Table of Contents

Getting Started With Unity SDK

- This Chroma SDK plugin requires `Unity 2021.3.15f1` or higher.

See Also

Docs:

- Chroma Animation Guide - Visual examples of the Chroma animation API methods

Plugins:

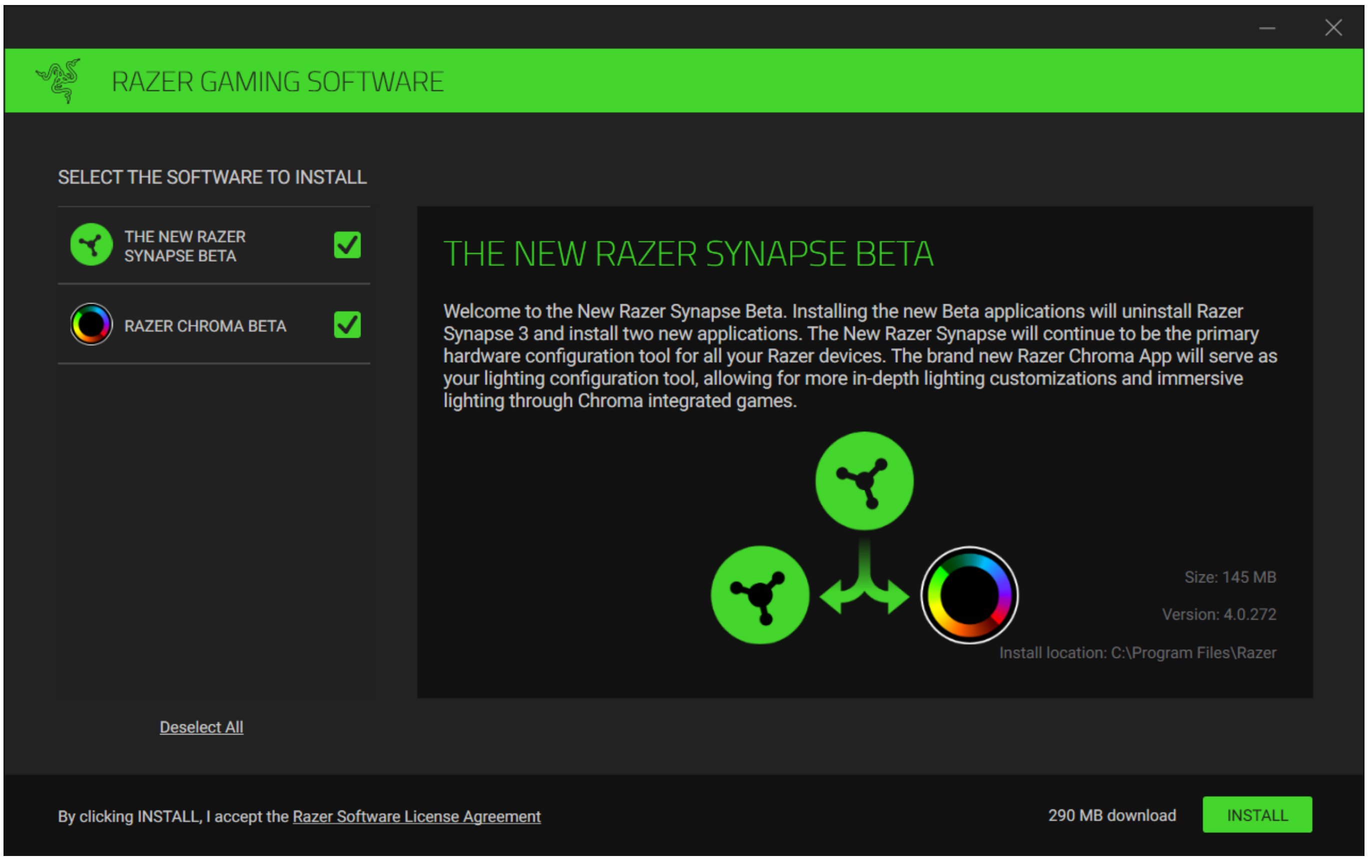
- CChromaEditor - C++ library for playing and editing Chroma animations

User Privacy

Note: The Chroma SDK requires only the minimum amount of information necessary to operate during initialization, including the title of the application or game, description of the application or game, application or game author, and application or game support contact information. This information is displayed in the Chroma app. The Chroma SDK does not monitor or collect any personal data related to users.

Dependencies

To use the Chroma SDK first install the new Razer Synapse and Chroma App.



THE NEW RAZER SYNAPSE BETA

Welcome to the New Razer Synapse Beta. Installing the new Beta applications will uninstall Razer Synapse 3 and install two new applications. The New Razer Synapse will continue to be the primary hardware configuration tool for all your Razer devices. The brand new Razer Chroma App will serve as your lighting configuration tool, allowing for more in-depth lighting customizations and immersive lighting through Chroma integrated games.

Size: 145 MB

Version: 4.0.272

Install location: C:\Program Files\Razer

By clicking INSTALL, I accept the Razer Software License Agreement

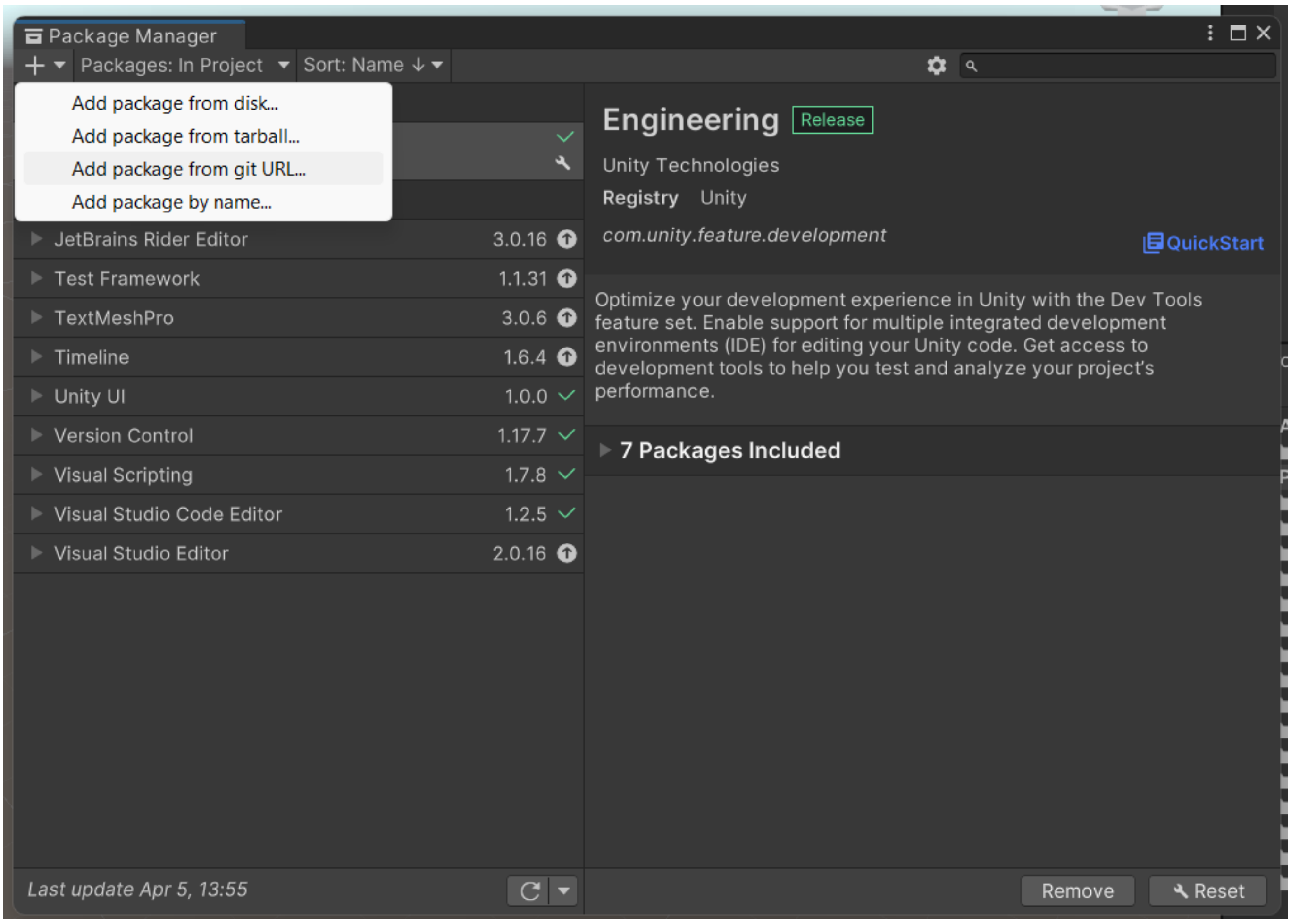
290 MB download

- If you don't have Chroma hardware, you can see Chroma effects with the Chroma Emulator

Requirements and Setup

To import the Chroma SDK into Unity using the Unity Package Manager, follow these steps:

1. Open the Unity Package Manager, then click the + button in the toolbar. Select `Add package from git URL` from the menu.



JetBrains Rider Editor	3.0.16
Test Framework	1.1.31
TextMeshPro	3.0.6
Timeline	1.6.4
Unity UI	1.0.0 ✓
Version Control	1.17.7 ✓
Visual Scripting	1.7.8 ✓
Visual Studio Code Editor	1.2.5 ✓
Visual Studio Editor	2.0.16

Engineering

Unity Technologies

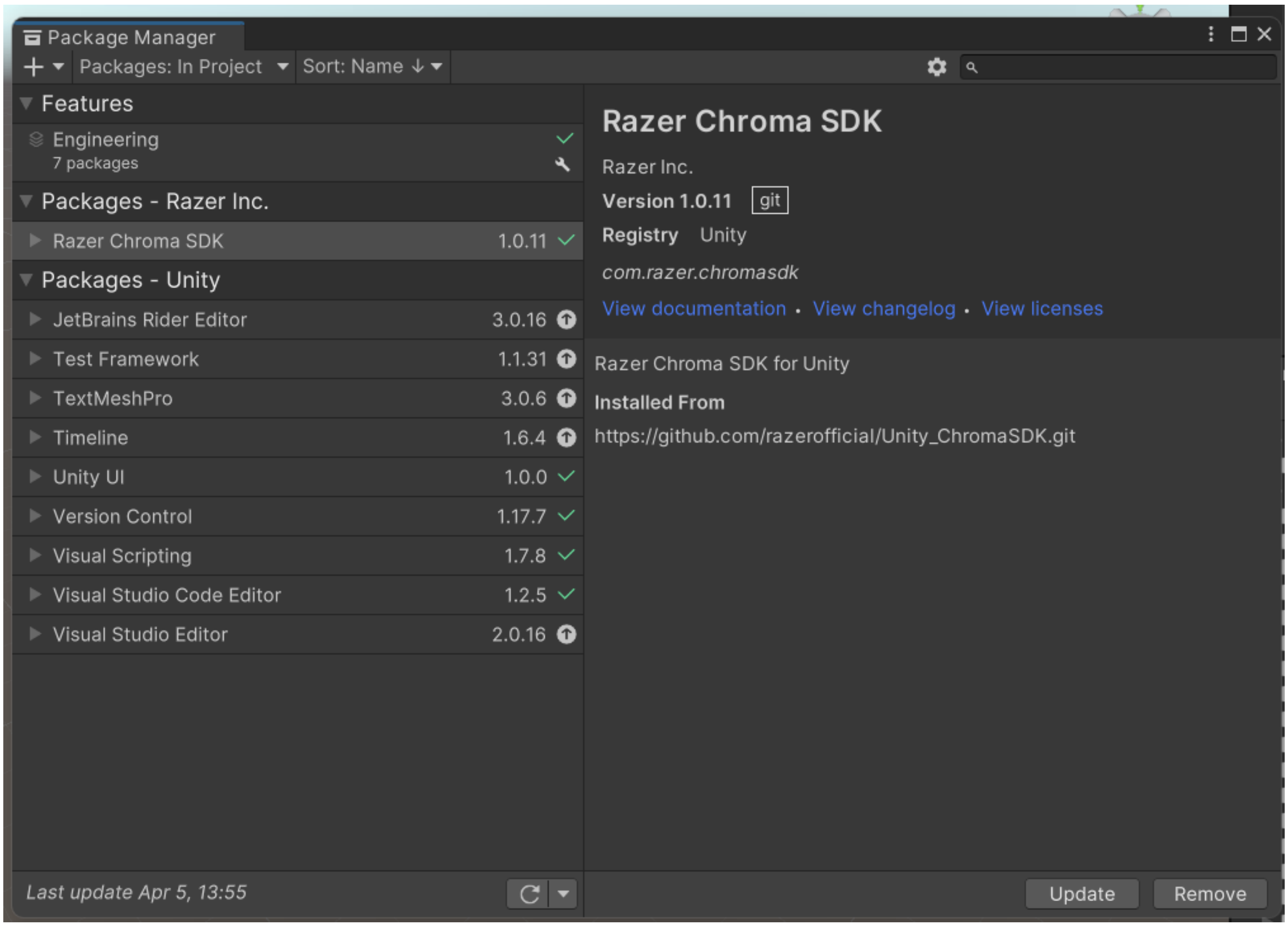
Registry Unity

com.unity.feature.development

Optimize your development experience in Unity with the Dev Tools feature set. Enable support for multiple integrated development environments (IDE) for editing your Unity code. Get access to development tools to help you test and analyze your project's performance.

2. In the text box that appears, enter the URL https://github.com/razerofficial/Unity_ChromaSDK.git and click Add.

If the installation is successful, the `Razer Chroma SDK` package will appear in the package list with the `git` tag.

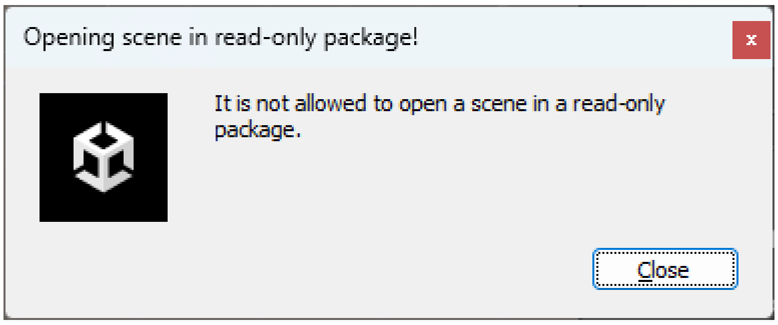


Package	Version
Razer Chroma SDK	1.0.11 ✓
JetBrains Rider Editor	3.0.16
Test Framework	1.1.31
TextMeshPro	3.0.6
Timeline	1.6.4
Unity UI	1.0.0 ✓
Version Control	1.17.7 ✓
Visual Scripting	1.7.8 ✓
Visual Studio Code Editor	1.2.5 ✓
Visual Studio Editor	2.0.16

Razer Chroma SDK
Razer Inc.
Version 1.0.11 git
Registry Unity
com.razer.chromasdk
View documentation • View changelog • View licenses

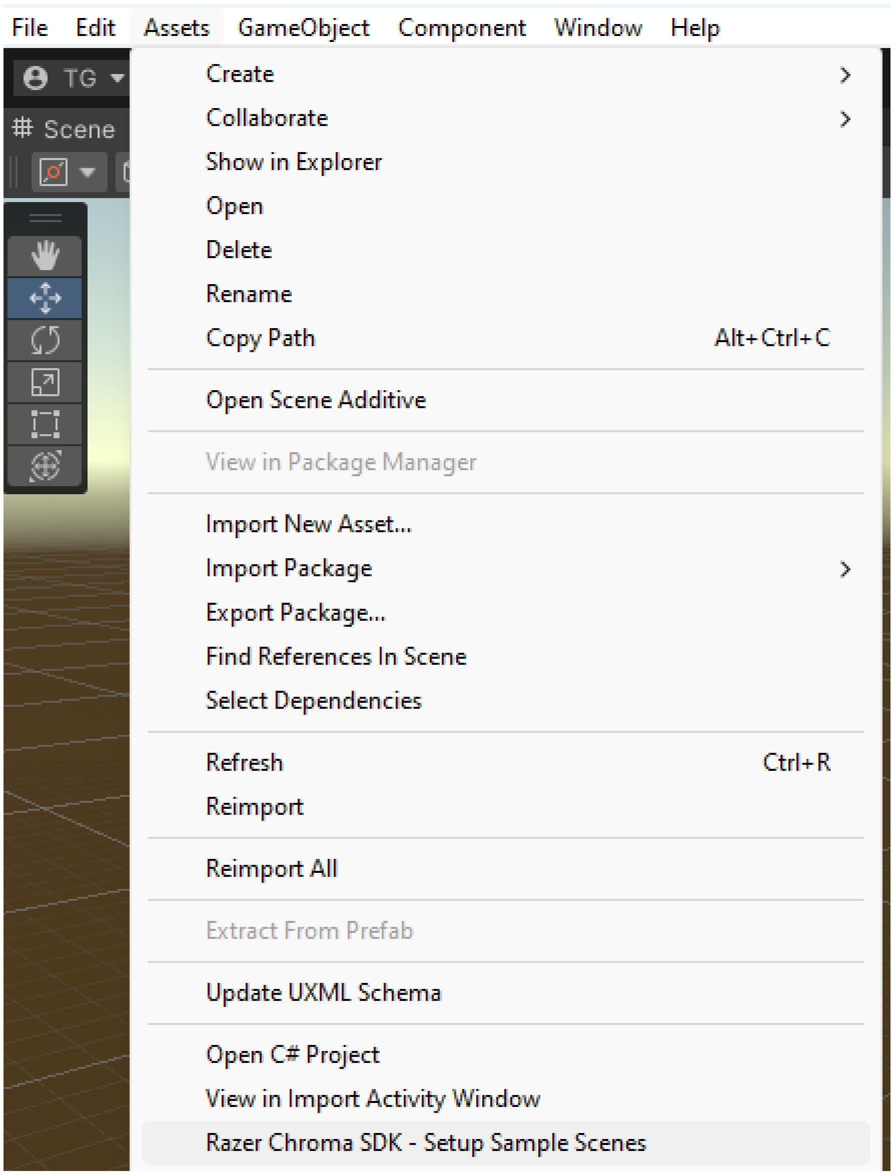
Razer Chroma SDK for Unity
Installed From
https://github.com/razerofficial/Unity_ChromaSDK.git

The package includes samples scenes that download to the `Packages\RazerChromaSDK\Tests\Scenes` folder. Unfortunately, opening scenes in this folder causes a readonly error.

