HIGHLIGHTS UNITY PLUGIN INTEGRATION GUIDE

1. INTRODUCTION

A highlight is a moment or interval of time in gameplay when something important occurs, i.e. an event worth viewing, saving, or sharing later. A game may leverage NVIDIA GeForce Experience SDK to gather the end user's interest in its highlight categories, save highlights when they happen as videos and screenshots, and provide a summary of those highlights for viewing, saving, and share at a time convenient to the end user. The Unity plugin for NVIDIA Highlights is developed to make it easy to integrate Highlights into an existing Unity game.

2. SETTING UP

2.1 MACHINE CONFIGURATION

In order to use NVIDIA Highlights you need:

- Windows PC with Windows 7 (64-bit) or newer
- GeForce GTX 900 series or newer
- Shadowplay-ready display driver. Any NVIDIA display driver of version 368.81 or higher will meet this requirement
- Unity version 5.6+
- GeForce Experience 3.9 or above

NOTE:

- We currently do not support Highlights for the following NVIDIA GPU / Display configurations
    - SLI
    - Optimus / Hybrid
    - Surround Support for the above is coming soon, though.

2.2 UNITY PROJECT CONFIGURATION

To enable Highlights in your projects you need to import the Highlights package which you can download from the Asset Store. Once package is downloaded from the store simply select Assets->Import Package->Custom Package from the menu in the Unity editor in order to import it.

3. CONCEPTS

The Unity game interfaces with the GFE runtime via the Highlights.cs C# script that comes bundled with the plugin. The sections below will cover some important concepts that will aid in understanding what the various API functions are meant for and how they can be used during the development process.

Versioning

Because there are two different parts, and the client / user's machine may be mismatched at times, the game should be aware of the versioning system. It's GfeSDK's goal to make this as seamless as possible, but there could still be compatibility issues to be aware of.

The GfeSDK version contains 4 parts, MAJOR.MINOR.BUILD.HASH. The BUILD and HASH components are descriptive and don't have any effect on functionality. The MAJOR component identifies overall compatibility. If the client and server mismatch on the major version number, no communication is possible. There are no current plans to update from 1, breaking communication. The major version number gives a way to show incompatibility if the fundamental architecture of GFE ever changes. The minor version number indicates feature compatibility. When a new feature gets added / modified on the SDK, the minor version number will be bumped. This means that for older games / newer GFE installations, the game is simply missing out on newer features. This will generally not be a problem. For a game with a newer version of the GfeSDK, and a user with an older installation of GFE, some features may not function, and the user should be encouraged to update GFE.

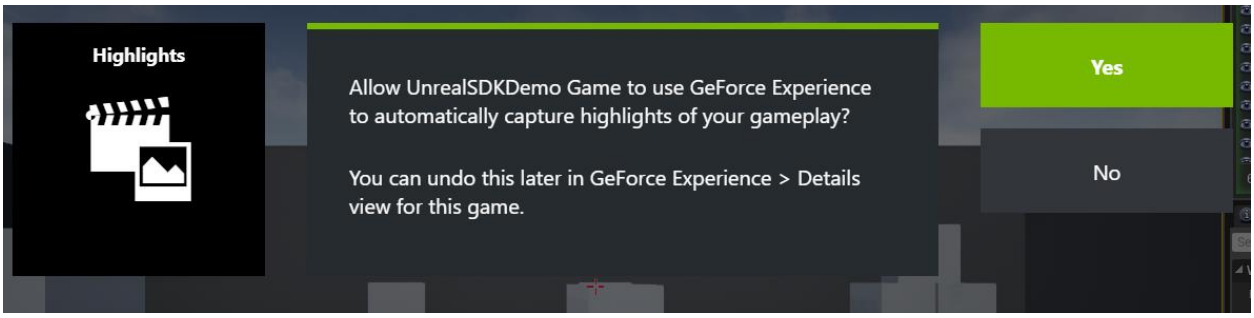
Permissions

Certain actions require permission from the user. For example, recording video for Highlights requires the user to agree to the recording. To achieve this, the app must know what features it wishes to enable. It will pass these "scopes" into the CreateHighlightsSDK() call. Consider the typical Highlights case as an example. The app will pass in a list of the scopes HighlightsScope.Highlights, HighlightsScope.HighlightsRecordVideo, and HighlightsScope.HighlightsRecordScreenshot. The first of these is required in order for any of the “Highlights” calls to succeed and send a message to the server. It will allocate the resources required in the DLL and on the server in order to achieve this. The second of these permissions is required in order to capture video of the gameplay, and the final is to capture a screenshot.

