LAMPIRAN

LISTING PROGRAM

VIRTUAL BUTTON VIDEO PLAYER

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class VBtn : MonoBehaviour
{

    // public
UnityEngine.Video.VideoSource
videoSource;
    public
UnityEngine.Video.VideoPlayer
videoPlayer1;
    public
UnityEngine.Video.VideoPlayer
videoPlayer2;
    public
UnityEngine.Video.VideoPlayer
videoPlayer3;
    public
UnityEngine.Video.VideoPlayer
videoPlayer4;
    public
UnityEngine.Video.VideoPlayer
videoPlayer5;
    public
UnityEngine.Video.VideoPlayer
videoPlayer6;
    public
UnityEngine.Video.VideoPlayer
videoPlayer7;
    public
UnityEngine.Video.VideoPlayer
videoPlayer8;
    public
UnityEngine.Video.VideoPlayer
videoPlayer9;
    public
UnityEngine.Video.VideoPlayer
videoPlayer10;
    public
UnityEngine.Video.VideoPlayer
videoPlayer11;
    public
UnityEngine.Video.VideoPlayer
videoPlayer12;
    public GameObject rawImage;

    public VirtualButtonBehaviour[]
virtualButtonBehaviours;
    // Start is called before the first
frame update
    void Start()
    {
        rawImage.SetActive(false);

        // Register with the virtual
buttons TrackableBehaviour
        virtualButtonBehaviours =
GetComponentsInChildren<VirtualB
uttonBehaviour>();

        for (int i = 0; i <
virtualButtonBehaviours.Length;
++i)
        {

virtualButtonBehaviours[i].Register
OnButtonPressed(OnButtonPressed);

virtualButtonBehaviours[i].Register
OnButtonReleased(OnButtonRelease
d);
        }
    }

    void Destroy()
    {
        // Register with the virtual
buttons TrackableBehaviour
        virtualButtonBehaviours =
GetComponentsInChildren<VirtualB
uttonBehaviour>();
```

```
        for (int i = 0; i <
virtualButtonBehaviours.Length;
++i)
        {

virtualButtonBehaviours[i].Unregiste
rOnButtonPressed(OnButtonPressed)
;

virtualButtonBehaviours[i].Unregiste
rOnButtonReleased(OnButtonReleas
ed);
        }
    }

    public void
OnButtonPressed(VirtualButtonBeha
viour vb)
    {
        rawImage.SetActive(true);
        if (vb.VirtualButtonName ==
"VBtn1")
        {
            videoPlayer1.Play();
            videoPlayer2.Stop();
            videoPlayer3.Stop();
            videoPlayer4.Stop();
            videoPlayer5.Stop();
            videoPlayer6.Stop();
            videoPlayer7.Stop();
            videoPlayer8.Stop();
            videoPlayer9.Stop();
            videoPlayer10.Stop();
            videoPlayer11.Stop();
            videoPlayer12.Stop();
        }
        else if (vb.VirtualButtonName
== "VBtn2")
        {
            videoPlayer2.Play();
            videoPlayer1.Stop();
            videoPlayer3.Stop();
            videoPlayer4.Stop();
            videoPlayer5.Stop();
            videoPlayer6.Stop();
            videoPlayer7.Stop();
            videoPlayer8.Stop();
            videoPlayer9.Stop();
            videoPlayer10.Stop();
            videoPlayer11.Stop();
            videoPlayer12.Stop();
        }
        else if (vb.VirtualButtonName
== "VBtn3")
        {
            videoPlayer3.Play();
            videoPlayer1.Stop();
            videoPlayer2.Stop();
            videoPlayer4.Stop();
            videoPlayer5.Stop();
            videoPlayer6.Stop();
            videoPlayer7.Stop();
            videoPlayer8.Stop();
            videoPlayer9.Stop();
            videoPlayer10.Stop();
            videoPlayer11.Stop();
            videoPlayer12.Stop();
        }
        else if (vb.VirtualButtonName
== "VBtn4")
        {
            videoPlayer4.Play();
            videoPlayer1.Stop();
            videoPlayer2.Stop();
            videoPlayer3.Stop();
            videoPlayer5.Stop();
            videoPlayer6.Stop();
            videoPlayer7.Stop();
            videoPlayer8.Stop();
            videoPlayer9.Stop();
            videoPlayer10.Stop();
            videoPlayer11.Stop();
            videoPlayer12.Stop();
        }
        else if (vb.VirtualButtonName
== "VBtn5")
        {
            videoPlayer5.Play();
            videoPlayer1.Stop();
            videoPlayer2.Stop();
            videoPlayer4.Stop();
            videoPlayer3.Stop();
```