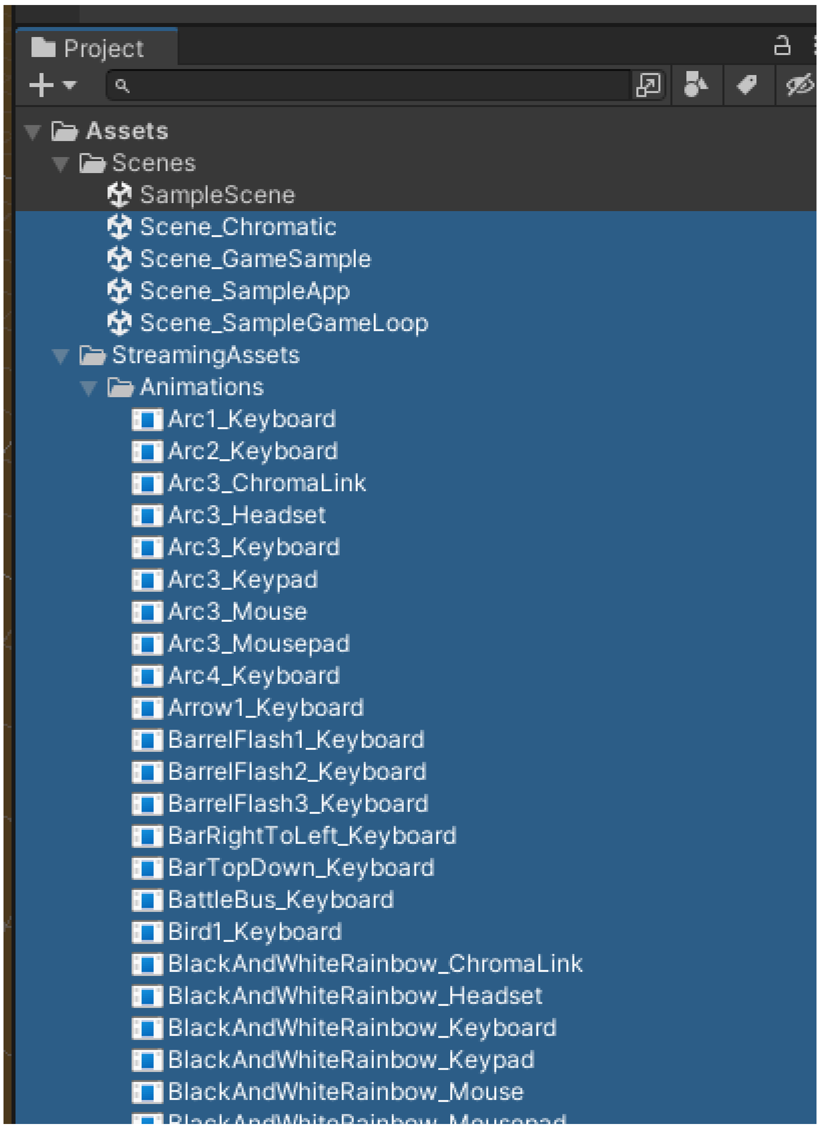

Opening scene in read-only package!

It is not allowed to open a scene in a read-only package.

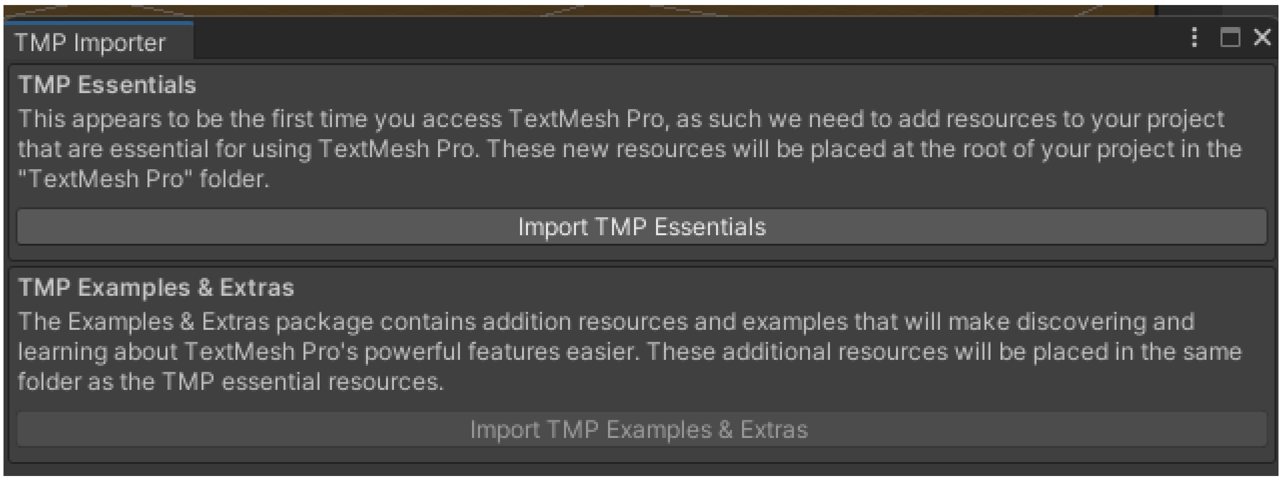
The `Packages\RazerChromaSDK\Editor\RazerChromaSDKMenu.cs` editor script adds a menuitem to setup the sample scenes and `StreamingAssets` sample Chroma animations. The `Assets->Razer ChromaSDK - Setup Sample Scenes` menu item will prepare assets automatically. This avoids the issue with packaged scenes showing the readonly package error.



After the menu item prepares assets, scenes will be added to `Assets/Scenes` and Chroma animations will be added to `Assets/StreamingAssets/Animations`.



Example scenes use `TextMesh Pro` so you may need to import `TMP Essentials` after opening a sample scene.


TMP Importer

TMP Essentials

This appears to be the first time you access TextMesh Pro, as such we need to add resources to your project that are essential for using TextMesh Pro. These new resources will be placed at the root of your project in the "TextMesh Pro" folder.

TMP Examples & Extras

The Examples & Extras package contains addition resources and examples that will make discovering and learning about TextMesh Pro's powerful features easier. These additional resources will be placed in the same folder as the TMP essential resources.

Windows PC

For `Windows PC` builds the `RzChromatic.dll` and `RzChromaStreamPlugin.dll` are not packaged with the build. These libraries are automatically updated and managed by Synapse and the Chroma Connect module. Avoid including these files in your build folder for `Windows PC` builds.

32-bit libraries

```
(Unity Editor 4.X or better)
Packages\RazerChromaSDK\Runtime\Plugins\x86\CChromaEditorLibrary.dll

(Standalone 32-bit builds)
Win32BuildFolder\Project_Data_Folder\Plugins\x86\CChromaEditorLibrary.dll
```

64-bit libraries

```
(Unity Editor 4.X or better)
Packages\RazerChromaSDK\Runtime\Plugins\x64\CChromaEditorLibrary64.dll

(Standalone 64-bit builds)
Win64BuildFolder\Project_Data_Folder\Plugins\x64\CChromaEditorLibrary64.dll
```

Windows Cloud

`Windows Cloud` builds run on cloud platforms using `Windows` such as `Amazon Luna`, `Microsoft Game Pass`, and `NVidia GeForce Now`. Game instances run in the cloud without direct access to Chroma hardware. Chroma effects stream across the Internet to reach your local machine and connected hardware. No extra code is required to add Cloud support. In the case with `NVidia GeForce Now`, the cloud runs the same Epic Games and Steam builds as the desktop version and support Chroma streaming. Viewers can watch the cloud stream via the Razer Stream Portal.

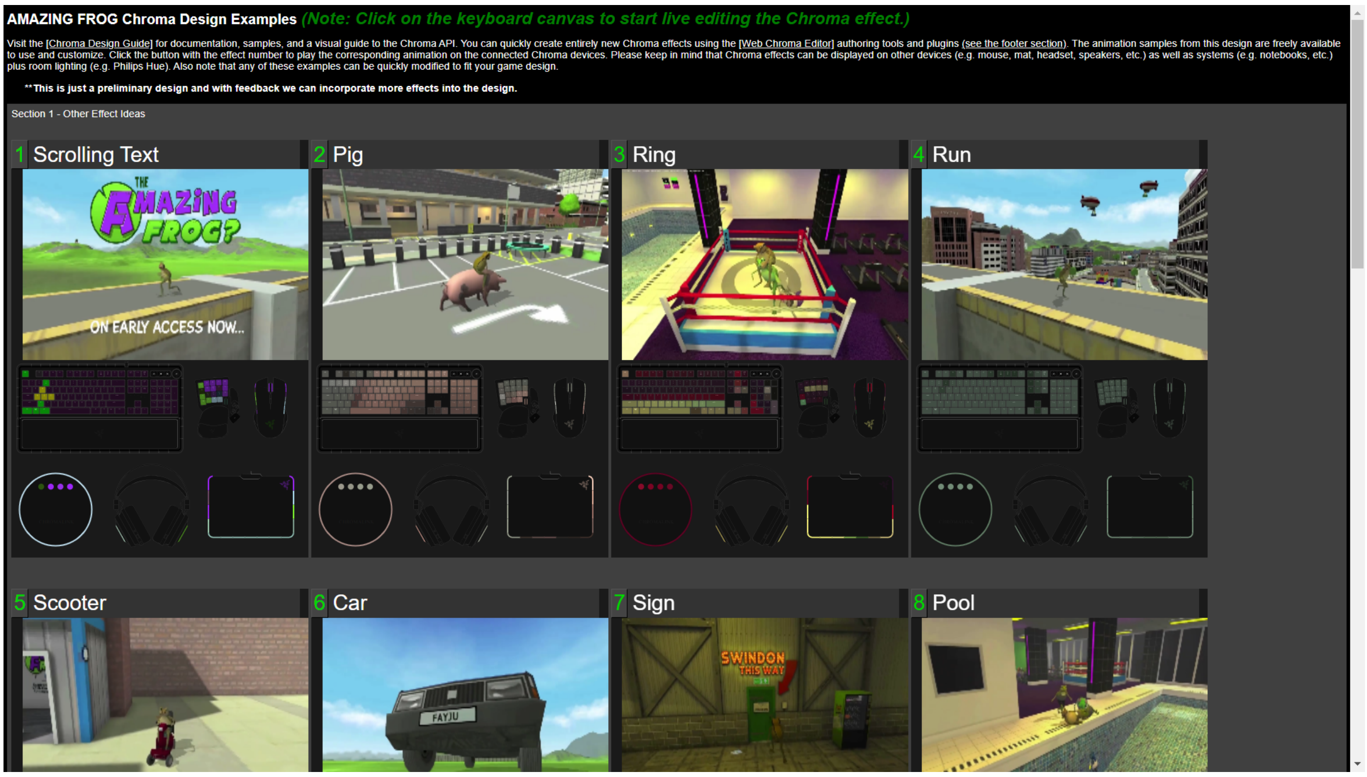
SDK Integration

The SDK integration process involves the following:

1. Chroma Design
2. Revisions
3. Sample Project
4. Tools
5. Integration
6. Testing
7. Haptic Design
8. Modding

Chroma Design

The Chroma Design is the starting point. The team provides 15 sample effects that play on an animated web page. The sample effects correspond to short gameplay video clips and give an idea to the type of animation that could play for a set of game events. The samples are available to use for the specified effect or can be used for any other effect which is completely up to the developer. The developer may ask for effect revisions or additional sample effects. If gameplay video is not available, the developer can provide a description or reference art to conceptualize the desired effect.


AMAZING FROG Chroma Design Examples *(Note: Click on the keyboard canvas to start live editing the Chroma effect.)*

Visit the [Chroma Design Guide] for documentation, samples, and a visual guide to the Chroma API. You can quickly create entirely new Chroma effects using the [Web Chroma Editor] authoring tools and plugins (see the footer section). The animation samples from this design are freely available to use and customize. Click the button with the effect number to play the corresponding animation on the connected Chroma devices. Please keep in mind that Chroma effects can be displayed on other devices (e.g. mouse, mat, headset, speakers, etc.) as well as systems (e.g. notebooks, etc.) plus room lighting (e.g. Philips Hue). Also note that any of these examples can be quickly modified to fit your game design.

****This is just a preliminary design and with feedback we can incorporate more effects into the design.**

Section 1 - Other Effect Ideas

Revisions

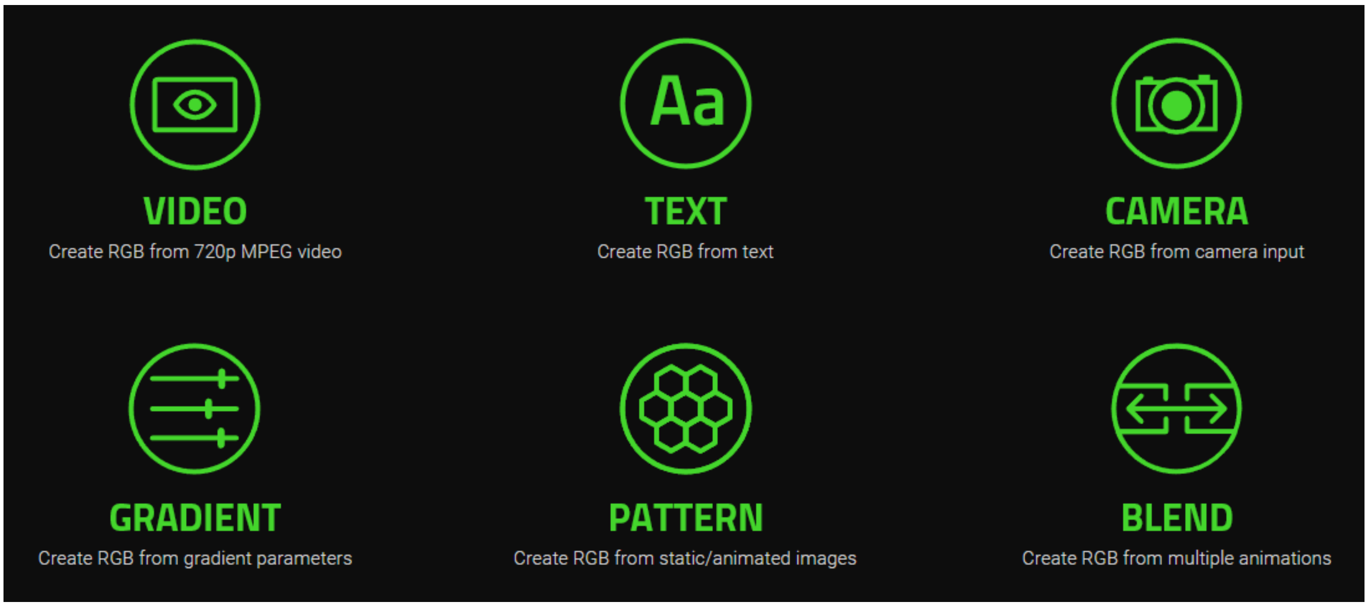
Some Chroma Designs require revisions to add more requested effects or to make changes through the feedback of reviewing the Chroma Design. Revisions can be requested which result in a subset of alterations from the previous design or add completely new game events. **Fill out the Chroma_Sensa_Template_Developers.xlsx Template which provides all the necessary fields for making design requests and revisions.**

Sample Project

The developer specifies which game engine is used by the game so that a sample project can be shared with sample code for the specified engine. The sample project will have the same effects that were defined in the Chroma Design and ported to the target language/game engine. The sample project will include a plugin to add the Chroma SDK to the specified game engine, and the ported sample code and sample animations from the `Chroma Design`.