The first time the user runs the game, and the game calls CreateHighlightsSDK(), and passes in these three permissions, the game might receive back that HighlightsScope.Highlights has been granted permission implicitly, but that HighlightsScope.HighlightsRecordVideo and HighlightsScope.HighlightsRecordScreenshot currently have "must ask" permission. In other words, the game must ask GFE for permission to record video before it will succeed in doing so. To achieve this, the game will call RequestPermissions() with two scopes in the list,

HighlightsScope.HighlightsRecordVideo and HighlightsScope.HighlightsRecordScreenshot. It's not necessary to request permission for a scope that has implicitly been granted permission already.

The call to RequestPermissions() is required because it will trigger GFE to put up an In Game Overlay. The game might not want this to occur during CreateHighlightsSDK() time. Once called, the user will see the overlay pop up, asking them for permission.



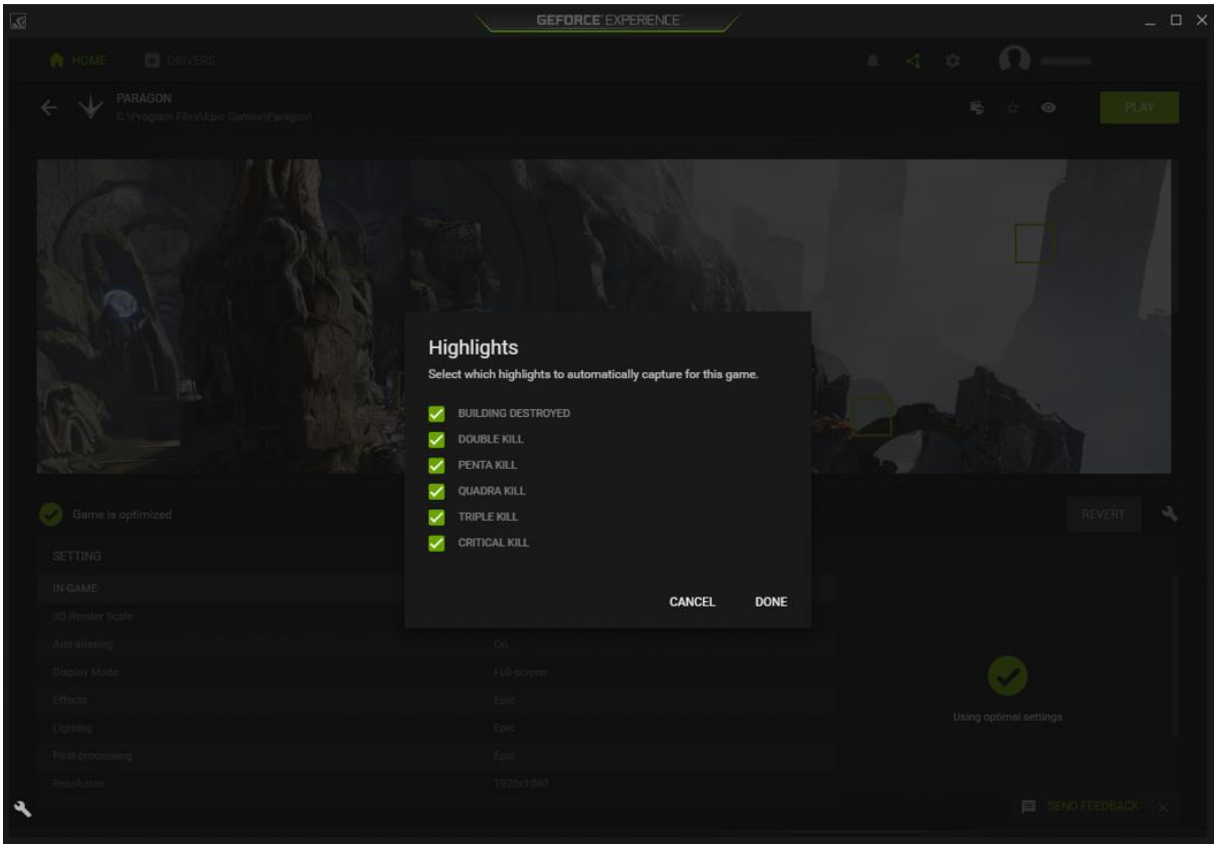
The user can reverse their decision in either case later on in GFE on the games details page.