```
            videoPlayer6.Stop();
            videoPlayer7.Stop();
            videoPlayer8.Stop();
            videoPlayer9.Stop();
            videoPlayer10.Stop();
            videoPlayer11.Stop();
            videoPlayer12.Stop();
        }

        else if (vb.VirtualButtonName
== "VBtn6")
        {
            videoPlayer6.Play();
            videoPlayer1.Stop();
            videoPlayer2.Stop();
            videoPlayer4.Stop();
            videoPlayer3.Stop();
            videoPlayer5.Stop();
            videoPlayer7.Stop();
            videoPlayer8.Stop();
            videoPlayer9.Stop();
            videoPlayer10.Stop();
            videoPlayer11.Stop();
            videoPlayer12.Stop();
        }
        else if (vb.VirtualButtonName
== "VBtn7")
        {
            videoPlayer7.Play();
            videoPlayer1.Stop();
            videoPlayer2.Stop();
            videoPlayer4.Stop();
            videoPlayer3.Stop();
            videoPlayer5.Stop();
            videoPlayer6.Stop();
            videoPlayer8.Stop();
            videoPlayer9.Stop();
            videoPlayer10.Stop();
            videoPlayer11.Stop();
            videoPlayer12.Stop();
        }
        else if (vb.VirtualButtonName
== "VBtn8")
        {
            videoPlayer8.Play();
            videoPlayer1.Stop();
            videoPlayer2.Stop();
            videoPlayer4.Stop();
            videoPlayer3.Stop();
            videoPlayer5.Stop();
            videoPlayer6.Stop();
            videoPlayer7.Stop();
            videoPlayer9.Stop();
            videoPlayer10.Stop();
            videoPlayer11.Stop();
            videoPlayer12.Stop();
        }
        else if (vb.VirtualButtonName
== "VBtn9")
        {
            videoPlayer9.Play();
            videoPlayer1.Stop();
            videoPlayer2.Stop();
            videoPlayer4.Stop();
            videoPlayer3.Stop();
            videoPlayer5.Stop();
            videoPlayer6.Stop();
            videoPlayer7.Stop();
            videoPlayer8.Stop();
            videoPlayer10.Stop();
            videoPlayer11.Stop();
            videoPlayer12.Stop();
        }
        else if (vb.VirtualButtonName
== "VBtn10")
        {
            videoPlayer10.Play();
            videoPlayer1.Stop();
            videoPlayer2.Stop();
            videoPlayer4.Stop();
            videoPlayer3.Stop();
            videoPlayer5.Stop();
            videoPlayer6.Stop();
            videoPlayer7.Stop();
            videoPlayer8.Stop();
            videoPlayer9.Stop();
            videoPlayer11.Stop();
            videoPlayer12.Stop();
        }
        else if (vb.VirtualButtonName
== "VBtn11")
        {
```

```
            videoPlayer11.Play();
            videoPlayer1.Stop();
            videoPlayer2.Stop();
            videoPlayer4.Stop();
            videoPlayer3.Stop();
            videoPlayer5.Stop();
            videoPlayer6.Stop();
            videoPlayer7.Stop();
            videoPlayer8.Stop();
            videoPlayer9.Stop();
            videoPlayer10.Stop();
            videoPlayer12.Stop();
        }
        else if (vb.VirtualButtonName
== "VBtn12")
        {
            videoPlayer12.Play();
            videoPlayer1.Stop();
            videoPlayer2.Stop();
            videoPlayer4.Stop();
            videoPlayer3.Stop();
            videoPlayer5.Stop();
            videoPlayer6.Stop();
            videoPlayer7.Stop();
            videoPlayer8.Stop();
            videoPlayer9.Stop();
            videoPlayer10.Stop();
            videoPlayer11.Stop();
        }
    }