Tools

- The Web Chroma Editor creates Chroma animations and code snippets from several input sources. Designers can create Chroma animations without writing any code. The toolset can use input sources as video, text, camera, web cam, desktop capture, gradients, patterns, images, and blended animations.


VIDEO
Create RGB from 720p MPEG video
TEXT
Create RGB from text
CAMERA
Create RGB from camera input
GRADIENT
Create RGB from gradient parameters
PATTERN
Create RGB from static/animated images
BLEND
Create RGB from multiple animations

- The Chroma Design Converter can automatically port a web based Chroma Design to several languages and game engines.
- The Synesthesia Console can generate haptic configurations automatically for your Chroma integration.

Integration

The integration process can be as easy as copy and paste from the sample project into the game code. Most likely, it's a matter of finding game triggers in the game code to find the optimal place to add a call to `PlayAnimation()`. The typical Chroma integration process lasts 3 - 5 days for a single developer. Haptics integration can take 0 days by using automatic mode. Manually adding haptics can take about the same amount of work as Chroma to add the calls to `SetEventName()` in the right places. Chroma and haptics are independent meaning sometimes they play together and sometimes they play separately, which is completely up to the designer. **In most cases for game engines after the game build completes, the Chroma animations need to be copied to the animation folder within the game's content folder.**

Testing

The team can provide QA on the game build when integration has completed. Steam beta keys and Epic Store beta keys make testing possible before a game launches. This can be a good way to provide design revisions by testing and giving feedback on the build. To support the QA process, it will be important to include a level selector and potentially console commands that make it easy to navigate the build to test the game triggers at the right moments to validate the visuals work as expected. Beta key access is limited to the engineering and QA review team.

Haptic Design

Just like Chroma Designs, the Haptic Design can be provided by the team. Adding haptic support does not require adding assets to the game. Haptics can be added to a game without code changes and after the game has released. Haptics can be added through creation of a haptic configuration file. Developers can use the Synesthesia Console which automates creation of the haptic configuration file within `HapticFolders` and will add some mockup haptic files (simple haptic effect which can be edited with Haptic Composer) when event names follow a naming convention. Haptic configuration files are automatically distributed by the team through `Chroma App` updates.

Modding

The decision to add Chroma mod support for a title is completely up to the developer. If the developer decides to block modding, Chroma animations can be loaded from a byte array which sandboxes and protects against any modifications to the Chroma animation assets. If the developer wants to use modding, Chroma animation assets are placed within the installation directory. Modders can modify the Chroma animations assets that are loaded by the title. The API provides `CloseAnimation` which reloads the Chroma animation from disk. This allows Chroma animations to be modified externally without needing to relaunch the title. Chroma animation playback also supports relative paths from the content folder. Relative paths can be used to organize several mods within the content folder. The title can have a configuration menu that switches between mod subfolder names which changes the relative path for loading the Chroma animations. The C++ Chroma Mod Sample shows how relative paths can be used to detect and use mods, which is applicable for any game or custom engine.

General

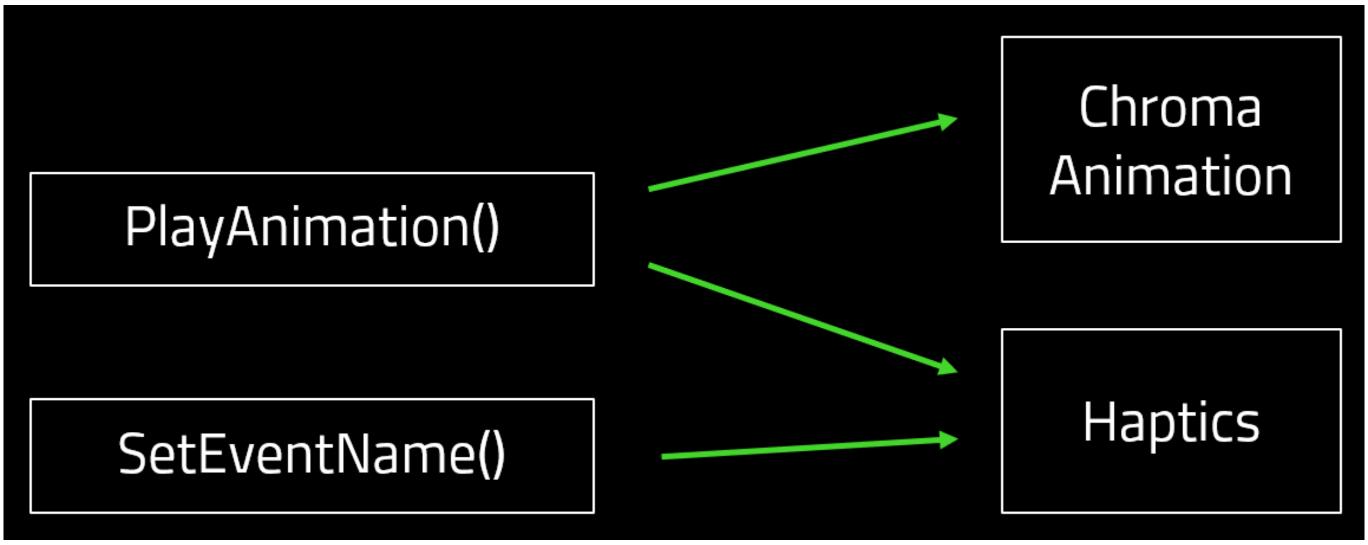
- The Chroma SDK allows an application or game to set the details in the Chroma Apps list within the `Chroma App`.

This document provides a guide to integrating Chroma RGB using the Chroma Unity SDK. Chroma can be included through premade Chroma animations or APIs. Here is the list of available methods:

- Initialize SDK: Initialize the Chroma SDK to use the library.
- Is Active: Check if the app/game has Chroma focus.
- Is Connected: Check if Chroma hardware is connected.
- Play Chroma Animation: Playback a Chroma animation asset.
- Set Event Name: Name a game event or game trigger in order to also add Haptics to the Chroma event.
- Use Forward Chroma Events: Enable or disable automatic invocation of `SetEventName()` when invoking `PlayAnimation()` using the animation name.

Chroma Sensa

Chroma Sensa is the combination of Chroma and Razer Sensa HD Haptics in a single SDK. By integrating RGB lighting and haptics into game environments and events, players can enjoy a truly immersive gaming experience. The `Chroma SDK` is capable of playing Chroma animations and haptics on the Razer Sensa HD Haptics devices. The default mode allows automatic triggering of haptics effects when Chroma animations are played with `PlayAnimation()`. Manual mode is set by `UseForwardChromaEvents(false)` and haptics can be triggered independently of Chroma animations with SetEventName().



Event names can follow a naming convention which assists with the generation of the haptics configuration for your title. Event names are specified with the `SetEventName()` method. The event name suffix can be left off or used to prepopulate common settings for `_ON`, `_OFF`, and `_MERGE`.

- "Jump" - (without a suffix) Existing haptics stop, the named haptic plays to completion and then ends
- "Attack_ON" - Existing haptics continue to play, the named haptic plays as a continuous looping haptic
- "Attack_OFF" - Existing haptics continue to play, the named looping haptic stops
- "Punch_MERGE" - Existing haptics continue to play, the named haptic plays to completion and ends
- "Block_MERGE" - Existing haptics continue to play, the named haptic plays to completion and ends

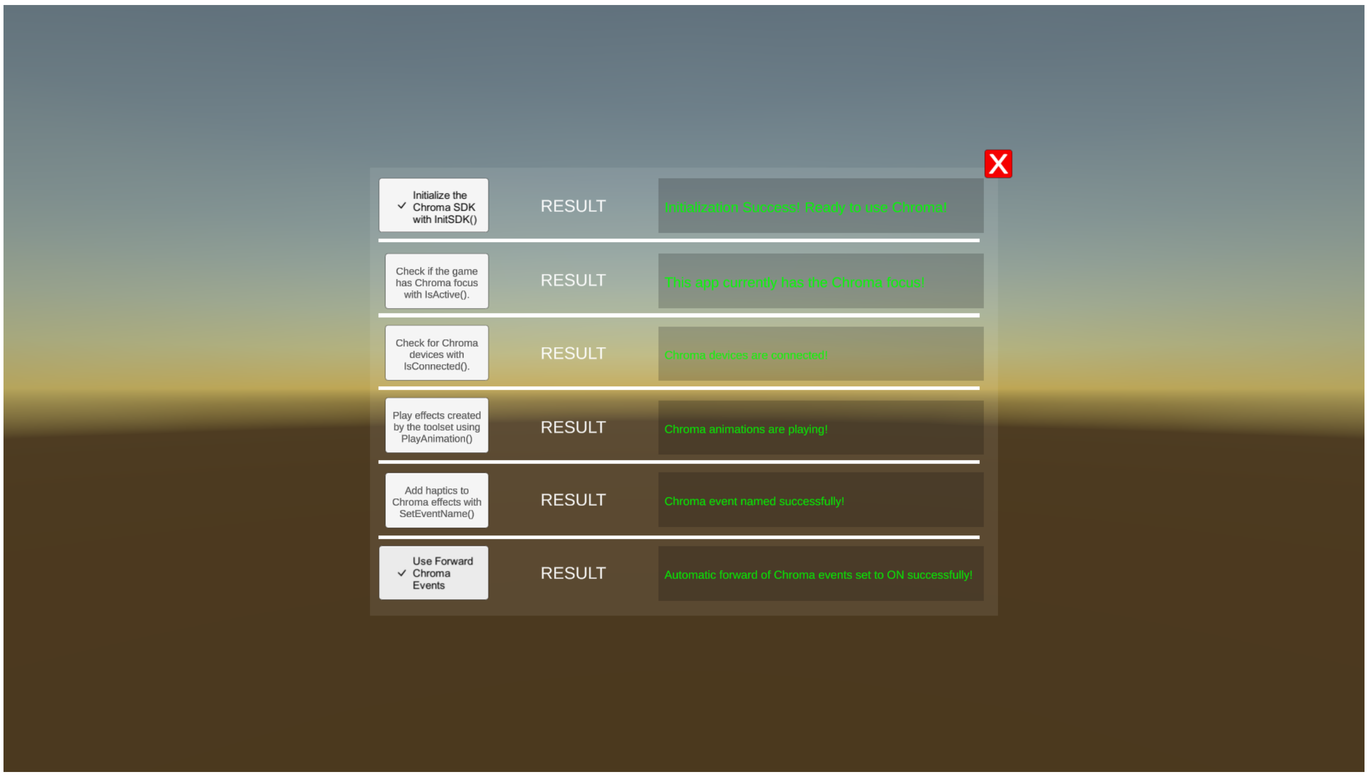
Upon completion of Chroma and haptic implementation, the list of Chroma events and game triggers should be shared with the team to be add to the game's Chroma Workshop entry.

Targeting features can be **optionally** described for each haptics effect.

- "Target" defaults to `"All"`. GroupID options can be found at https://www.interhaptics.com/doc/interhaptics-engine/#groupid
- "Spatialization" defaults to `"Global"`. Other LateralFlag options can be found at https://www.interhaptics.com/doc/interhaptics-engine/#lateralflag
- "Gain" defaults to 1.0.

Chromatic Scene

- The following APIs are demonstrated in the `Assets\Scenes\Scene_Chromatic.unity` sample scene.

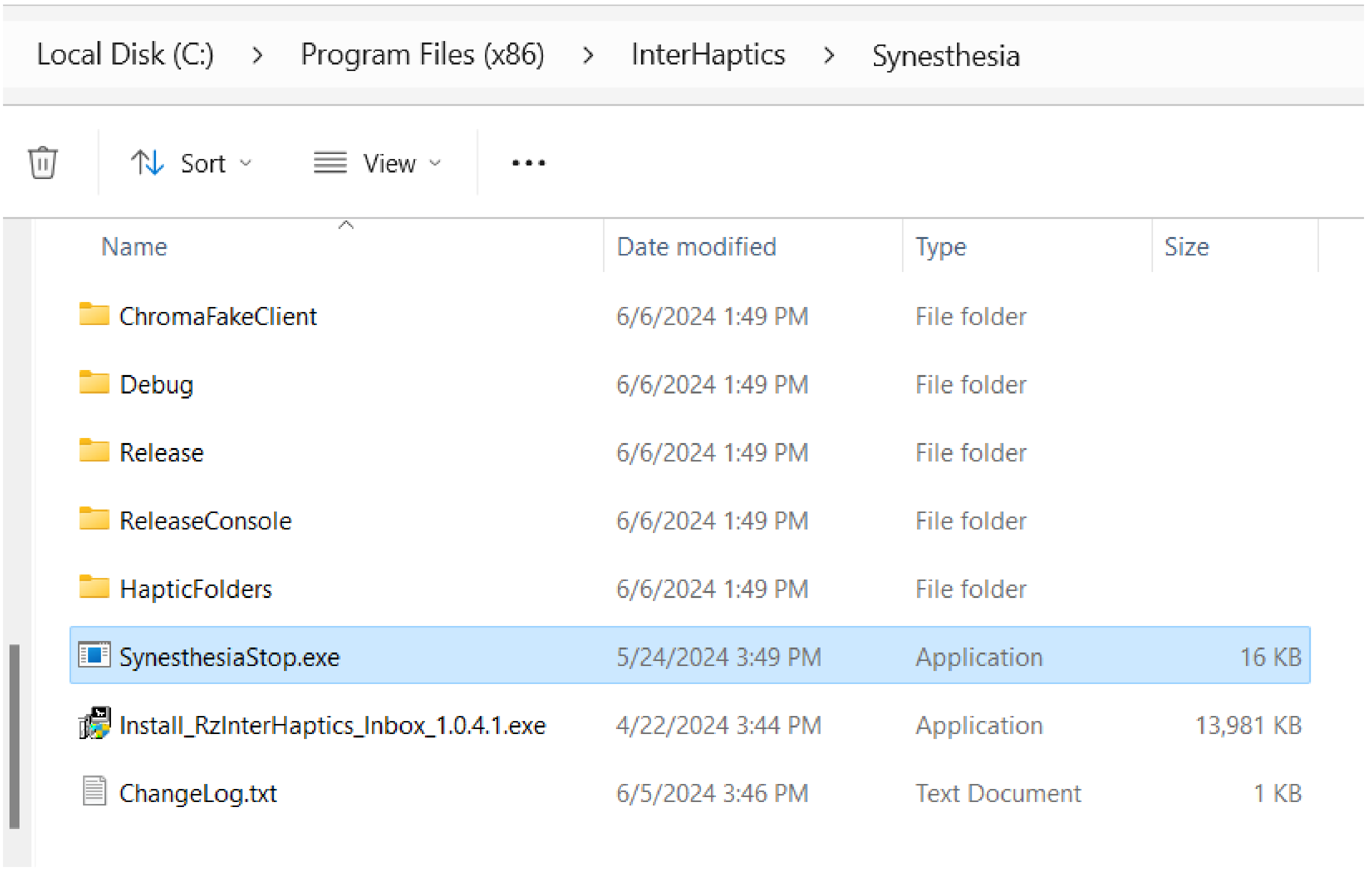

✓ Initialize the Chroma SDK with InitSDK()
Check if the game has Chroma focus with IsActive().
Check for Chroma devices with IsConnected().
Play effects created by the toolset using PlayAnimation()
Add haptics to Chroma effects with SetEventName()
✓ Use Forward Chroma Events
RESULT
X

```
Initialization Success! Ready to use Chroma!
This app currently has the Chroma focus!
Chroma devices are connected!
Chroma animations are playing!
Chroma event named successfully!
Automatic forward of Chroma events set to ON successfully!
```