Asynchronous Calls

Most of the calls to GfeSDK are asynchronous. This is due to the client/server architecture. This is the reason most of the calls do not return any value. They are submitted to the server and are processed there and the server would report errors if any. A client delegate function is passed to the server, for the server to remotely call at the end of server processing.

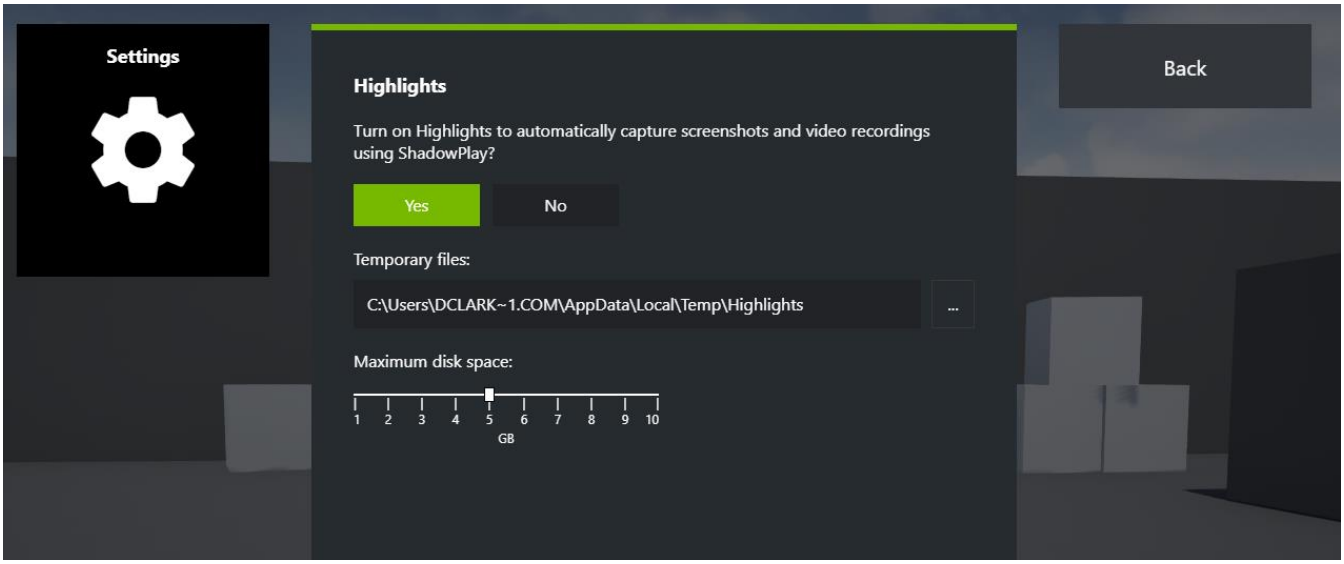
Highlights and Groups

Highlights need to be configured only once during the initialization phase and the data remains persistent on the server till the game exits. A highlight can be identified uniquely with its ID. It also has additional properties such as the significance level of the highlight and also a tag which classifies a highlight into various categories. The “Unannounced” highlight type allows the developer to capture screenshots or videos without notifications, but will still be available in the summary view. The user has the ability to turn of individual highlights. This gives the user flexibility to choose to capture only the highlights he is interested in. This option can be found in GeForce Experience under the specific game’s settings.



A highlight capture is triggered based on a game event. Highlights are also associated with groups. Groups make it easier to decide what highlights are going to be displayed during the summary phase. Each group has a unique Group ID and a translation table which facilitates multi-language support for games. At least one group has to be defined before any highlights can be captured. Groups are defined using a GroupDescriptionTable structure and the defined group can be opened using the OpenGroup() call passing the GroupDescriptionTable as an argument. Depending on the lifetime of a session, global groups may be opened or groups may be opened or closed per session.

In Game Overlay



The In-Game overlay can be used by the user to change Highlights settings, and view Highlights that have been saved to the gallery. It's also used to display the permissions dialog from RequestPermissions(), and the group summary from OpenSummary(). The user can open it up by themselves using the default keybinding Alt+Z.