    // Update is called once per frame
    public void
OnButtonReleased(VirtualButtonBeh
aviour vb)
    {
        rawImage.SetActive(false);

        videoPlayer1.Stop();
        videoPlayer2.Stop();
        videoPlayer3.Stop();
        videoPlayer4.Stop();
        videoPlayer5.Stop();
        videoPlayer6.Stop();
        videoPlayer7.Stop();
        videoPlayer8.Stop();
        videoPlayer9.Stop();
        videoPlayer10.Stop();
        videoPlayer11.Stop();
        videoPlayer12.Stop();
    }
}
```

URLOPENER

```
using UnityEngine;

public class UrlOpener :
MonoBehaviour
{
    public string Url;
    // Start is called before the first
frame update

    public void Open()
    {
        Application.OpenURL(Url);
    }

}
```

VUFORIA ENGINE PACKAGE

```
using System;
using System.IO;
using System.Linq;
using System.Text;
using UnityEditor;
using UnityEngine;

[InitializeOnLoad]
public class
AddVuforiaEnginePackage
{
    static readonly string
sPackagesPath =
Path.Combine(Application.dataPath,
"..", "Packages");
    static readonly string
sManifestJsonPath =
Path.Combine(sPackagesPath,
"manifest.json");
    const string
VUFORIA_VERSION = "10.0.12";
```

```
    const string PACKAGE_KEY =
"com.ptc.vuforia.engine";
    const string GIT_URL =
"git+https://git-
packages.developer.vuforia.com";
    const string
VUFORIA_TAR_FILE_DIR =
"Assets/Editor/Migration/";
    const string
PACKAGES_RELATIVE_PATH =
"Packages";

    static readonly ScopedRegistry
sVuforiaRegistry = new
ScopedRegistry()
    {
        name = "Vuforia",
        url =
"https://registry.packages.developer.
vuforia.com/",
        scopes = new[] {
"com.ptc.vuforia" }
    };

    static AddVuforiaEnginePackage()
    {
        if (Application.isBatchMode)
            return;

        var manifest =
Manifest.JsonDeserialize(sManifestJ
sonPath);

        if
(!IsVuforiaUpToDate(manifest))

DisplayAddPackageDialogue(manife
st);
    }

    static bool
IsVuforiaUpToDate(Manifest
manifest)
    {
        var dependencies =
manifest.Dependencies.Split(',').ToLi
st();
        bool upToDate = false;
        // is git dependency
        if (dependencies.Any(d =>
d.Contains(PACKAGE_KEY) &&
d.Contains(GIT_URL)))
            upToDate =
IsUsingRightGitUrl(manifest);
        // tarball path dependency
        if (dependencies.Any(d =>
d.Contains(PACKAGE_KEY) &&
d.Contains("file:")))
            upToDate =
IsUsingRightTarballPath(manifest);

        return upToDate;
    }

    static bool
IsUsingRightGitUrl(Manifest
manifest)
    {
        var dependencies =
manifest.Dependencies.Split(',').ToLi
st();
        return dependencies.Any(d =>
d.Contains(PACKAGE_KEY) &&
d.Contains(GIT_URL) &&
VersionNumberIsTheLatestGit(d));
    }

    static bool
IsUsingRightTarballPath(Manifest
manifest)
    {
        var dependencies =
manifest.Dependencies.Split(',').ToLi
st();
        return dependencies.Any(d =>
d.Contains(PACKAGE_KEY) &&
d.Contains("file:") &&
VersionNumberIsTheLatestTarball(d
));
    }
```