Synesthesia

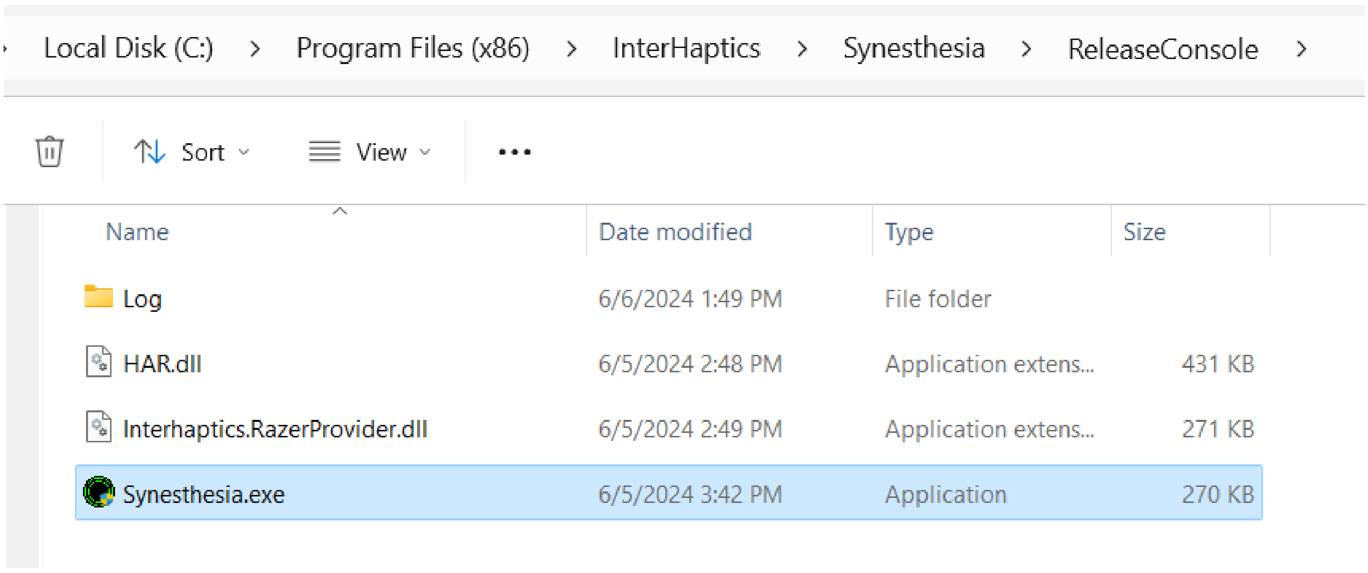
The Synesthesia Console makes creating the haptics configuration for game integration super easy. Download and run the installer to get started creating a haptics config.

1. Run `SynesthesiaStop.exe` to stop any existing background or haptic consoles


Local Disk (C:) > Program Files (x86) > InterHaptics > Synesthesia
Sort
View

Name	Date modified	Type	Size
ChromaFakeClient	6/6/2024 1:49 PM	File folder	
Debug	6/6/2024 1:49 PM	File folder	
Release	6/6/2024 1:49 PM	File folder	
ReleaseConsole	6/6/2024 1:49 PM	File folder	
HapticFolders	6/6/2024 1:49 PM	File folder	
SynesthesiaStop.exe	5/24/2024 3:49 PM	Application	16 KB
Install_RzInterHaptics_Inbox_1.0.4.1.exe	4/22/2024 3:44 PM	Application	13,981 KB
ChangeLog.txt	6/5/2024 3:46 PM	Text Document	1 KB

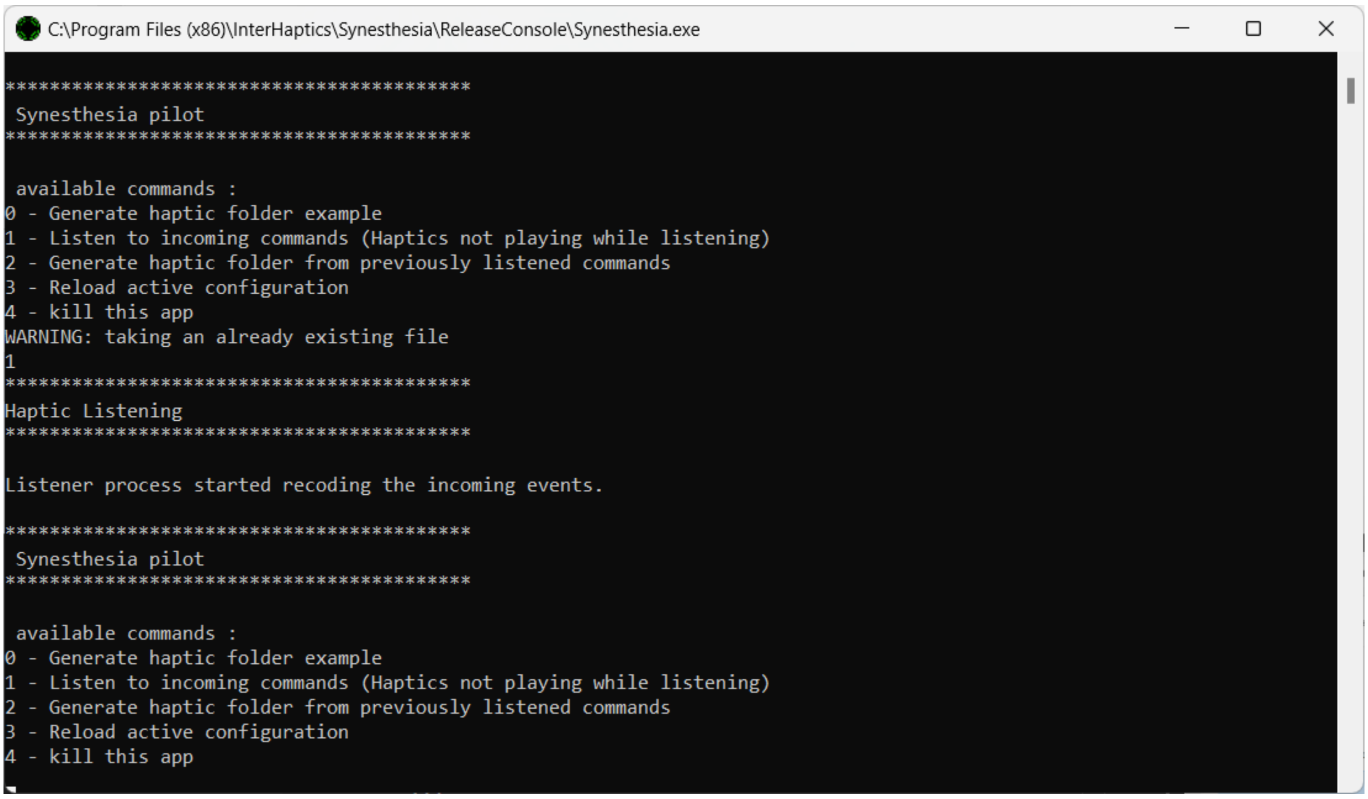
2. Run the `Synesthesia Console` for the interactive prompt



Local Disk (C:) > Program Files (x86) > InterHaptics > Synesthesia > ReleaseConsole >
Sort
View

Name	Date modified	Type	Size
Log	6/6/2024 1:49 PM	File folder	
HAR.dll	6/5/2024 2:48 PM	Application extens...	431 KB
Interhaptics.RazerProvider.dll	6/5/2024 2:49 PM	Application extens...	271 KB
Synesthesia.exe	6/5/2024 3:42 PM	Application	270 KB

3. Enter option `1` and press `Enter` to listen for incoming commands



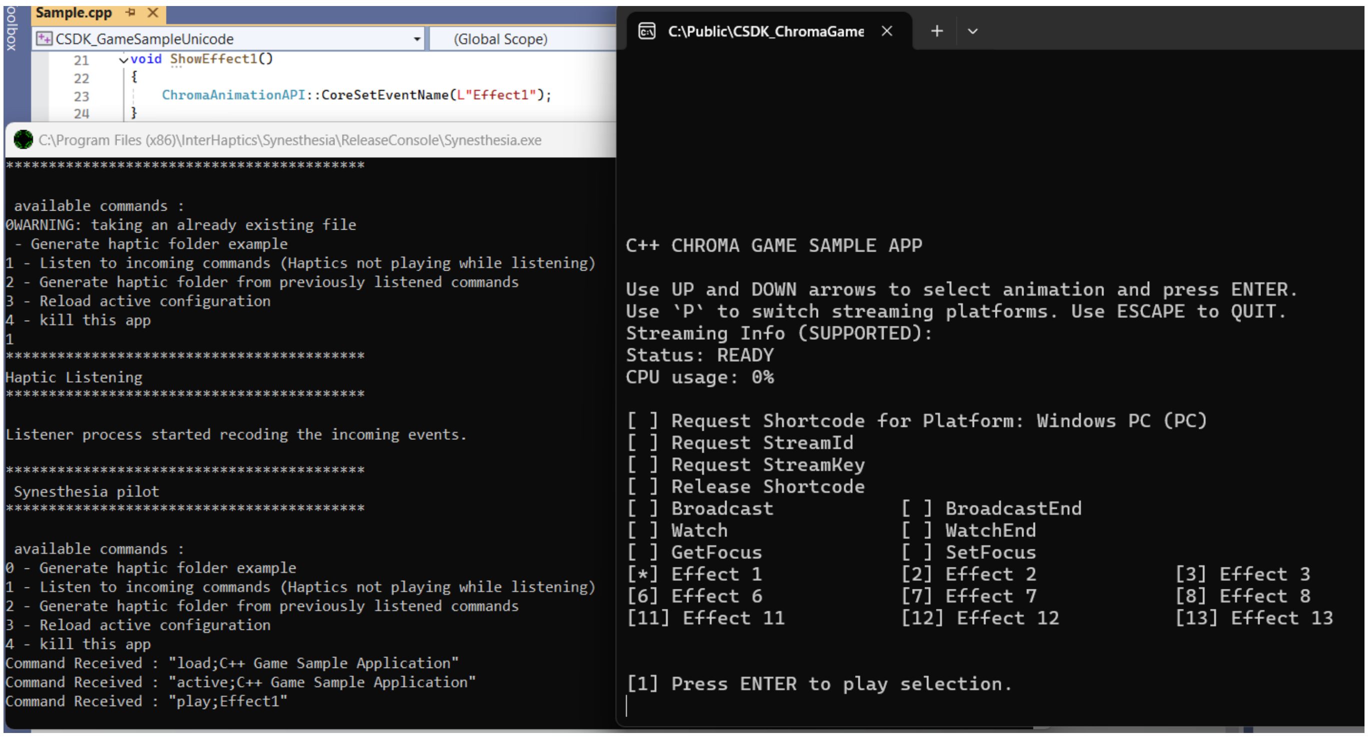
C:\Program Files (x86)\InterHaptics\Synesthesia\ReleaseConsole\Synesthesia.exe

```
*******************************************
 Synesthesia pilot
*******************************************

 available commands :
0 - Generate haptic folder example
1 - Listen to incoming commands (Haptics not playing while listening)
2 - Generate haptic folder from previously listened commands
3 - Reload active configuration
4 - kill this app
WARNING: taking an already existing file
1
*******************************************
Haptic Listening
*******************************************

Listener process started recoding the incoming events.

*******************************************
 Synesthesia pilot
*******************************************

 available commands :
0 - Generate haptic folder example
1 - Listen to incoming commands (Haptics not playing while listening)
2 - Generate haptic folder from previously listened commands
3 - Reload active configuration
4 - kill this app
```

4. Launch your game that uses `PlayAnimation` or `SetEvent` directly to trigger haptic commands.

When the application launches and initializes Chroma, the command to `load` the haptic configuration file is sent. When the application receives Chroma focus, the `active` command is sent. When `PlayAnimation` or `SetEvent` is called, the `play` command is sent.

```
Command Received : "load;C++ Game Sample Application"
Command Received : "active;C++ Game Sample Application"
Command Received : "play;Effect1"
```



Sample.cpp

CSDK_GameSampleUnicode (Global Scope)

```
void ShowEffect1()
{
    ChromaAnimationAPI::CoreSetEventName(L"Effect1");
}
```

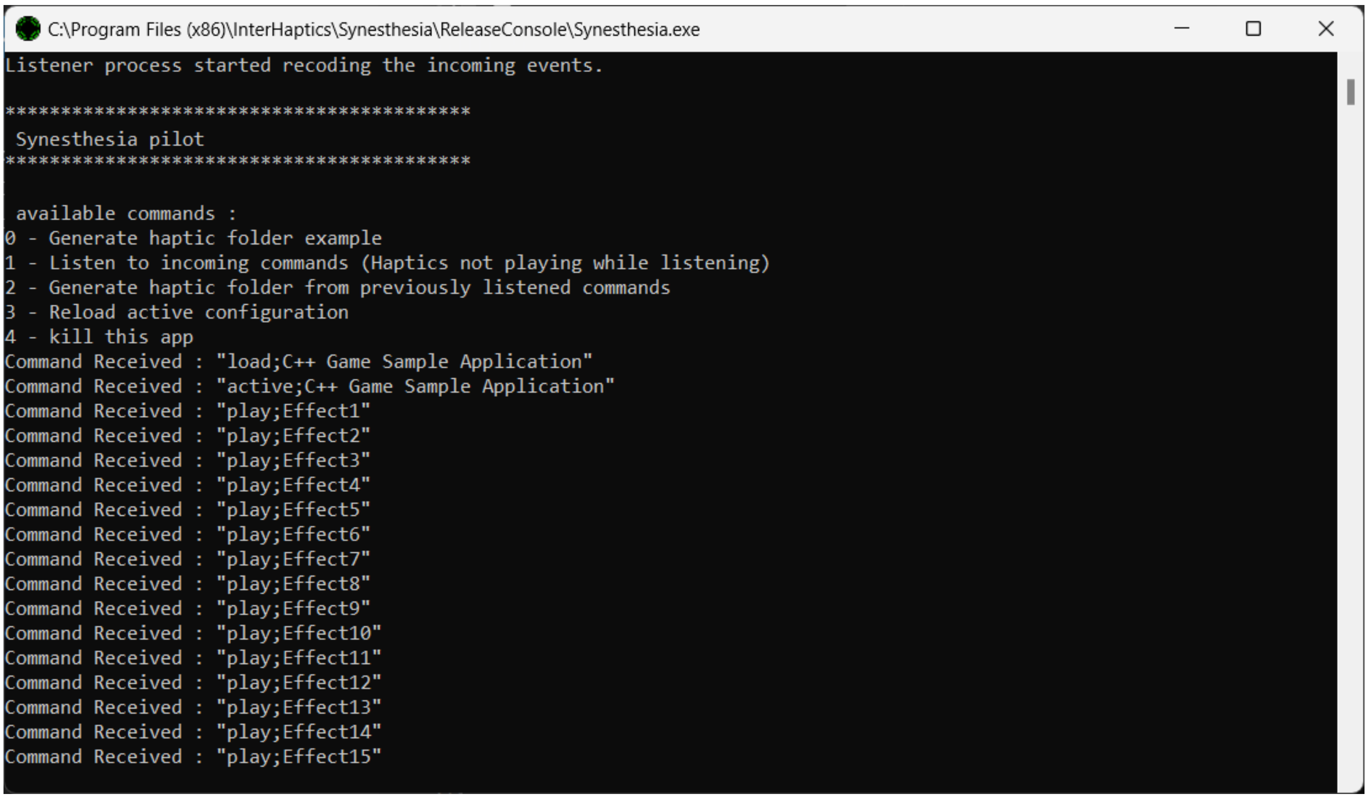
C:\Program Files (x86)\InterHaptics\Synesthesia\ReleaseConsole\Synesthesia.exe

```
******************************************

 available commands :
0WARNING: taking an already existing file
 - Generate haptic folder example
1 - Listen to incoming commands (Haptics not playing while listening)
2 - Generate haptic folder from previously listened commands
3 - Reload active configuration
4 - kill this app
1
******************************************
Haptic Listening
******************************************

Listener process started recoding the incoming events.

******************************************
 Synesthesia pilot
******************************************

 available commands :
0 - Generate haptic folder example
1 - Listen to incoming commands (Haptics not playing while listening)
2 - Generate haptic folder from previously listened commands
3 - Reload active configuration
4 - kill this app
Command Received : "load;C++ Game Sample Application"
Command Received : "active;C++ Game Sample Application"
Command Received : "play;Effect1"
```