Logging

As mentioned earlier, calls to GfeSDK are asynchronous and are treated serially. If the server reports SUCCESS or is silent, it means the call was successful. Otherwise it reports error codes which are treated in the native implementation to provide useful error messages. The Unity plugin has a helper function UpdateLog() which reads the current error log buffer and dumps it to Unity's console. This function is designed to be called every frame. More logs from the SDK itself can be found by AttachLogListener().

4. INTEGRATING PLUGIN WITH YOUR GAME

4.1 ADDING HIGHLIGHTS FUNCTIONALITY TO YOUR SCENE

Highlights are meant to capture important moments in a game that a user would like to save for saving beautiful gaming memories, sharing with friends online, etc. Of course all this is meant to be carried out in a non-intrusive way such that a gamer can choose to completely ignore the feature if that is what he/she prefers.

Highlights can be screenshots or videos. The process of setting up highlights involves initializing the Highlights SDK, configuring the highlights that are going to be used in the game, defining groups for the highlights and setting up events to trigger various highlight captures. At the end of a sessions (or wherever appropriate), the user will be able to view all captured highlights and decide what he/she wants to do with it (save, share, edit, delete). Finally when play is complete, the SDK is released. All API calls are asynchronous and that is why they don't return any value. This is in order to maintain smooth operation and uninterrupted gameplay.

4.2 APPLICATION PROGRAM INTERFACE

The Highlights C# script will be imported along with the package from the Asset store. This allows scripts to make use of the Highlights API:

Function calls

Constructs the main SDK interface. Also performs the version check. The function takes the app's name and the scopes that require permissions sent in as an array of HighlightScopes. This should be called before attempting to make any other calls from the API. The return value is a return code which can be checked to see if highlights have been initialized successfully.

```
ReturnCode CreateHighlightsSDK(string AppName, HighlightScope[] RequiredScopes)
```

Release the main SDK interface after create. This should be called when the game exits/no longer needs to use the API.

```
void ReleaseHighlightsSDK()
```

Updates the unity log with info and error messages passed from the highlights sdk. This is useful when debugging. Is most useful when it is in the Update() loop of the script.

```
void UpdateLog()
```

Configure Highlights. Takes an array of highlight definition objects to define highlights in the game. See example for usage information. DefaultConfigureCallback may be used to do simple console logging. Locality like "en-US" can be passed to defaultLocale.

```
void ConfigureHighlights(HighlightDefinition[] highlightDefinitions, string defaultLocale, EmptyCallbackDelegate callback)
```

Request permissions from the user (if not already granted to provide permissions for highlight capture). DefaultRequestPermissionsCallback may be used to do simple console logging.

```
void RequestPermissions(EmptyCallbackDelegate callback)
```

Begins a "group" which groups several Highlights together. As an example, an online game will commonly consist of a round of action, followed by a down period sitting in the lobby. The game can choose to define a Group as the time between starting and finishing this round of action. At the end, all of the highlights recorded during that group may be displayed to the user in a group summary. The OpenGroupParams input parameter takes a structure which has a group id and GroupDescriptionTable which contains language translations for what the group represents. DefaultOpenGroupCallback may be used to do simple console logging.

```
void OpenGroup(OpenGroupParams openGroupParams, EmptyCallbackDelegate callback)
```

Closes out a group and purges the unsaved contents. If the second argument in the CloseGroupParams structure is set to true, the highlights will be deleted when the group is closed. DefaultCloseGroupCallback may be used to do simple console logging.

```
void CloseGroup(CloseGroupParams closeGroupParams, EmptyCallbackDelegate callback)
```

Records a screenshot highlight for the given group. Attached metadata to it to make the Highlight more interesting. DefaultSetScreenshotCallback may be used to do simple console logging.

```
void SetScreenshotHighlight(ScreenshotHighlightParams screenshotHighlightParams, EmptyCallbackDelegate callback)
```

Records a video highlight for the given group. Attached metadata to it to make the Highlight more interesting. Set the start and end delta to change the length of the video clip. DefaultSetVideoCallback may be used to do simple console logging.

```
void SetVideoHighlight(VideoHighlightParams videoHighlightParams, EmptyCallbackDelegate callback)
```

Opens up Summary Dialog for one or more groups. Each GroupView structure stores the id of the group to display in the summary and also a filter for the HighlightTypes and Significance that should be shown in the summary. For example if the HighlightSignificance has HighlightSignificance.Good | HighlightSignificance.VeryGood as a value, then it would only show highlights that have Good and VeryGood significance levels. DefaultOpenSummaryCallback may be used to do simple console logging.

```
void OpenSummary(GroupView[] summaryParams, EmptyCallbackDelegate callback)
```

