Cross Platform Replay Kit

Cross Platform Replay Kit allows to record your screen easily with unified API on both iOS and Android Platforms.

Project Paths

Demos : Assets/Plugins/VoxelBusters/CrossPlatformReplayKit/Demo/Scenes

Settings : Window -> Voxel Busters -> Cross Platform Replay Kit -> Settings

Online Links

Tutorials Support

Settings (Window -> Voxel Busters -> Cross Platform Replay Kit -> Settings)

> **Android (For a more granular control on Android Platform)**
>
> **Video Quality** - Set max quality you want to allow for recording the video on Android

```
 QUALITY_MATCH_SCREEN_SIZE - Video will consider recording at screen resolution if the device supports
QUALITY_1080P       - Video will be recorded at 1080p max resolution if the device supports.
QUALITY_720P            - Video will be recorded at 720p max resolution if the device supports.
QUALITY_480P            - Video will be recorded at 480p max resolution if the device supports.
```

Custom Video Bitrate - Enable this if you want to setup custom video bitrates. Usually default bitrates on Android for video recording uses higher bitrates leading to good quality but bigger file sizes. This setting allows you to choose from optimal recommended settings and choose a factor between min and max values.

> This will give a better file size without compromising much for the quality. Choosing the factor is based on the content length you may record in your game.

Platforms Supported

> iOS 11.0+ (iPhone 5S or later, iPad mini 2 or later, iPod Touch 6th generation, iPad Air or later.) Android API 21 or later (Devices running Android Lollipop & Above)

Plugin Usage

For using this plugin methods you need to import the required namespace.

Please include the below statement before using any plugin methods

> *using VoxelBusters.ReplayKit;*

We expose ****ReplayKitManager**** class for accessing the plugin functionalities.

Initialisation & Query Methods

Check API Availability

Check if the Replay kit features can be usable as the running device needs to be on the supported list(please check Platforms Supported section above). It's good to check this before calling any of the methods.

```
 public void IsAvailable()
{
    bool isRecordingAPIAvailable  =  ReplayKitManager.IsRecordingAPIAvailable();

    string message  = isRecordingAPIAvailable ? "Replay Kit recording API is available!" : "Replay Kit recording API is not availabl

    Debug.Log(message);
}
```

Initialisation

Initialise Replay Kit plugin by calling Initialise method. This initialises the plugin and **DidInitialise** event is triggered with the status

```
 ReplayKitManager.Initialise();
```

Make sure you register to DidInitialise event ahead so that you get the required callbacks.

```
 void OnEnable()
{
    ReplayKitManager.DidInitialise += DidInitialise;
    ...
}
void OnDisable()
{
    ReplayKitManager.DidInitialise -= DidInitialise;
    ...
}

private  void  DidInitialise(ReplayKitInitialisationState  state,  string  message)
{
    Debug.Log("Received Event Callback : DidInitialise [State:"  +  state.ToString()  +  " "  +  "Message:"  +  message);

    switch  (state)
    {
        case  ReplayKitInitialisationState.Success:
            Debug.Log("ReplayKitManager.DidInitialise : Initialisation Success");
            break;
        case  ReplayKitInitialisationState.Failed:
            Debug.Log("ReplayKitManager.DidInitialise : Initialisation Failed with message["+message+"]");
            break;
        default:
            Debug.Log("Unknown State");
            break;
    }
}
```

Is Recording

Check if already there is a screen recording happening. This returns true if a screen recording already stared and running

```
 public void IsRecording()
{
  return ReplayKitManager.IsRecording();
}
```

Is Preview Available

Check if a preview is available for the recording that was done earlier

```
 public void IsPreviewAvailable()
{
    return ReplayKitManager.IsPreviewAvailable();
}
```

Screen Recording API

Start Recording

This starts recording and have an option to pass if microphone needs to be enabled.

> On Android, the game music can only be recorded only through microphone.

```
public void StartRecording()
{
    ReplayKitManager.SetMicrophoneStatus(enable: true);
    ReplayKitManager.StartRecording();
}
```

Stop Recording

This call stops the current recording and do the required steps for making the recorded video available for preview.

```
public void StopRecording()
{
    ReplayKitManager.StopRecording((filePath, error) => {
        Debug.Log("File path available : " + ReplayKitManager.GetPreviewFilePath());
    });
}
```

Callback events for Recording State

We can receive the recording state changes by listening to **DidRecordingStateChange** event. The event informs about the current status of the recording.