C:\Public\CSDK_ChromaGame

```
C++ CHROMA GAME SAMPLE APP

Use UP and DOWN arrows to select animation and press ENTER.
Use `P` to switch streaming platforms. Use ESCAPE to QUIT.
Streaming Info (SUPPORTED):
Status: READY
CPU usage: 0%

[ ] Request Shortcode for Platform: Windows PC (PC)
[ ] Request StreamId
[ ] Request StreamKey
[ ] Release Shortcode
[ ] Broadcast            [ ] BroadcastEnd
[ ] Watch                [ ] WatchEnd
[ ] GetFocus             [ ] SetFocus
[*] Effect 1             [2] Effect 2             [3] Effect 3
[6] Effect 6             [7] Effect 7             [8] Effect 8
[11] Effect 11           [12] Effect 12           [13] Effect 13


[1] Press ENTER to play selection.
```

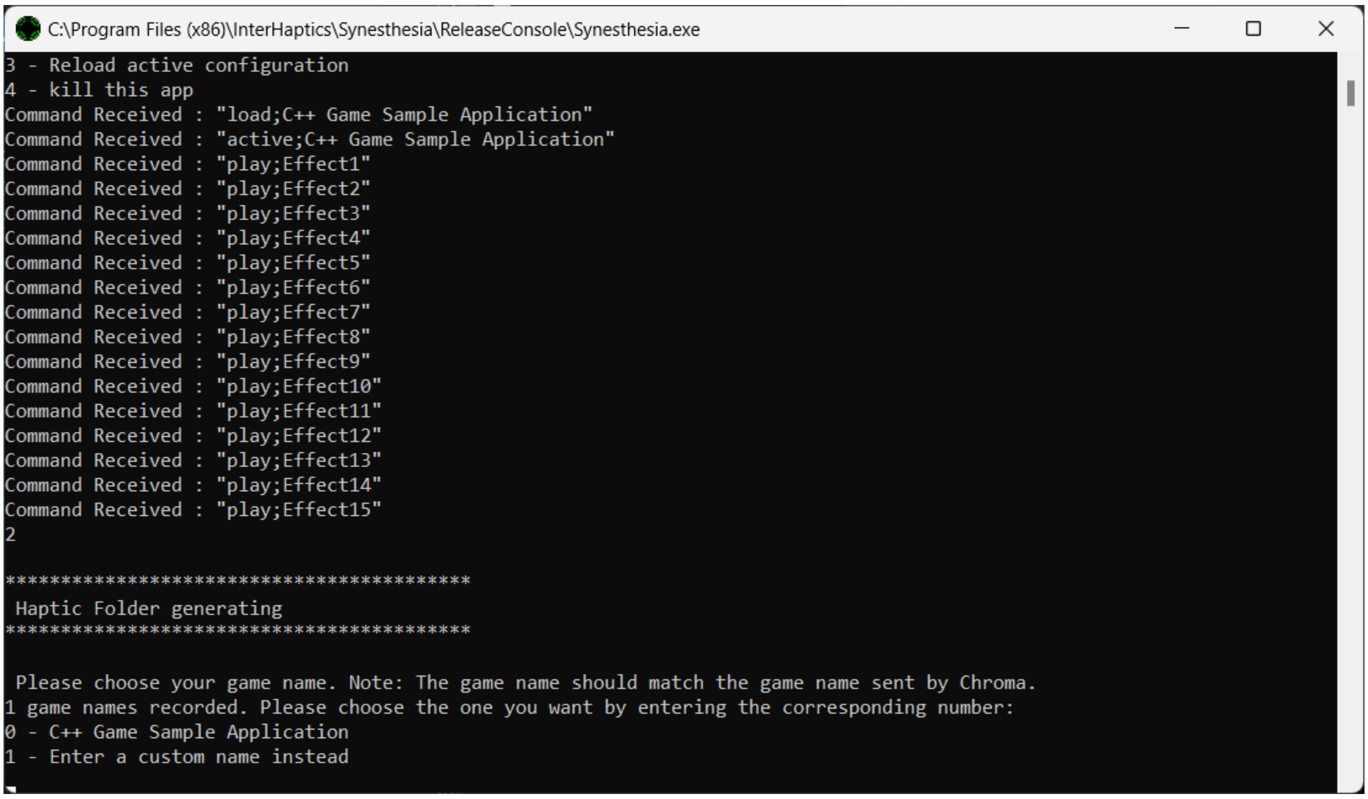
5. Play through all the game triggers to send any possible commands the game might use. This will be useful for generating the haptic configuration next.



C:\Program Files (x86)\InterHaptics\Synesthesia\ReleaseConsole\Synesthesia.exe

```
Listener process started recoding the incoming events.

******************************************
 Synesthesia pilot
******************************************

 available commands :
0 - Generate haptic folder example
1 - Listen to incoming commands (Haptics not playing while listening)
2 - Generate haptic folder from previously listened commands
3 - Reload active configuration
4 - kill this app
Command Received : "load;C++ Game Sample Application"
Command Received : "active;C++ Game Sample Application"
Command Received : "play;Effect1"
Command Received : "play;Effect2"
Command Received : "play;Effect3"
Command Received : "play;Effect4"
Command Received : "play;Effect5"
Command Received : "play;Effect6"
Command Received : "play;Effect7"
Command Received : "play;Effect8"
Command Received : "play;Effect9"
Command Received : "play;Effect10"
Command Received : "play;Effect11"
Command Received : "play;Effect12"
Command Received : "play;Effect13"
Command Received : "play;Effect14"
Command Received : "play;Effect15"
```

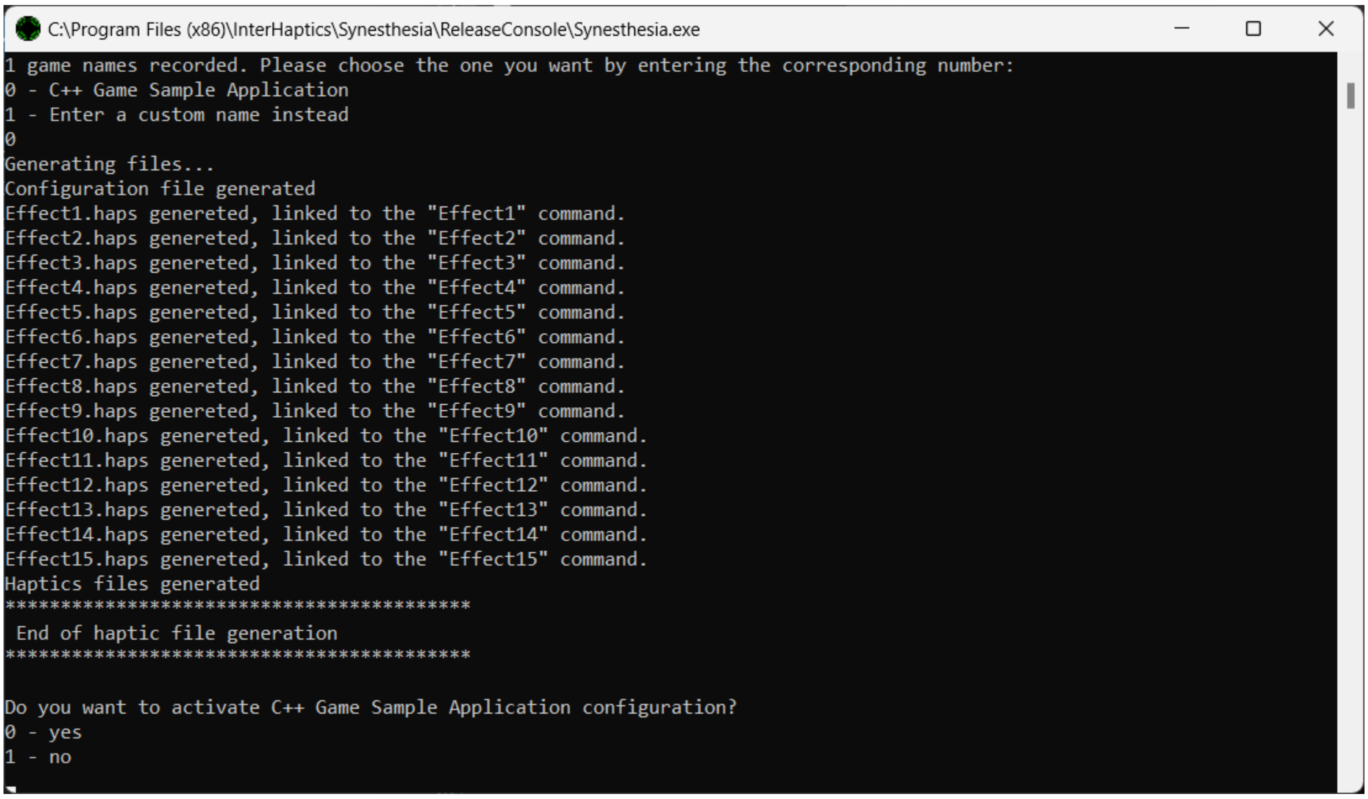
6. Enter option `2` and press `Enter` to generate the haptics configuration



C:\Program Files (x86)\InterHaptics\Synesthesia\ReleaseConsole\Synesthesia.exe

```
3 - Reload active configuration
4 - kill this app
Command Received : "load;C++ Game Sample Application"
Command Received : "active;C++ Game Sample Application"
Command Received : "play;Effect1"
Command Received : "play;Effect2"
Command Received : "play;Effect3"
Command Received : "play;Effect4"
Command Received : "play;Effect5"
Command Received : "play;Effect6"
Command Received : "play;Effect7"
Command Received : "play;Effect8"
Command Received : "play;Effect9"
Command Received : "play;Effect10"
Command Received : "play;Effect11"
Command Received : "play;Effect12"
Command Received : "play;Effect13"
Command Received : "play;Effect14"
Command Received : "play;Effect15"
2

******************************************
 Haptic Folder generating
******************************************

 Please choose your game name. Note: The game name should match the game name sent by Chroma.
1 game names recorded. Please choose the one you want by entering the corresponding number:
0 - C++ Game Sample Application
1 - Enter a custom name instead
```

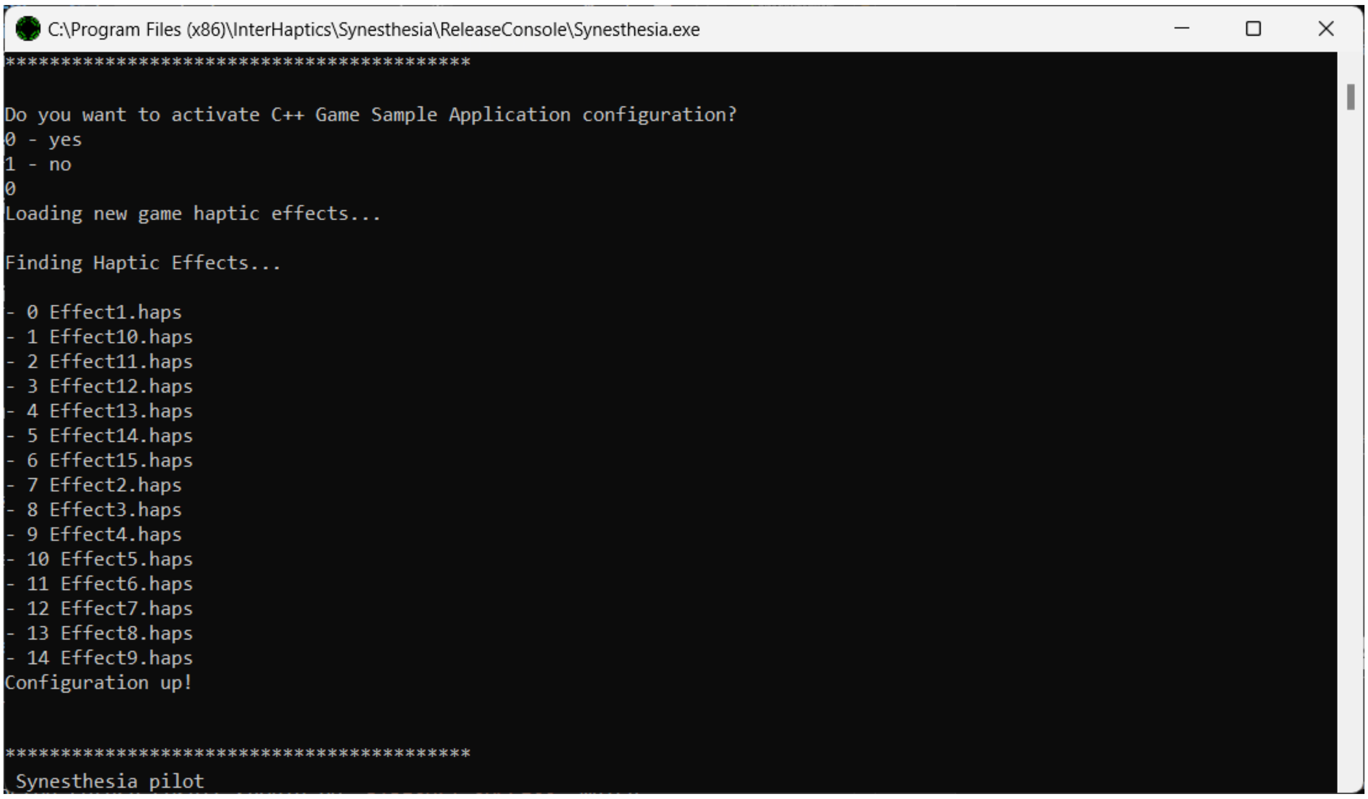
7. Enter option `0` and press `Enter` to use the detected application name used by the Chroma initialization



C:\Program Files (x86)\InterHaptics\Synesthesia\ReleaseConsole\Synesthesia.exe

```
1 game names recorded. Please choose the one you want by entering the corresponding number:
0 - C++ Game Sample Application
1 - Enter a custom name instead
0
Generating files...
Configuration file generated
Effect1.haps genereted, linked to the "Effect1" command.
Effect2.haps genereted, linked to the "Effect2" command.
Effect3.haps genereted, linked to the "Effect3" command.
Effect4.haps genereted, linked to the "Effect4" command.
Effect5.haps genereted, linked to the "Effect5" command.
Effect6.haps genereted, linked to the "Effect6" command.
Effect7.haps genereted, linked to the "Effect7" command.
Effect8.haps genereted, linked to the "Effect8" command.
Effect9.haps genereted, linked to the "Effect9" command.
Effect10.haps genereted, linked to the "Effect10" command.
Effect11.haps genereted, linked to the "Effect11" command.
Effect12.haps genereted, linked to the "Effect12" command.
Effect13.haps genereted, linked to the "Effect13" command.
Effect14.haps genereted, linked to the "Effect14" command.
Effect15.haps genereted, linked to the "Effect15" command.
Haptics files generated
******************************************
 End of haptic file generation
******************************************

Do you want to activate C++ Game Sample Application configuration?
0 - yes
1 - no
```