Retrieves the number of highlights given the group ID and filtering params. DefaultGetNumberOfHighlightsCallback may be used to do simple console logging.

```
void GetNumberOfHighlights(GroupView groupView, GetNumberOfHighlightsCallbackDelegate callback)
```

Retrieves the language set by the user for the GFE onscreen UI. DefaultGetUILanguageCallback may be used to do simple console logging.

```
void GetUILanguage(GetUILanguageCallbackDelegate callback)
```

Retrieves which Highlights are enabled and which Highlights are disabled by the user from the GFE UI. DefaultGetUserSettingsCallback may be used to do simple console logging.

```
void GetUserSettings(GetUserSettingsCallbackDelegate callback)
```

The getter of static member CallbackId, which should be incremented for every async Highlights call and returned by the callback function. The result can be used to map a callback to its original call.

```
int PeekCallbackId()
```

All logs above the level will be stored by Highlights. Look into native SDK document for further details.

```
void SetLogLevel(LogLevel level)
```

Set the processing function of Highlights native log. DefaultLogListener may be used to do simple console logging.

```
void AttachLogListener(LogListenerDelegate listener)
```

All logs above the level will be passed to the listener.

```
void SetListenerLogLevel(LogLevel level)
```

Supporting data structures

Enums

```
enum HighlightScope {
  Highlights,
  HighlightsRecordVideo,
  HighlightsRecordScreenshot,
  Ops
};

enum HighlightType {
  Milestone,
  Achievement,
  Incident,
  StateChange,
  Unannounced
};

enum HighlightSignificance {
  ExtremelyBad,
  VeryBad,
  Bad,
  Neutral,
  Good,
  VeryGood,
```

```
    ExtremelyGood
};

enum Permission {
    Granted,
    Denied,
    MustAsk,
    Unknown
};

enum LogLevel {
    None,
    Error,
    Info,
    Debug,
    Verbose,
};

enum   ReturnCode
{
    SUCCESS = 0,
    SUCCESS_VERSION_OLD_SDK = 1001,
    SUCCESS_VERSION_OLD_GFE = 1002,
    SUCCESS_PENDING = 1003,
    SUCCESS_USER_NOT_INTERESTED = 1004,
    SUCCESS_PERMISSION_GRANTED = 1005,

    ERR_GENERIC = -1001,
    ERR_GFE_VERSION = -1002,
    ERR_SDK_VERSION = -1003,
    ERR_NOT_IMPLEMENTED = -1004,
    ERR_INVALID_PARAMETER = -1005,
    ERR_NOT_SET = -1006,
    ERR_SHADOWPLAY_IR_DISABLED = -1007,
    ERR_SDK_IN_USE = -1008,
    ERR_GROUP_NOT_FOUND = -1009,
    ERR_FILE_NOT_FOUND = -1010,
    ERR_HIGHLIGHTS_SETUP_FAILED = -1011,
    ERR_HIGHLIGHTS_NOT_CONFIGURED = -1012,
    ERR_HIGHLIGHTS_SAVE_FAILED = -1013,
    ERR_UNEXPECTED_EXCEPTION = -1014,
    ERR_NO_HIGHLIGHTS = -1015,
    ERR_NO_CONNECTION = -1016,
    ERR_PERMISSION_NOT_GRANTED = -1017,
    ERR_PERMISSION_DENIED = -1018,
    ERR_INVALID_HANDLE = -1019,
    ERR_UNHANDLED_EXCEPTION = -1020,
    ERR_OUT_OF_MEMORY = -1021,
    ERR_LOAD_LIBRARY = -1022,
    ERR_LIB_CALL_FAILED = -1023,
    ERR_IPC_FAILED = -1024,
    ERR_CONNECTION = -1025,
    ERR_MODULE_NOT_LOADED = -1026,
    ERR_LIB_CALL_TIMEOUT = -1027,
    ERR_APPLICATION_LOOKUP_FAILED = -1028,
    ERR_APPLICATION_NOT_KNOWN = -1029,
    ERR_FEATURE_DISABLED = -1030,
    ERR_APP_NO_OPTIMIZATION = -1031,
    ERR_APP_SETTINGS_READ = -1032,
    ERR_APP_SETTINGS_WRITE = -1033,
};
```