```
    static bool
VersionNumberIsTheLatestGit(string
package)
    {
        var version = package.Split('#');
        if (version.Length >= 2)
        {
            version[1] =
version[1].TrimEnd(new[] { "" });
            return
IsCurrentVersionHigher(version[1]);
        }

        return false;
    }

    static bool
VersionNumberIsTheLatestTarball(st
ring package)
    {
        var version = package.Split('-');
        if (version.Length >= 2)
        {
            version[1] =
version[1].TrimEnd(".tgz\"".ToChar
Array());
            return
IsCurrentVersionHigher(version[1]);
        }

        return false;
    }

    static bool
IsCurrentVersionHigher(string
currentVersionString)
    {
        if
(string.IsNullOrEmpty(currentVersio
nString) ||
string.IsNullOrEmpty(VUFORIA_V
ERSION))
            return false;

        var currentVersion =
TryConvertStringToVersion(current
VersionString);
        var updatingVersion =
TryConvertStringToVersion(VUFO
RIA_VERSION);
        if (currentVersion >=
updatingVersion)
            return true;

        return false;
    }

    static Version
TryConvertStringToVersion(string
versionString)
    {
        Version res;
        try
        {
            res = new
Version(versionString);
        }
        catch (Exception e)
        {
            return new Version();
        }

        return new Version(res.Major,
res.Minor, res.Build);
    }


    static void
DisplayAddPackageDialogue(Manife
st manifest)
    {
        if
(EditorUtility.DisplayDialog("Add
Vuforia Engine Package",
            $"Would you like to update
your project to include the Vuforia
Engine {VUFORIA_VERSION}
package from the unitypackage?\n" +
            $"If an older Vuforia Engine
package is already present in your
```

```
project it will be upgraded to version
{VUFORIA_VERSION}\n\n",
            "Update", "Cancel"))
        {
            CopyTarball();
            UpdateManifest(manifest);
        }
    }

    static void CopyTarball()
    {
        var sourceFile =
Path.Combine(Directory.GetCurrent
Directory(),
VUFORIA_TAR_FILE_DIR,
$"{PACKAGE_KEY}-
{VUFORIA_VERSION}.tgz");
        var destFile =
Path.Combine(Directory.GetCurrent
Directory(),
PACKAGES_RELATIVE_PATH,
$"{PACKAGE_KEY}-
{VUFORIA_VERSION}.tgz");
        File.Copy(sourceFile, destFile,
true);
        File.Delete(sourceFile);
    }

    static void
UpdateManifest(Manifest manifest)
    {
        //remove existing, outdated
NPM scoped registry if present
        var registries =
manifest.ScopedRegistries.ToList();
        if
(registries.Contains(sVuforiaRegistry
))
        {

registries.Remove(sVuforiaRegistry)
;
            manifest.ScopedRegistries =
registries.ToArray();
        }

        //add specified vuforia version
via Git URL
        SetVuforiaVersion(manifest);


manifest.JsonSerialize(sManifestJson
Path);

        AssetDatabase.Refresh();
    }

    static void
SetVuforiaVersion(Manifest
manifest)
    {
        var dependencies =
manifest.Dependencies.Split(',').ToLi
st();

        var versionEntry =
$"\"file:{PACKAGE_KEY}-
{VUFORIA_VERSION}.tgz\"";
        var versionSet = false;
        for (var i = 0; i <
dependencies.Count; i++)
        {
            if
(!dependencies[i].Contains(PACKA
GE_KEY))
                continue;

            var kvp =
dependencies[i].Split(':');
            dependencies[i] = kvp[0] + ":
" + versionEntry;
            versionSet = true;
        }

        if (!versionSet)
            dependencies.Insert(0, $"\n
\"{PACKAGE_KEY}\":
{versionEntry}");

        manifest.Dependencies =
string.Join(",", dependencies);
    }
```

```
    class Manifest
    {
        const int
INDEX_NOT_FOUND = -1;
        const string
DEPENDENCIES_KEY =
"\"dependencies\"";

        public ScopedRegistry[]
ScopedRegistries;
        public string Dependencies;

        public void JsonSerialize(string
path)
        {
            var jsonString =
GetJsonString();

            var startIndex =
GetDependenciesStart(jsonString);
            var endIndex =
GetDependenciesEnd(jsonString,
startIndex);

            var stringBuilder = new
StringBuilder();


stringBuilder.Append(jsonString.Sub
string(0, startIndex));

stringBuilder.Append(Dependencies)
;

stringBuilder.Append(jsonString.Sub
string(endIndex, jsonString.Length -
endIndex));