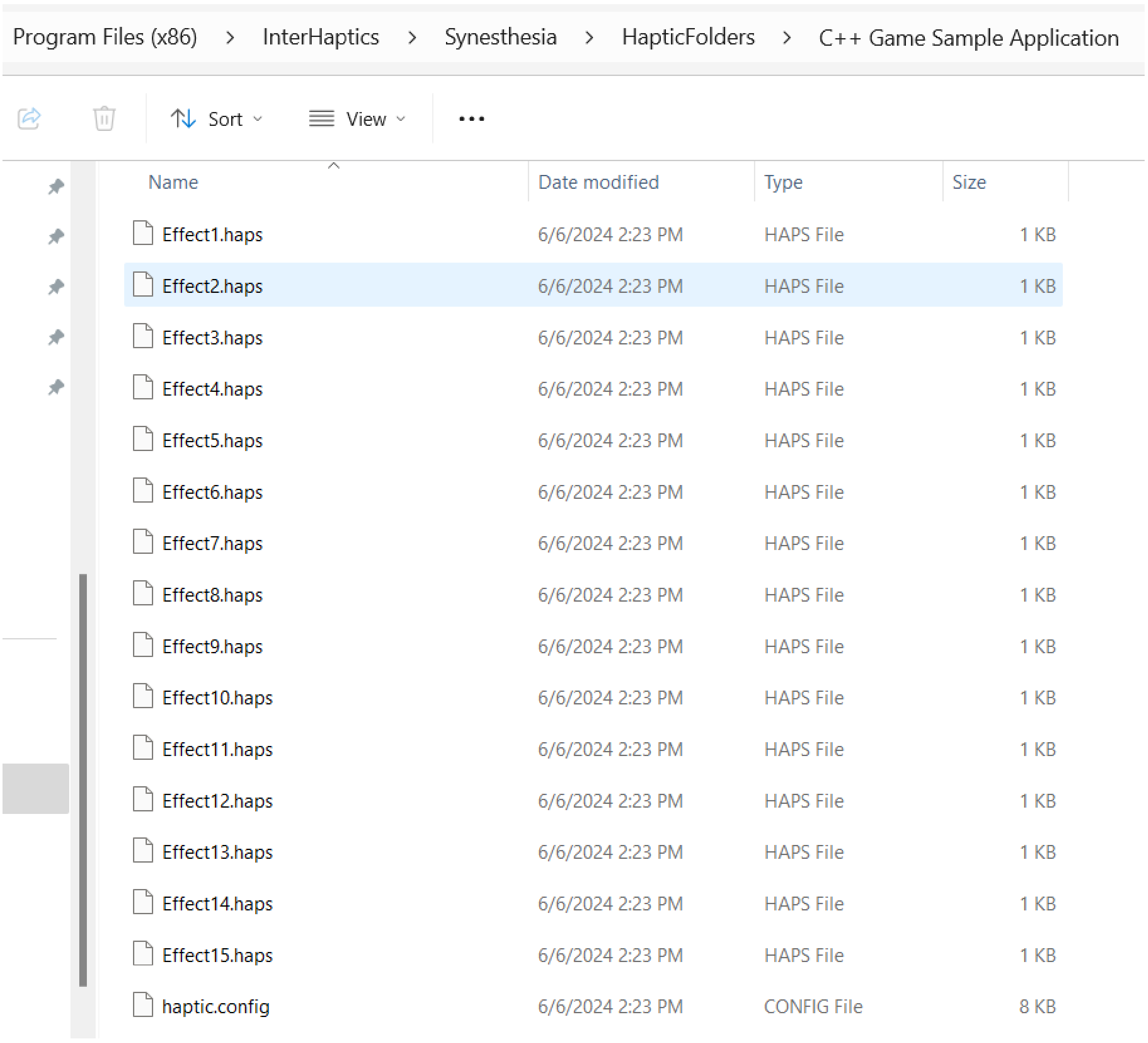
8. Enter option `0` and press `Enter` to use activate the new haptic configuration file. Now when the game triggers haptic events, the configured haptic events will play.



C:\Program Files (x86)\InterHaptics\Synesthesia\ReleaseConsole\Synesthesia.exe

```
*******************************************

Do you want to activate C++ Game Sample Application configuration?
0 - yes
1 - no
0
Loading new game haptic effects...

Finding Haptic Effects...

- 0 Effect1.haps
- 1 Effect10.haps
- 2 Effect11.haps
- 3 Effect12.haps
- 4 Effect13.haps
- 5 Effect14.haps
- 6 Effect15.haps
- 7 Effect2.haps
- 8 Effect3.haps
- 9 Effect4.haps
- 10 Effect5.haps
- 11 Effect6.haps
- 12 Effect7.haps
- 13 Effect8.haps
- 14 Effect9.haps
Configuration up!


*******************************************
 Synesthesia pilot
```

The `haptic.config` and `haps` default haptics effects were generated in the `HapticFolders` by the console.


Program Files (x86) > InterHaptics > Synesthesia > HapticFolders > C++ Game Sample Application

Name	Date modified	Type	Size
Effect1.haps	6/6/2024 2:23 PM	HAPS File	1 KB
Effect2.haps	6/6/2024 2:23 PM	HAPS File	1 KB
Effect3.haps	6/6/2024 2:23 PM	HAPS File	1 KB
Effect4.haps	6/6/2024 2:23 PM	HAPS File	1 KB
Effect5.haps	6/6/2024 2:23 PM	HAPS File	1 KB
Effect6.haps	6/6/2024 2:23 PM	HAPS File	1 KB
Effect7.haps	6/6/2024 2:23 PM	HAPS File	1 KB
Effect8.haps	6/6/2024 2:23 PM	HAPS File	1 KB
Effect9.haps	6/6/2024 2:23 PM	HAPS File	1 KB
Effect10.haps	6/6/2024 2:23 PM	HAPS File	1 KB
Effect11.haps	6/6/2024 2:23 PM	HAPS File	1 KB
Effect12.haps	6/6/2024 2:23 PM	HAPS File	1 KB
Effect13.haps	6/6/2024 2:23 PM	HAPS File	1 KB
Effect14.haps	6/6/2024 2:23 PM	HAPS File	1 KB
Effect15.haps	6/6/2024 2:23 PM	HAPS File	1 KB
haptic.config	6/6/2024 2:23 PM	CONFIG File	8 KB

The `haptic.config` contains default targeting for the generated entries for each detected command.

```
{
    "ExternalCommands": [
        {
            "External_Command_ID": "Effect1",
            "Haptic_Events": [
                {
                    "Haptic_Effect": "Effect1",
                    "Loop": 1,
                    "Mixing": "Override",
                    "Targeting": [
                        {
                            "Gain": 1.0,
                            "Spatialization": "Global",
                            "Target": "All"
                        }
                    ]
                }
                ]
            },
            ...
        ]
}
```

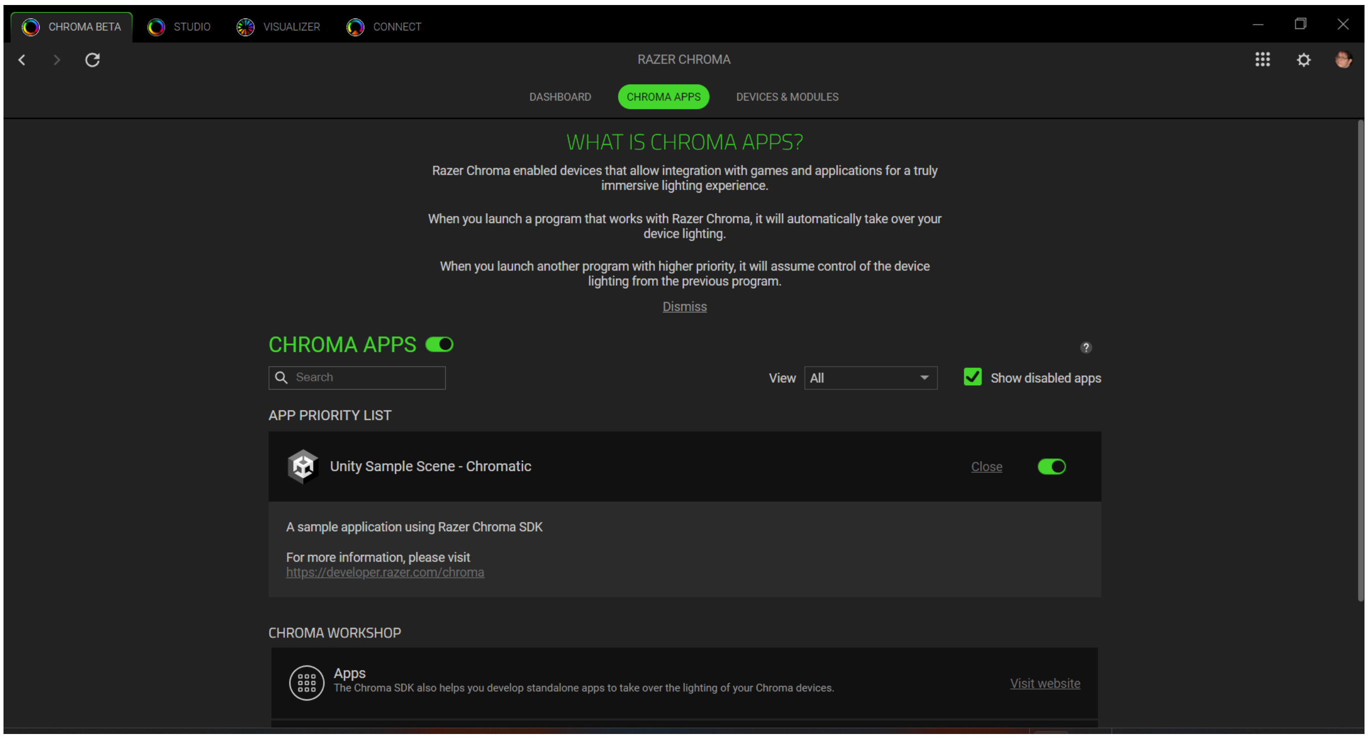
Namespace

Add the Chroma SDK namespace to use the API.

```
using ChromaSDK;
```

Initialize SDK

Initialize the Chroma SDK in order to utilize the API. The `InitSDK` method takes an `AppInfo` parameter which defines the application or game details that will appear in the `Chroma App` within the `Chroma Apps` tab. The expected return result should be `RazerErrors.RZRESULT_SUCCESS` which indicates the API is ready for use. If a non-success result is returned, the Chroma implementation should be disabled until the next time the application or game is launched. Reasons for failure are likely to be the user does not have the `Synapse` or the `Chroma App` installed. After successfully initializing the Chroma SDK, wait approximately 100 ms before playing Chroma animations.



WHAT IS CHROMA APPS?

Razer Chroma enabled devices that allow integration with games and applications for a truly immersive lighting experience.

When you launch a program that works with Razer Chroma, it will automatically take over your device lighting.

When you launch another program with higher priority, it will assume control of the device lighting from the previous program.

APP PRIORITY LIST

Unity Sample Scene - Chromatic

A sample application using Razer Chroma SDK

For more information, please visit https://developer.razer.com/chroma

CHROMA WORKSHOP

Apps
The Chroma SDK also helps you develop standalone apps to take over the lighting of your Chroma devices.

```
APPINFOTYPE appInfo = new APPINFOTYPE();
appInfo.Title = "Unity Sample Scene - Chromatic";
appInfo.Description = "A sample application using Razer Chroma SDK";

appInfo.Author_Name = "Razer";
appInfo.Author_Contact = "https://developer.razer.com/chroma";
```

```
//appInfo.SupportedDevice = 
//    0x01 | // Keyboards
//    0x02 | // Mice
//    0x04 | // Headset
//    0x08 | // Mousepads
//    0x10 | // Keypads
//    0x20   // ChromaLink devices
appInfo.SupportedDevice = (0x01 | 0x02 | 0x04 | 0x08 | 0x10 | 0x20);
//    0x01 | // Utility. (To specifiy this is an utility application)
//    0x02   // Game. (To specifiy this is a game);
appInfo.Category = 1;

int result = ChromaAnimationAPI.InitSDK(ref appInfo);
if (result == RazerErrors.RZRESULT_SUCCESS)
{
    // Init Success! Ready to use the Chroma SDK!
}
else
{
    // Init Failed! Stop using the Chroma SDK until the next game launch!";
}
```

Applications should uninitialize the Chroma SDK with Uninit() for a clean exit. Uninitialization is only needed if the Chroma SDK was successfully initialized.

```
int result = ChromaAnimationAPI::Uninit();
if (result == RazerErrors.RZRESULT_SUCCESS)
{
    // Chroma has been uninitialized!
}
else
{
    // Uninitialization was unsuccessful!
}
```

Is Active

Many applications and games can use the Chroma SDK at the same time, yet only one can have the Chroma focus. The `APP PRIORITY LIST` defines the priority order and the highest on the list receives the Chroma focus when more than one are actively using the Chroma SDK. Users can adjust the priority order by dragging and dropping or toggling the app completely off. The IsActive() method allows an application or game to check if it has Chroma focus. This allows the title to free up overhead when Chroma is not in use. If a title uses this to check for focus, the state should be periodically checked to turn Chroma back on when focus is returned. When active returns false, the title can stop playing Chroma animations, disable idle animations, and inactivate dynamic Chroma to free up some overhead. Keep in mind that some apps use Chroma notifications so they will only briefly take Chroma focus and then return it typically over a 5 second period.

```
bool isActive;
int result = ChromaAnimationAPI.CoreIsActive(out isActive);
if (result == RazerErrors.RZRESULT_SUCCESS)
{
    if (isActive)
    {
        // The game currently has the Chroma focus!
    }
    else
    {
        // The game does not currently have the Chroma focus!"
    }
}
else
{
    // Unable to check for Chroma focus. Unexpected result!
}
```

Is Connected

To further reduce overhead, a title can check if supported devices are connected before showing Chroma effects. The IsConnected() method can indicate if supported devices are in use to help determine if Chroma should be active. Games often will include a menu settings option to toggle Chroma RGB support, with being on by default as an additional way that users can minimize overhead.

```
DEVICE_INFO_TYPE deviceInfo = new DEVICE_INFO_TYPE();
deviceInfo.DeviceType = 255; // all devices
int result = ChromaAnimationAPI.CoreIsConnected(ref deviceInfo);
if (result == RazerErrors.RZRESULT_SUCCESS)
{
    if (deviceInfo.Connected > 0)
    {
        // Chroma devices are connected!
    }
    else
    {
        // "No Chroma devices are connected!";
    }
}
else
{
    // "Unable to check for Chroma devices. Unexpected result!";
}
```

Play Chroma Animation

The Chroma SDK supports playing premade Chroma animations which are placed in the `StreamingAssets` folder or subfolders within. Chroma animations can be created in the web authoring tools, or dynamically created and modified using the API. Call PlayAnimation() to play Chroma animations with or without looping. Animations have a device category, and playing an animation will stop an existing animation from playing before playing the new animation for the given device category. The animation name is file path of the Chroma animation relative to the `StreamingAssets` folder.

```
bool loop = false;
string[] devices =
{
    "ChromaLink",
    "Headset",
    "Keyboard",
    "Keypad",
    "Mouse",
    "Mousepad"
};
foreach (string device in devices)
{
    // The Chroma animation files are located in the subfolder at:
    // Assets/StreamingAssets/Animations/
    string animationName = string.Format("Animations/Spiral_{0}.chroma", device);
    ChromaAnimationAPI.PlayAnimationName(animationName, loop);
}
```

Set Event Name

Chroma events can be named to add supplemental technology to your lighting experience. By naming game events and game triggers, the event name can be used as a lookup to play things like haptics effects. `Jump_2s` could be used when playing a Chroma animation of a jump effect that lasts for 2 seconds. Using "Jump_2s" a corresponding haptic effect with similar duration can be added with the Chroma effect to enhance emersion for the title. No other APIs are required to add haptics effects other than to invoke SetEventtName(). To stop haptics playback use SetEventName() with an empty string. A Chroma animation does not need to be playing in order to trigger haptics manually with SetEventName().

```
int result = ChromaAnimationAPI.CoreSetEventName("Jump_2s");
if (result == RazerErrors.RZRESULT_SUCCESS)
{
    // Chroma event named successfully!"
}
else
{
    // Unable to set event name. Unexpected result!"
}

// Stop haptic playback
result = ChromaAnimationAPI::CoreSetEventName("");
if (result == RazerErrors.RZRESULT_SUCCESS)
{
    // Haptics stopped successfully!"
}
```

```
else
{
    // Unable to stop haptics. Unexpected result!"
}
```