Structures

```
Facilitates multi-language support for the game
struct TranslationEntry {
  string Language;
  string Translation;
};

Definition for a Highlight. The UserDefaultInterest property states whether a highlight is enabled by default. The NameTranslationTable is for translations to various languages.
struct HighlightDefinition {
  string Id;
  bool UserDefaultInterest;
  HighlightType HighlightTags;
  HighlightSignificance Significance;
  TranslationEntry[] NameTranslationTable;
};

Structure that defines the properties of group to be opened
struct OpenGroupParams {
  string Id;
  TranslationEntry[] GroupDescriptionTable;
};

Structure that states the group that has to be closed. Captured highlights from the stated group will be deleted if destroyHighlights is set to true.
struct CloseGroupParams {
  string id;
  bool destroyHighlights;
};

Structure that states the screenshot highlight to be captured and to which group it should be placed.
struct ScreenshotHighlightParams {
  string groupId;
  string highlightId;
};

Structure that states the video highlight to be captured and to which group it should be placed.
struct VideoHighlightParams {
  string groupId;
  string highlightId;
  int startDelta;
  int endDelta;
};

A GroupView definition is required for displaying a summary of highlights. GroupViews can be used to further filter down highlights within the group.
struct GroupView {
  string GroupId;
  HighlightType TagFilter;
  HighlightSignificance SignificanceFilter;
};
```

4.3 SIMPLE HIGHLIGHTS INITIALIZATION IN UNITY

Adding the highlights feature to an existing or a new Unity level is pretty simple. After the plugin has been imported, any script can refer to the available API. The relevant structures and functions are in the ‘Highlights’ class under the ‘NVIDIA’ namespace.

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

using NVIDIA;

public class HighlightsTest : MonoBehaviour {

  // Use this for initialization
  void Start () {
    // Create Highlights SDK
    Highlights.HighlightScope[] RequiredScopes = new Highlights.HighlightScope[3] {
      Highlights.HighlightScope.Highlights,
      Highlights.HighlightScope.HighlightsRecordVideo,
      Highlights.HighlightScope.HighlightsRecordScreenshot };
    System.String AppName = "UnityTestApp";
    if (Highlights.CreateHighlightsSDK (AppName, RequiredScopes) != Highlights.ReturnCode.SUCCESS)
    {
      Debug.LogError("Failed to initialize highlights");
    }

    // Configure Highlights
    Highlights.HighlightDefinition[] highlightDefinitions = new Highlights.HighlightDefinition[2];
    highlightDefinitions [0].Id = "LEVEL_COMPLETE";
    highlightDefinitions [0].HighlightTags = Highlights.HighlightType.Achievement;
    highlightDefinitions [0].Significance = Highlights.HighlightSignificance.Good;
    highlightDefinitions [0].UserDefaultInterest = true;
    highlightDefinitions [0].NameTranslationTable = new Highlights.TranslationEntry[] {
      new Highlights.TranslationEntry ("en-US", "Level Complete"),
      new Highlights.TranslationEntry ("es-mx", "Nivel completado"),
      new Highlights.TranslationEntry ("zh-cn", "级别完成"),
    };
    highlightDefinitions [1].Id = "TRIPLE_KILL";
    highlightDefinitions [1].HighlightTags = Highlights.HighlightType.Milestone;
    highlightDefinitions [1].Significance = Highlights.HighlightSignificance.Neutral;
    highlightDefinitions [1].UserDefaultInterest = true;
    highlightDefinitions [1].NameTranslationTable = new Highlights.TranslationEntry[] {
      new Highlights.TranslationEntry ("en-US", "Triple Kill"),
      new Highlights.TranslationEntry ("es-mx", "Triple matar"),
    };
    Highlights.ConfigureHighlights (highlightDefinitions, "en-US", Highlights.DefaultConfigureCallback);

    // Request Permissions from user
    Highlights.RequestPermissions (Highlights.DefaultRequestPermissionsCallback);

    // Open Groups
    Highlights.OpenGroupParams ogp1 = new Highlights.OpenGroupParams();
    ogp1.Id = "TEST_GROUP1";
    ogp1.GroupDescriptionTable = new Highlights.TranslationEntry[] {
      new Highlights.TranslationEntry ("en-US", "Test Highlight Group 1"),
      new Highlights.TranslationEntry ("es-mx", "Prueba Resaltar Grupo 1"),
      new Highlights.TranslationEntry ("zh-cn", "测试亮点组1"),
    };
    Highlights.OpenGroup (ogp1, Highlights.DefaultOpenGroupCallback);

