

You need to find the special TWV prefab in "Prefabs" folder that give you possibility to manage all web view functionality and move it to your Unity scene:

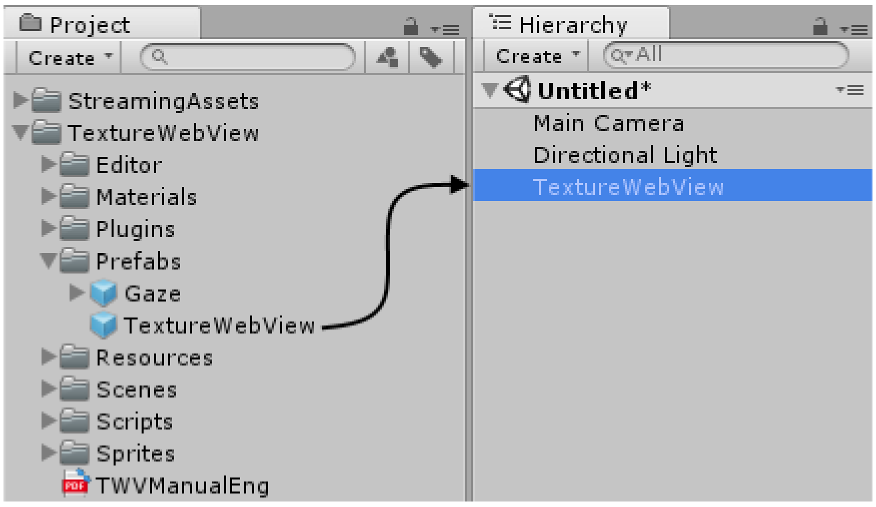
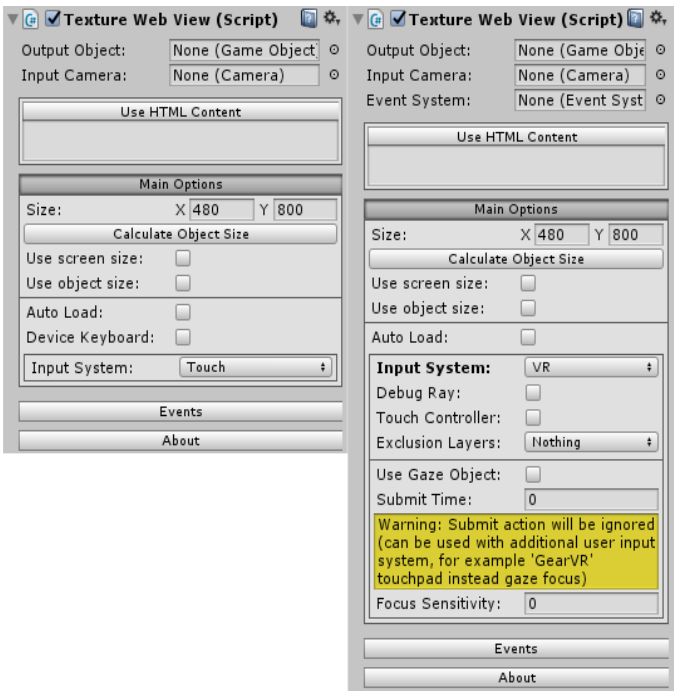


Figure 4.1: Add new TWV instance

When you select the "TextureWebView" component in your hierarchy window you can manage it with the component inspector:



Texture Web View (Script)

Output Object: None (Game Object)
Input Camera: None (Camera)

Use HTML Content

Main Options

Size: X 480 Y 800
Calculate Object Size
Use screen size:
Use object size:
Auto Load:
Device Keyboard:
Input System: Touch

Events
About

Texture Web View (Script)

Output Object: None (Game Obje
Input Camera: None (Camera)
Event System: None (Event Syst

Use HTML Content

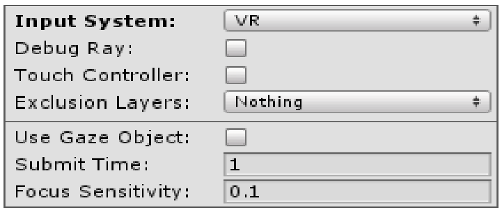
Main Options

Size: X 480 Y 800
Calculate Object Size
Use screen size:
Use object size:
Auto Load:
Input System: VR
Debug Ray:
Touch Controller:
Exclusion Layers: Nothing
Use Gaze Object:
Submit Time: 0
Warning: Submit action will be ignored (can be used with additional user input system, for example 'GearVR' touchpad instead gaze focus)
Focus Sensitivity: 0