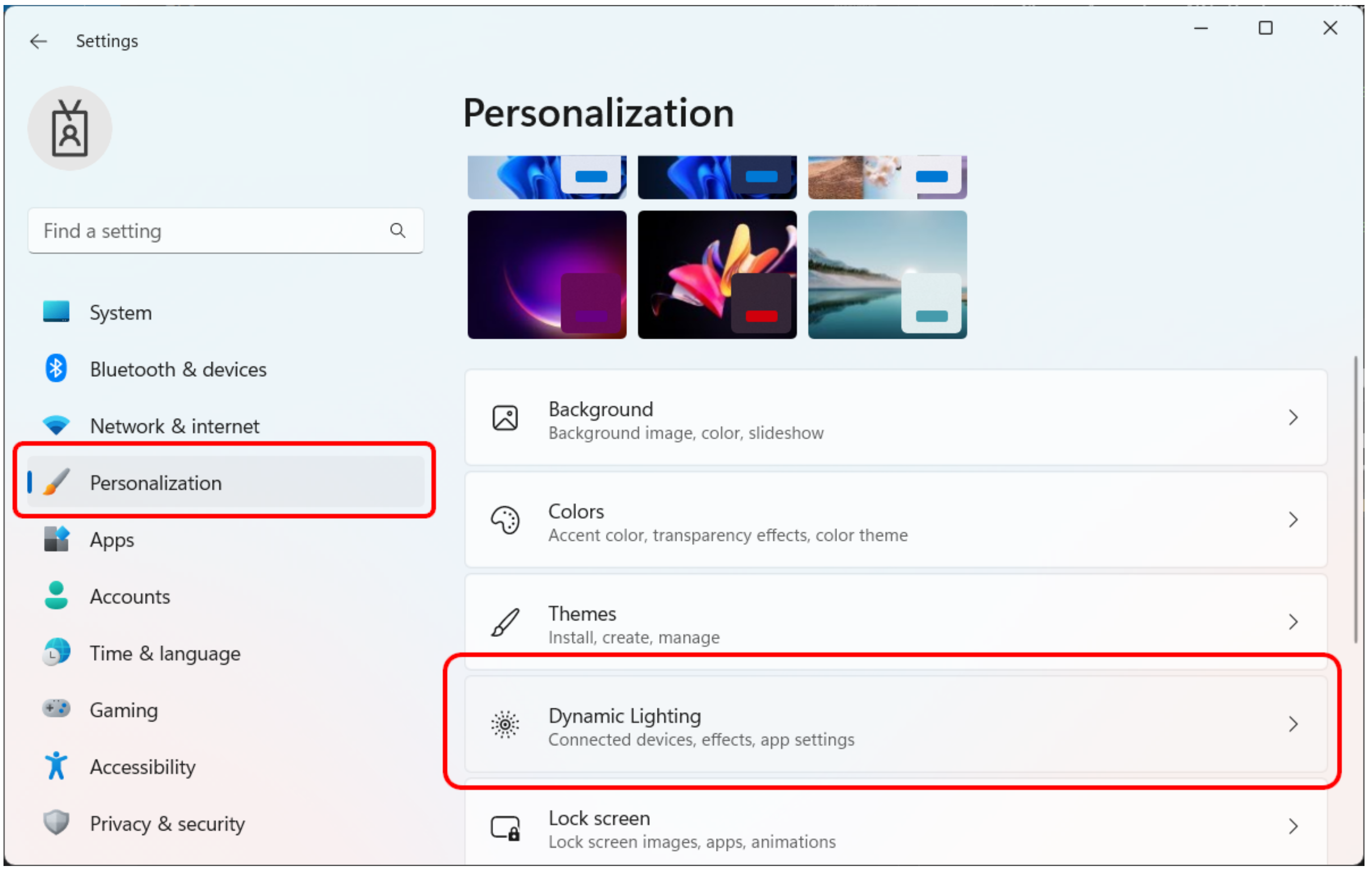
Use Forward Chroma Events

By default when PlayAnimation is called, the animation name is automatically sent to SetEventName(). In order to disable the default behaviour set the toggle to false. PlayAnimation() as shown above is called for each device category. It will be more efficent to use SetEventName() once for the Chroma animation set. Manual mode gives the title explicit control over when SetEventName() is called.

```
bool toggle = false; // manual mode
ChromaAnimationAPI.UseForwardChromaEvents(toggle);
if (toggle)
{
    // When PlayAnimation is used, the name is sent to SetEventName().
}
else
{
    // The PlayAnimation name is not forwarded.
}
```

Microsoft Dynamic Lighting

Windows 11 launched Microsoft Dynamic Lighting which is built-in to the Windows Settings Personalization on Windows. Microsoft DL became generally available in `Windows 11 22H2`. See the list of supported devices.



Personalization

Background
Background image, color, slideshow

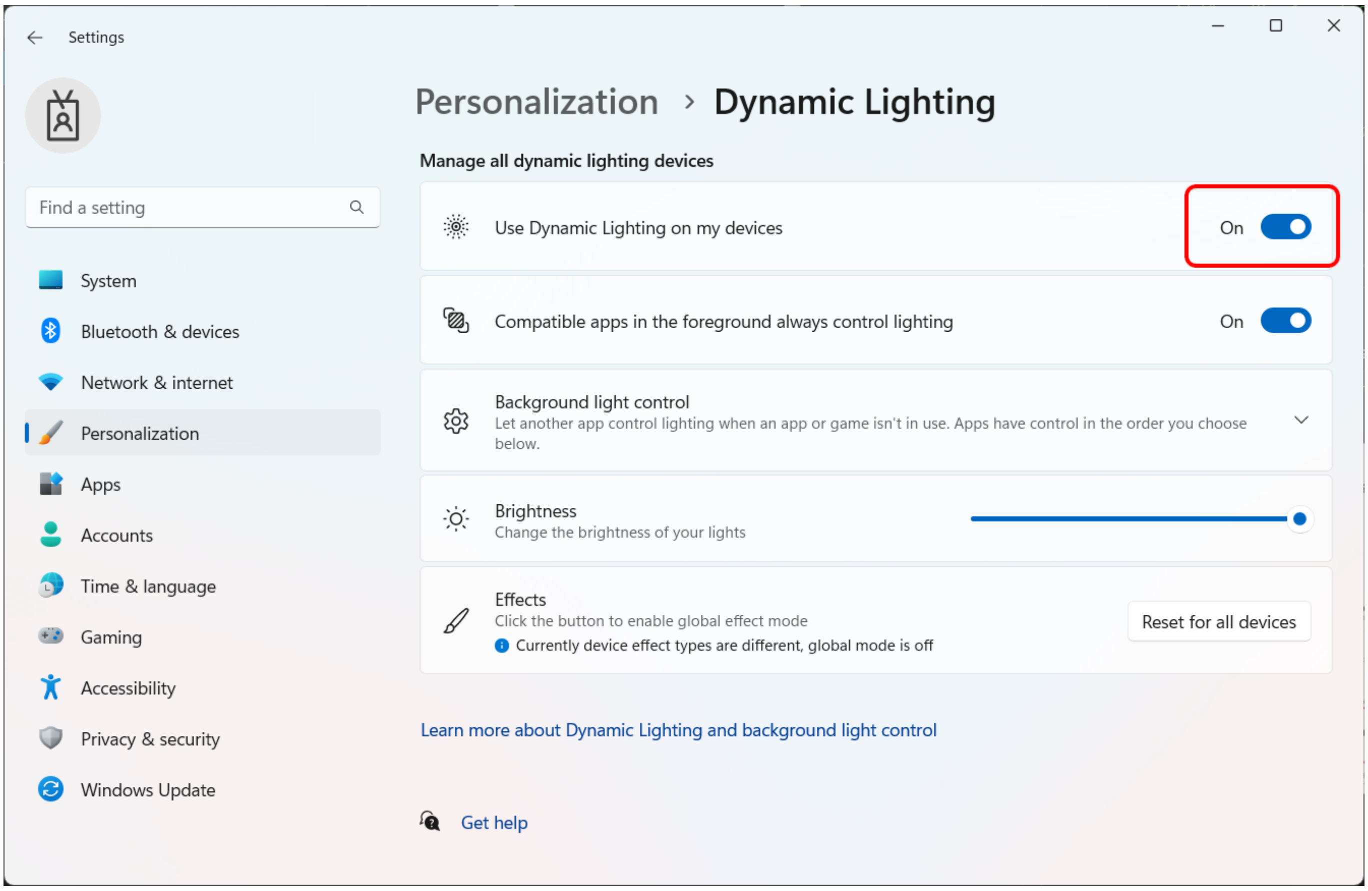
Colors
Accent color, transparency effects, color theme

Themes
Install, create, manage

Dynamic Lighting
Connected devices, effects, app settings

Lock screen
Lock screen images, apps, animations

For HID compatible devices, with `Dynamic Lighting` set to `ON` and `Chroma App` set as the ambient controller, Chroma effects will display on DL compatible hardware. No extra coding is required to add this compatibility. `Chroma App` handles Chroma compatibility with DL and it is completely automatic.



Personalization > Dynamic Lighting

Manage all dynamic lighting devices

Use Dynamic Lighting on my devices: On

Compatible apps in the foreground always control lighting: On

Background light control
Let another app control lighting when an app or game isn't in use. Apps have control in the order you choose below.

Brightness
Change the brightness of your lights

Effects
Click the button to enable global effect mode
Currently device effect types are different, global mode is off
Reset for all devices

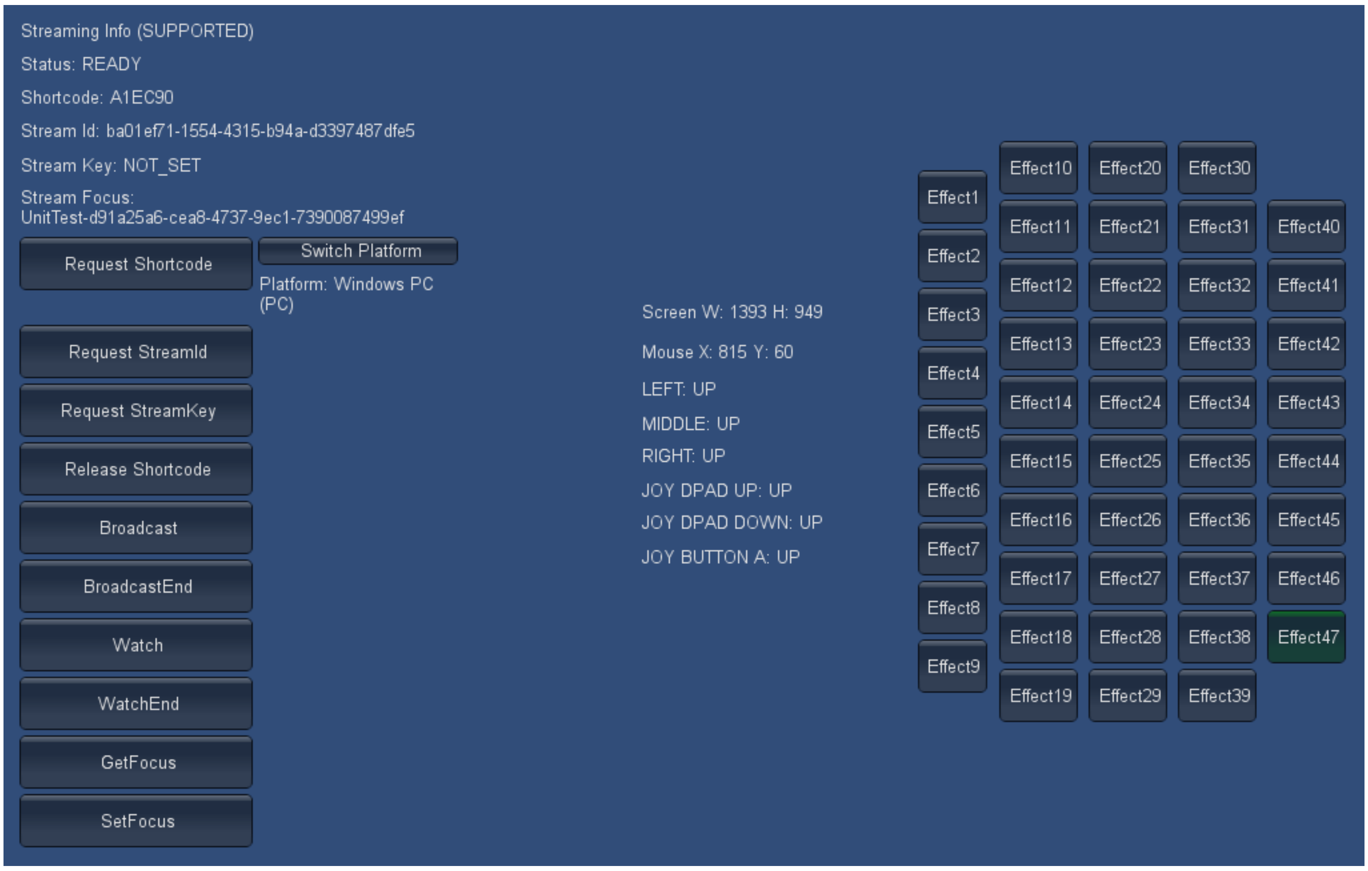
Learn more about Dynamic Lighting and background light control

Get help

About

The `Assets/Scenes/Scene_SampleApp.unity` scene shows the sample animations from the Chroma Animation Guide. The referenced sample script can be found at `Packages\RazerChromaSDK\Tests\Scripts\SampleApp.cs`.

Screenshot:

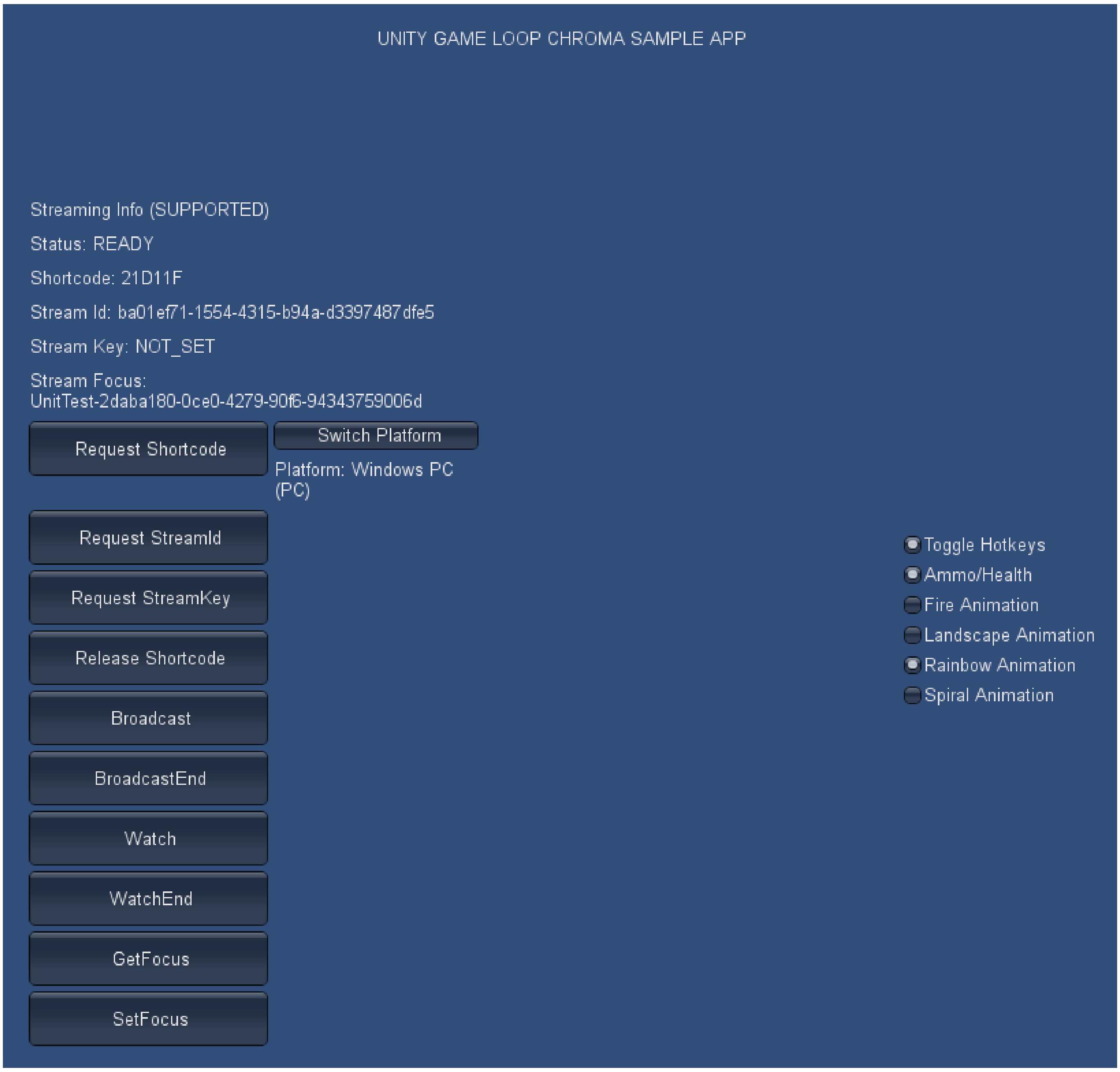


Streaming Info (SUPPORTED)
Status: READY
Shortcode: A1EC90
Stream Id: ba01ef71-1554-4315-b94a-d3397487dfe5
Stream Key: NOT_SET
Stream Focus: UnitTest-d91a25a6-cea8-4737-9ec1-7390087499ef
Platform: Windows PC (PC)

Screen W: 1393 H: 949
Mouse X: 815 Y: 60
LEFT: UP
MIDDLE: UP
RIGHT: UP
JOY DPAD UP: UP
JOY DPAD DOWN: UP
JOY BUTTON A: UP

The `Assets/Scenes/Scene_SampleGameLoop.unity` scene shows how to dynamically set color effects directly through the API and while also playing several animations at the same time using various blending operations. This sample shows how to do Chroma effects without using premade Chroma animations. Chroma animations can be used as source color information when doing dynamic blending. The referenced sample script can be found at `Packages\RazerChromaSDK\Tests\Scripts\SampleGameLoop.cs`.