    Highlights.OpenGroupParams ogp2 = new Highlights.OpenGroupParams();
    ogp2.Id = "TEST_GROUP2";
    ogp2.GroupDescriptionTable = new Highlights.TranslationEntry[] {
      new Highlights.TranslationEntry ("en-US", "Test Highlight Group 2"),
    };
    Highlights.OpenGroup (ogp2, Highlights.DefaultOpenGroupCallback);
  }

  // Update is called once per frame
  void Update () {
    // Update log every frame
```

```
    Highlights.UpdateLog ();
  }

  void OnDestroy() {
    // Release Highlights SDK
    Highlights.ReleaseHighlightsSDK ();
  }
}
```

Please refer to the Unity sample app that demonstrates usage of NVIDIA Highlights for more details. The *NVIDIA Highlights Sample* can be found on the Unity Asset Store.

5. FAQ

In this section we collected commonly occurring problems we've seen while integrating Highlights with games. This section can hopefully help you resolve a problem or two.

5.1 DELETING LOCAL HIGHLIGHTS CONFIG DATA

When an app using highlights requests permissions from a user and permission has been granted/revoked, that information is stored in a 'permissions.json' file in **X**:/Users/**username**/AppData/Local/NVIDIA Corporation/GfeSDK/**AppName**

Where X is the drive letter of the drive where Windows is installed, username is the current user's username and AppName is the name of the respective app.

If the request permissions API call doesn't seem to be showing up the dialog requesting for permission, try deleting this file.

5.2 GENERAL FAQ ON NVIDIA HIGHLIGHTS

The links below provide answers to some additional frequently asked questions related to NVIDIA Highlights in general.

Technical/Integration Questions

https://github.com/NVIDIAGameWorks/GfeSDK/blob/master/doc/NVIDIA_GfeSDK_Developer_FAQ.md

NVIDIA Highlights Frequently Asked Questions

https://github.com/NVIDIAGameWorks/GfeSDK/blob/master/doc/NVIDIA_Highlights_FAQ.md

Usage Guidelines

https://github.com/NVIDIAGameWorks/GfeSDK/blob/master/doc/NVIDIA-HIGHLIGHTS_Usage_Guidelines_.pdf

To access these, an NVIDIA Developer account will be required. It can be created here:

https://developer.nvidia.com/gameworks-source-github

APPENDIX A

Notice

such inclusion and/or use is at customer's own risk. NVIDIA makes no representation or warranty that products based on these specifications will be suitable for any specified use without further testing or modification. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer's sole responsibility to ensure the product is suitable and fit for the application planned by customer and to do the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer's product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this specification. NVIDIA does not accept any liability related to any default, damage, costs or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this specification, or (ii) customer product designs. No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this specification. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA. Reproduction of information in this specification is permissible only if reproduction is approved by NVIDIA in writing, is reproduced without alteration, and is accompanied by all associated conditions, limitations, and notices. ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA's aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the NVIDIA terms and conditions of sale for the product.

VESA DisplayPort DisplayPort and DisplayPort Compliance Logo, DisplayPort Compliance Logo for Dual-mode Sources, and DisplayPort Compliance Logo for Active Cables are trademarks owned by the Video Electronics Standards Association in the United States and other countries.

HDMI HDMI, the HDMI logo, and High-Definition Multimedia Interface are trademarks or registered trademarks of HDMI Licensing LLC.

ROVI Compliance Statement NVIDIA Products that support Rovi Corporation's Revision 7.1.L1 Anti-Copy Process (ACP) encoding technology can only be sold or distributed to buyers with a valid and existing authorization from ROVI to purchase and incorporate the device into buyer's products. This device is protected by U.S. patent numbers 6,516,132; 5,583,936; 6,836,549; 7,050,698; and 7,492,896 and other intellectual property rights. The use of ROVI Corporation's copy protection technology in the device must be authorized by ROVI Corporation and is intended for home and other limited pay-per-view uses only, unless otherwise authorized in writing by ROVI Corporation. Reverse engineering or disassembly is prohibited.

OpenCL OpenCL is a trademark of Apple Inc. used under license to the Khronos Group Inc.

www.nvidia.com