Events
About

Figure 4.2: TWV inspector view

- **Output Object**: Simple “GameObject” that have "Mesh Renderer" with some material or "Raw Image" component. If you will be use 3D object with “Mesh Renderer” component and you want add possibility to interact with web page on this object, please add “**Mesh Collider**” instead using default “Box Collider”;
- **Input Camera**: Main scene camera component that used for web view input system;
- **Event System**: Default Unity “EventSystem” component, will be ignored if you will be use 3D object for **Output Object**;
- **Page URL/HTML Content**: Page url or html content that will be used to render on your **Output Object** component;
- **Size**: Use fixed view size in pixels. Also, you have possibility to "Calculate Object Size", that will try to automatically calculate output object size in pixels in “Unity Editor” mode;
- **Use Screen Size**: Ignore **Size** property and use device screen size instead;
- **Use Object Size**: Ignore **Size** property and try to automatically calculate output object size in pixels in runtime mode;
- **Auto Load**: Auto load current web view component if it's enable;
- **Device Keyboard**: Show default device keyboard that allows a user to enter data into current active web page input field;
- **Input System**: Input system that will be used to handle user actions. If you will be use “VR” input system that you will have some additional options that will be useful only in VR:

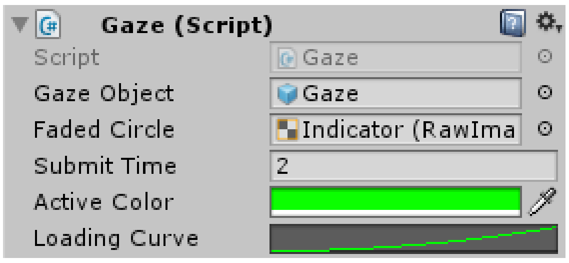


Input System:	VR
Debug Ray:	☐
Touch Controller:	☐
Exclusion Layers:	Nothing
Use Gaze Object:	☐
Submit Time:	1
Focus Sensitivity:	0.1

Figure 4.3: VR input system options

- **Debug Ray**: Show debug ray (only in 'Editor' mode);

- **Touch Controller**: Use additional touch controller (useful for Oculus headset);
- **Exclusion Layers**: Layers to exclude from the raycast, this layers will be ignored from user input system;
- **Use Gaze Object**: Use special gaze object in VR environment. Gaze object it's a special component (script) that included in TWV asset in "Scripts" folder (also you can use gaze prefab in "Prefabs" folder) and can be used for VR gaze input system:


Gaze (Script)

Script	Gaze
Gaze Object	Gaze
Faded Circle	Indicator (RawIma
Submit Time	2
Active Color	🟩
Loading Curve	

Figure 4.4: Additional "Gaze" script

- **Submit Time**: Time in seconds, to wait for admit actions;
- **Focus Sensitivity**: Sensitivity to disable current action when change gaze position;
- **Events**: Gives possibility to add listeners for special TWV callbacks:
  - **PageStarted** - Notify the host application that a page has started loading;
  - **PageLoading** - Tell the host application the current progress of loading a page;
  - **PageFinished** - Notify the host application that a page has finished loading;
  - **PageImageReady** - Notify the host application that a page image has ready to use;
  - **PageConsoleMessage** - Report a JavaScript console message to the host application
  - **PageError** - Report w;eb resource loading error to the host application;
  - **PageHttpError** - Notify the host application that an HTTP error has been received from the server while loading a resource. HTTP errors have status codes >= 400. This callback will be called for any resource (iframe, image, etc.), not just for the main page;
  - **MotionEvents** - Motion events callback for input system;
  - **Error** - Report web view loading error.

CLASSES/METHODS DESCRIPTION

Classes

- `WebView` – main class that has all web view functions;
- `WebViewHelper` – add possibility to make additional stuff with web view.

Interfaces

- **`IPageListener`**:

```
 IPageStartedListener,
 IPageLoadingListener,
 IPageFinishedListener,
 IPageImageReadyListener,
 IPageConsoleMessageListener,
 IPageErrorListener,
 IPageHttpErrorListener
{ }
```

- `IPageStartedListener`

```
{
    void OnPageStarted(string url);
}
```

- `IPageLoadingListener`

```
{
    void OnPageLoading(int progress);
}
```

- `IPageFinishedListener`

```
{
    void OnPageFinished(string url);
}
```

- `IPageImageReadyListener`

```
{
    void OnPageImageReady(Texture2D viewTexture);
}
```

- IPageConsoleMessageListener

```
{
    void OnPageConsoleMessage(string consoleMessage);
}
```

- IPageErrorListener

```
{
    void OnPageError(string message);
}
```

- IPageHttpErrorListener

```
{
    void OnPageHttpError(string message);
}
```

- IJsDataReceivedListener

```
{
    void OnJsDataReceived(string value);
}
```

- IErrorListener

```
{
    void OnError(string message);
}
```

- IInputManagerListener

```
{
    void OnMotionEvents(GameObject target, MotionActions action, Vector2 coord);
}
```

WebView

Constructors

```
/// <summary>
///  Create new instance of WebView object for current supported platform with device
screen size by default
/// </summary>
/// <param name="monoObject">MonoBehaviour instanse</param>
/// <param name="outputObject">WebView rendering object</param>
public WebView(MonoBehaviour monoObject, GameObject outputObject)
```

```
/// <summary>
///  Create new instance of WebView object for current supported platform
/// </summary>
/// <param name="monoObject">MonoBehaviour instanse</param>
/// <param name="outputObject">WebView rendering object</param>
/// <param name="size">WebView size in pixels (width and height should be bigger than
'0', instead will be use device screen size by default)</param>
public WebView(MonoBehaviour monoObject, GameObject outputObject, Vector2 size)
```