            File.WriteAllText(path,
stringBuilder.ToString());
        }

        string GetJsonString()
        {
            if (ScopedRegistries.Length >
0)
                return JsonUtility.ToJson(
                    new
UnitySerializableManifest {
scopedRegistries = ScopedRegistries,
dependencies = new
DependencyPlaceholder() },
                    true);

            return JsonUtility.ToJson(
                new
UnitySerializableManifestDependenc
iesOnly() { dependencies = new
DependencyPlaceholder() },
                true);
        }


        public static Manifest
JsonDeserialize(string path)
        {
            var jsonString =
File.ReadAllText(path);

            var registries =
JsonUtility.FromJson<UnitySerializa
bleManifest>(jsonString).scopedRegi
stries ?? new ScopedRegistry[0];
            var dependencies =
DeserializeDependencies(jsonString)
;

            return new Manifest {
ScopedRegistries = registries,
Dependencies = dependencies };
        }

        static string
DeserializeDependencies(string json)
        {
            var startIndex =
GetDependenciesStart(json);
            var endIndex =
GetDependenciesEnd(json,
startIndex);
```

```
            if (startIndex ==
INDEX_NOT_FOUND || endIndex
== INDEX_NOT_FOUND)
                return null;

            var dependencies =
json.Substring(startIndex, endIndex -
startIndex);
            return dependencies;
        }

        static int
GetDependenciesStart(string json)
        {
            var dependenciesIndex =
json.IndexOf(DEPENDENCIES_KE
Y,
StringComparison.InvariantCulture);
            if (dependenciesIndex ==
INDEX_NOT_FOUND)
                return
INDEX_NOT_FOUND;

            var dependenciesStartIndex =
json.IndexOf('{', dependenciesIndex
+ DEPENDENCIES_KEY.Length);

            if (dependenciesStartIndex
== INDEX_NOT_FOUND)
                return
INDEX_NOT_FOUND;

            dependenciesStartIndex++;
//add length of '{' to starting point

            return
dependenciesStartIndex;
        }

        static int
GetDependenciesEnd(string
jsonString, int
dependenciesStartIndex)
        {
            return jsonString.IndexOf('}',
dependenciesStartIndex);
        }
    }

    class
UnitySerializableManifestDependenc
iesOnly
    {
        public DependencyPlaceholder
dependencies;
    }

    class UnitySerializableManifest
    {
        public ScopedRegistry[]
scopedRegistries;
        public DependencyPlaceholder
dependencies;
    }

    [Serializable]
    struct ScopedRegistry
    {
        public string name;
        public string url;
        public string[] scopes;

        public override bool
Equals(object obj)
        {
            if (!(obj is ScopedRegistry))
                return false;

            var other =
(ScopedRegistry)obj;

            return name == other.name
&&
                   url == other.url &&

scopes.SequenceEqual(other.scopes);
        }
```

```
        public static bool operator
==(ScopedRegistry a,
ScopedRegistry b)
        {
            return a.Equals(b);
        }

        public static bool operator
!=(ScopedRegistry a,
ScopedRegistry b)
        {
            return !a.Equals(b);
        }

        public override int
GetHashCode()
        {
            var hash = 17;

            foreach (var scope in scopes)
                hash = hash * 23 + (scope
== null ? 0 : scope.GetHashCode());

            hash = hash * 23 + (name ==
null ? 0 : name.GetHashCode());
            hash = hash * 23 + (url ==
null ? 0 : url.GetHashCode());

            return hash;
        }
    }

    [Serializable]
    struct DependencyPlaceholder { }
}
```