- ReplayKitRecordingState.Started : Status triggered when video recording starts
- ReplayKitRecordingState.Stopped : Status triggered when video recording stopped
- ReplayKitRecordingState.Failed : Status triggered when video recording failed recording
- **ReplayKitRecordingState.Available** : Status triggered when recorded video is available for preview. File is available once you get this status. **Only access/preview the recording after getting this status.**

```
 void OnEnable()
{
    // Register to the events
    ReplayKitManager.DidInitialise              += DidInitialise;
    ReplayKitManager.DidRecordingStateChange    += DidRecordingStateChange;
    ...
}
void OnDisable()
{
    // Deregister the events
    ReplayKitManager.DidInitialise              -= DidInitialise;
    ReplayKitManager.DidRecordingStateChange    -= DidRecordingStateChange;
    ...
}

private  void  DidRecordingStateChange(ReplayKitRecordingState  state,  string  message)
{
    Debug.Log("Received Event Callback : DidRecordingStateChange [State:"  +  state.ToString()  +  " "  +  "Message:"  +  message);

    switch(state)
    {
        case  ReplayKitRecordingState.Started:
            Debug.Log("ReplayKitManager.DidRecordingStateChange : Video Recording Started");
            break;
        case  ReplayKitRecordingState.Stopped:
            Debug.Log("ReplayKitManager.DidRecordingStateChange : Video Recording Stopped");
            break;
        case  ReplayKitRecordingState.Failed:
            Debug.Log("ReplayKitManager.DidRecordingStateChange : Video Recording Failed with message ["  +  message+"]");
            break;
        case  ReplayKitRecordingState.Available:
            Debug.Log("ReplayKitManager.DidRecordingStateChange : Video Recording available for preview / file access");
            break;
        default:
            Debug.Log("Unknown State");
            break;
    }
}
```

Preview

Once you recording state changes to **ReplayKitRecordingState.Available** the preview will be available. Calling this method opens up a view/intent to play the recorded video.

This call returns a bool where it's true when the preview is opened successfully. It returns false if preview is not yet ready or there is no recording done earlier.

```
 bool Preview()
{
    if(ReplayKitManager.IsPreviewAvailable())
    {
        return ReplayKitManager.Preview();
    }
    // Still preview is not available. Make sure you call preview after you receive ReplayKitRecordingState.Available status from Di
    return false;
}
```

Discard

Once after recording, the video can be discarded with this call. It's always recommended to call discard incase if you don't need the recording.

This returns true if the recorded video is successfully deleted.

```
bool Discard()
{
    if(ReplayKitManager.IsPreviewAvailable())
    {
        return ReplayKitManager.Discard();
    }
    return false;
}
```

Get Recorded File

Once your recording state changes to **ReplayKitRecordingState.Available** the file can be accessed. This is a file internal to the app and if you want it to use it later, make sure you copy it to your own folders. Else, on next start of recording, the file will get deleted by default.

The file can be used to play it in your own custom video player or upload to your server if needed.

```
string GetRecordingFile()
{
    if(ReplayKitManager.IsPreviewAvailable())
    {
        return ReplayKitManager.GetPreviewFilePath();
    }
    else
    {
        Debug.LogError("File not yet available. Please wait for ReplayKitRecordingState.Available status");
    }
}
```

Save Recorded Video to Gallery

Once you recording state changes to **ReplayKitRecordingState.Available** the file can be saved to gallery. This saves the recorded file to gallery.

```
public void SavePreview() //Saves preview to gallery
{
    if(ReplayKitManager.IsPreviewAvailable())
    {
        ReplayKitManager.SavePreview((error) =>
        {
            Debug.Log("Saved preview to gallery with error : " + ((error == null) ? "null" : error));
        });
    }
    else
    {
        Debug.LogError("Recorded file not yet available. Please wait for ReplayKitRecordingState.Available status");
    }
}
```

Share Recorded Video

Once you recording state changes to **ReplayKitRecordingState.Available** the file can be shared. This allows sharing of the recorded video.

```
public void SharePreview()
 {
     if(ReplayKitManager.IsPreviewAvailable())
     {
         ReplayKitManager.SharePreview();
         Debug.Log("Shared video preview");
     }
     else
    {
        Debug.LogError("Recorded file not yet available. Please wait for ReplayKitRecordingState.Available status");
    }
 }
```

Limitations

Android

Currently on Android, It's not possible to record internal sounds without the microphone. This is because of no public API availability.

> Even though in-game sounds can be recorded in root mode, this is out of the scope of this plugin as it may be removed from store without notice by google anytime if publishing using private API's.

However, you can record the screen with the sound through microphone which is acceptable.