Properties

```
/// <summary>
/// Get web view object for current running platform
/// for get more additional possibilities that exists only for this platform.
/// </summary>
public object PlatformWebView
```

```
/// <summary>
/// Get/Set Unity 'GameObject'
/// that have 'Mesh Renderer' (with some material)
/// or 'Raw Image' component
/// </summary>
public GameObject OutputObject
```

```
/// <summary>
/// Get event manager for current web view
/// to add possibility to attach/detach special playback listeners
/// </summary>
public WebViewManagerEvents EventManager
```

```
/// <summary>
/// Get current web view state
/// </summary>
public WebStates State
```

```
/// <summary>
/// Get current web view state value (can be float, long or string type)
/// </summary>
public object StateValue
```

```
/// <summary>
/// Gets the URL for the current page. This is not always the same as the URL passed
to Load,
/// because the current page can be changed.
/// </summary>
public string PageUrl
```

```
/// <summary>
/// Is web page ready to use and output texture is available
/// </summary>
public bool IsReady
```

```
/// <summary>
/// Show default device keyboard that allows a user
/// to enter data into current active web page input field

/// </summary>
public bool DeviceKeyboard
```

```
/// <summary>
/// Get the width of your view, in pixels.
/// </summary>
public int Width
```

```
/// <summary>
/// Get the height of your view, in pixels.
/// </summary>
public int Height
```

Methods

```
/// <summary>
/// Add new main group of listeners to current web view instance
/// </summary>
/// <param name="listener">Group of listeners</param>
public void AddWebListener(IWebListener listener)
```

```
/// <summary>
/// Remove group of listeners from current web view instance
/// </summary>
/// <param name="listener">Group of listeners</param>
public void RemoveWebListener(IWebListener listener)
```

```
/// <summary>
/// Load new web page based on current url
/// </summary>
/// <param name="url">Page url</param>
public void Load(Uri url)
```

```
/// <summary>
/// Load new web page based on current data
/// </summary>
/// <param name="data">HTML data</param>
public void Load(string data)
```

```
/// <summary>
/// Unload current web page
/// </summary>
/// <param name="resetTexture">Clear the last frame</param>
public void UnLoad(bool resetTexture)
```

```
/// <summary>
/// Unload current web page and clear last frame
/// </summary>
public void UnLoad()
```

```
/// <summary>
/// Release a current web view instance
/// </summary>
public void Release()
```

WebViewHelper

```
/// <summary>
/// Apply texture to Unity game object that has 'RawImage' or 'MeshRenderer' component
/// </summary>
/// <param name="texture">Texture to render video output</param>
/// <param name="renderingObject">Game object where will be rendering video
output</param>
/// <returns></returns>
public static void ApplyTextureToRenderingObject(Texture2D texture, GameObject
renderingObject)
```

```
/// <summary>
/// Get size in pixels of gameobject that has "MeshRenderer" or "RawImage" component
/// </summary>
/// <param name="meshRenderer">Gameobject that has "MeshRenderer" or "RawImage"
component</param>
/// <param name="camera">Main camera of current scene</param>
/// <returns></returns>
public static Vector2 GetPixelSizeOfObject(GameObject obj, Camera camera)
```

```
/// <summary>
/// Get size in pixels of gameobject that have "MeshRenderer" component
/// </summary>
/// <param name="meshRenderer">Gameobject that has "MeshRenderer" component</param>
/// <param name="camera">Main camera of current scene</param>
/// <returns></returns>
public static Vector2 GetPixelSizeOfMeshRenderer(MeshRenderer meshRenderer, Camera
camera)
```

```
/// <summary>
/// Get size in pixels of gameobject that have "RawImage" component
/// </summary>
/// <param name="rawImage">Gameobject that has "RawImage" component</param>
/// <returns></returns>
public static Vector2 GetPixelSizeOfRawImage(RawImage rawImage)
```

```
/// <summary>
/// Getting average color from frame buffer array
/// </summary>
/// <returns></returns>
public static Color GetAverageColor(byte[] frameBuffer)
```

```
/// <summary>
/// Getting colors from frame buffer array
/// </summary>
/// <returns></returns>
public static Color32[] GetFrameColors(byte[] frameBuffer)
```

```
/// <summary>
/// Getting root storage menory for current platform
/// Windows, Mac OS, Linux will return path to project folder
/// Android, iOS will return root of internal/external memory root
/// </summary>
/// <returns></returns>
public static string GetDeviceRootPath()
```

```
/// <summary>
/// Check if the runtime platforms is supported by MWV asset
/// </summary>
/// <returns></returns>
public static bool IsSupportedPlatform
```