Screenshot:


UNITY GAME LOOP CHROMA SAMPLE APP

Streaming Info (SUPPORTED)

Status: READY

Shortcode: 21D11F

Stream Id: ba01ef71-1554-4315-b94a-d3397487dfe5

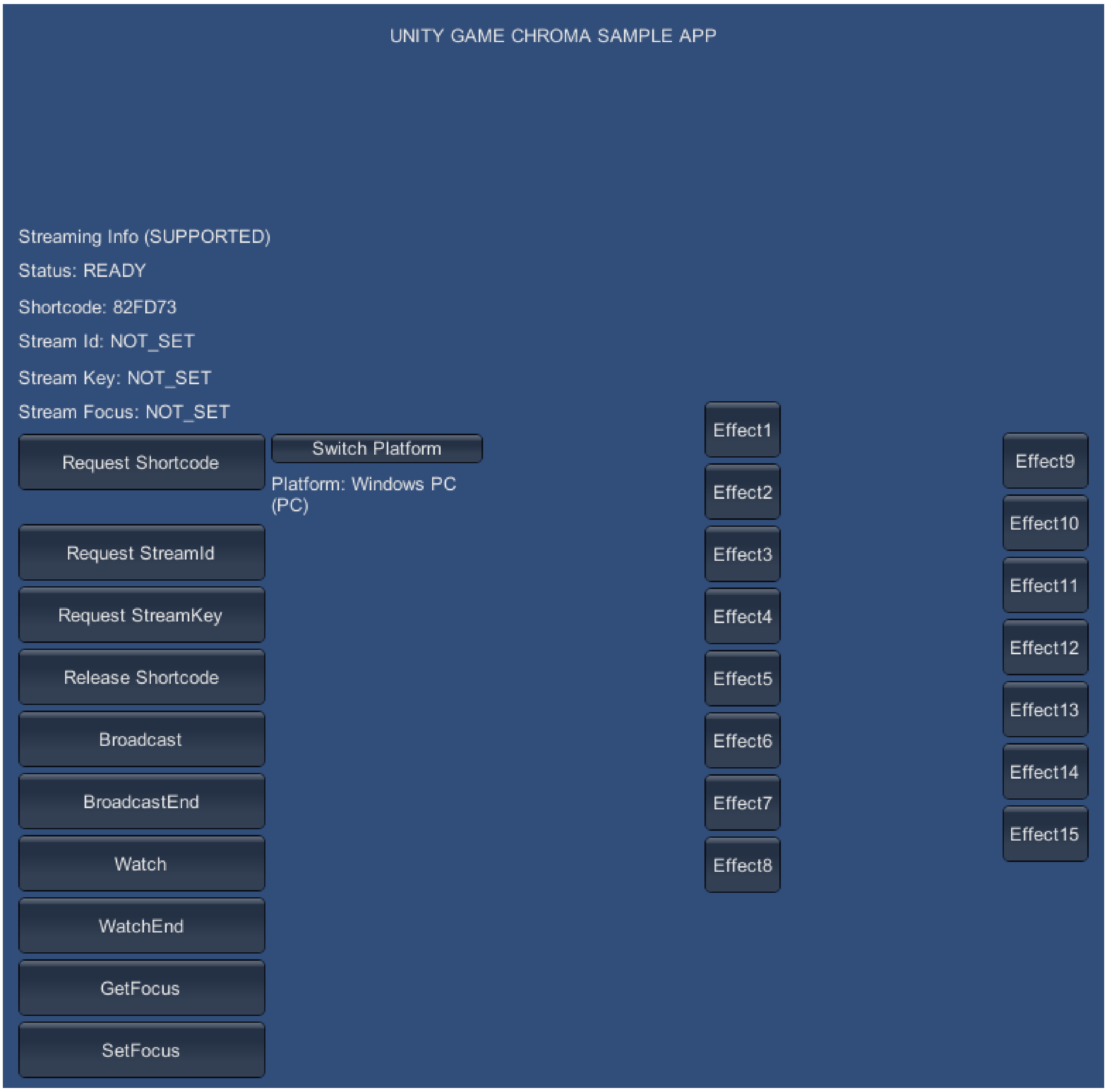
Stream Key: NOT_SET

Stream Focus:
UnitTest-2daba180-0ce0-4279-90f6-94343759006d

Platform: Windows PC (PC)

The `Assets/Scenes/Scene_GameSample.unity` scene is a template intended to work with the automated Chroma Design Converter for quickly porting sample effects from HTML5 to Unity. The referenced sample script can be found at `Packages\RazerChromaSDK\Tests\Scripts\GameSample.cs`. Chroma Design samples are commonly created with 15 sample effects which is why the template has that many buttons to play the sample effects from the ported code. The Chroma Design Converter is not limited to just 15 sample effects and can generate more effect code from the input HTML5 script.

Screenshot:


UNITY GAME CHROMA SAMPLE APP

Streaming Info (SUPPORTED)
Status: READY
Shortcode: 82FD73
Stream Id: NOT_SET
Stream Key: NOT_SET
Stream Focus: NOT_SET

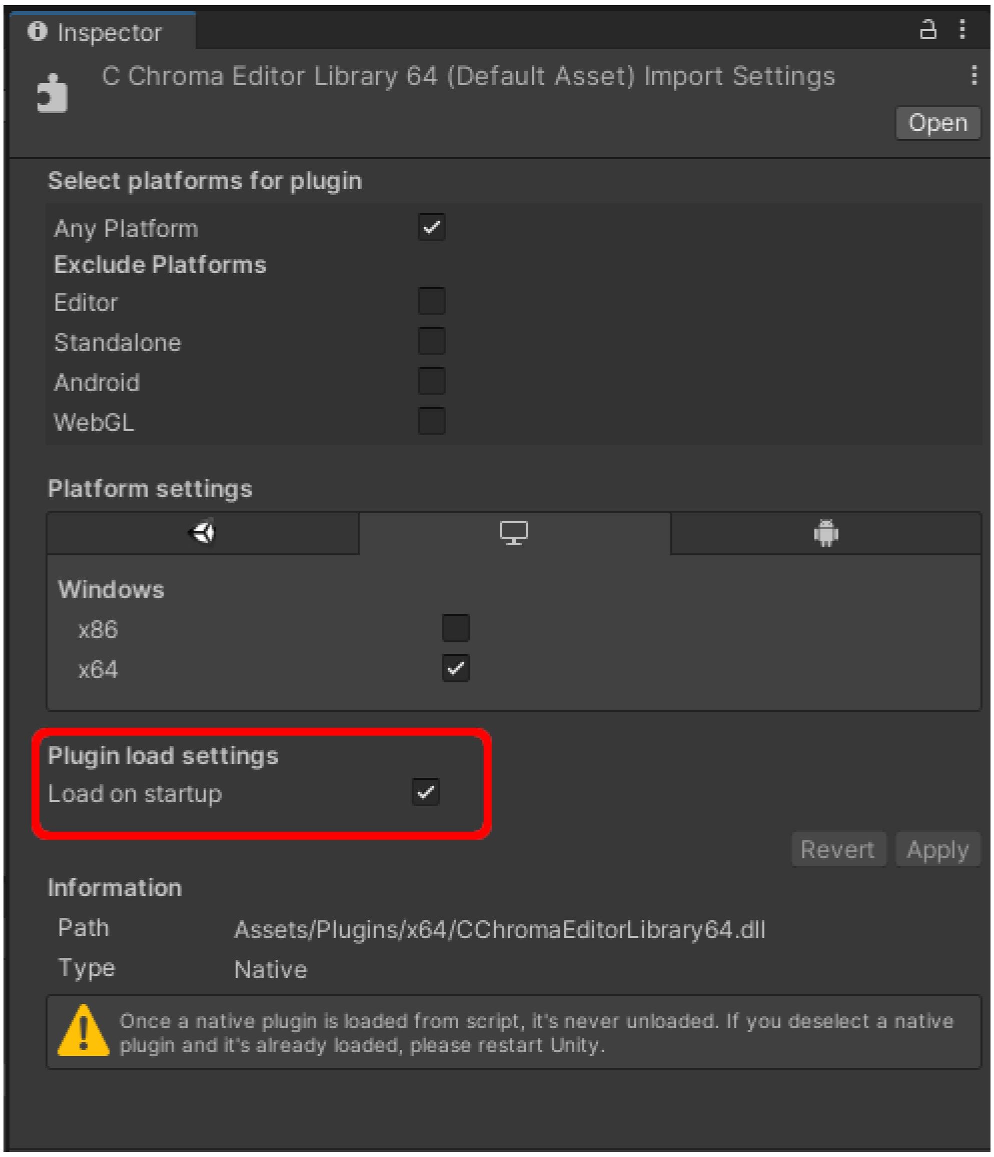
Platform: Windows PC (PC)

Chroma Editor Library

The `Chroma Editor Library` is a helper library for Chroma animation playback and realtime manipulation of Chroma animations.

The latest versions of the `Chroma Editor Library` can be found in Releases for `Windows-PC` and `Windows-Cloud`.

With recent Unity versions, be sure to inspect the native plugins and enable `Load on startup`. `Editor` and `Standalone` platforms are supported for 32-bit and 64-bit.



C Chroma Editor Library 64 (Default Asset) Import Settings

Select platforms for plugin

Any Platform ✓

Exclude Platforms

Editor

Standalone

Android

WebGL

Platform settings

Windows

x86

x64 ✓

Plugin load settings

Load on startup ✓

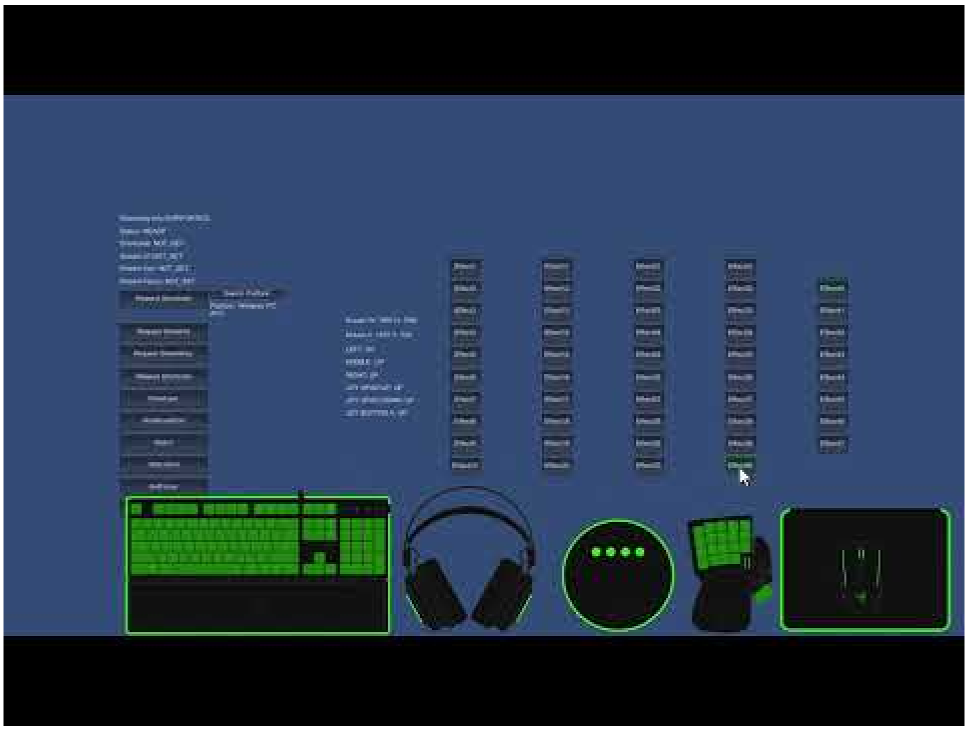
Information

Path Assets/Plugins/x64/CChromaEditorLibrary64.dll

Type Native

Once a native plugin is loaded from script, it's never unloaded. If you deselect a native plugin and it's already loaded, please restart Unity.

Video: **Unity Chroma Animation Sample App - Streaming on Windows PC and Cloud**



API Class

The `ChromaAnimationAPI` class provides a wrapper for the Chroma Editor Library. The wrapper for the API can be found at `Packages\RazerChromaSDK\Runtime\ChromaAnimationAPI.cs`.

Full API

- AddColor
- AddFrame
- AddNonZeroAllKeys
- AddNonZeroAllKeysAllFrames
- AddNonZeroAllKeysAllFramesName
- AddNonZeroAllKeysAllFramesNameD
- AddNonZeroAllKeysAllFramesOffset
- AddNonZeroAllKeysAllFramesOffsetName
- AddNonZeroAllKeysAllFramesOffsetNameD
- AddNonZeroAllKeysName
- AddNonZeroAllKeysOffset
- AddNonZeroAllKeysOffsetName
- AddNonZeroAllKeysOffsetNameD
- AddNonZeroTargetAllKeysAllFrames
- AddNonZeroTargetAllKeysAllFramesName
- AddNonZeroTargetAllKeysAllFramesNameD
- AddNonZeroTargetAllKeysAllFramesOffset
- AddNonZeroTargetAllKeysAllFramesOffsetName
- AddNonZeroTargetAllKeysAllFramesOffsetNameD
- AddNonZeroTargetAllKeysOffset
- AddNonZeroTargetAllKeysOffsetName
- AddNonZeroTargetAllKeysOffsetNameD
- AppendAllFrames
- AppendAllFramesName
- AppendAllFramesNameD
- ClearAll
- ClearAnimationType
- CloseAll
- CloseAnimation
- CloseAnimationD
- CloseAnimationName
- CloseAnimationNameD
- CloseComposite
- CloseCompositeD
- CopyAllKeys
- CopyAllKeysName
- CopyAnimation
- CopyAnimationName
- CopyAnimationNameD
- CopyBlueChannelAllFrames

- CopyBlueChannelAllFramesName
- CopyBlueChannelAllFramesNameD
- CopyGreenChannelAllFrames
- CopyGreenChannelAllFramesName
- CopyGreenChannelAllFramesNameD
- CopyKeyColor
- CopyKeyColorAllFrames
- CopyKeyColorAllFramesName
- CopyKeyColorAllFramesNameD
- CopyKeyColorAllFramesOffset
- CopyKeyColorAllFramesOffsetName
- CopyKeyColorAllFramesOffsetNameD
- CopyKeyColorName
- CopyKeyColorNameD
- CopyKeysColor
- CopyKeysColorAllFrames
- CopyKeysColorAllFramesName
- CopyKeysColorName
- CopyKeysColorOffset
- CopyKeysColorOffsetName
- CopyNonZeroAllKeys
- CopyNonZeroAllKeysAllFrames
- CopyNonZeroAllKeysAllFramesName
- CopyNonZeroAllKeysAllFramesNameD
- CopyNonZeroAllKeysAllFramesOffset
- CopyNonZeroAllKeysAllFramesOffsetName
- CopyNonZeroAllKeysAllFramesOffsetNameD
- CopyNonZeroAllKeysName
- CopyNonZeroAllKeysNameD
- CopyNonZeroAllKeysOffset
- CopyNonZeroAllKeysOffsetName
- CopyNonZeroAllKeysOffsetNameD
- CopyNonZeroKeyColor
- CopyNonZeroKeyColorName
- CopyNonZeroKeyColorNameD
- CopyNonZeroTargetAllKeys
- CopyNonZeroTargetAllKeysAllFrames
- CopyNonZeroTargetAllKeysAllFramesName
- CopyNonZeroTargetAllKeysAllFramesNameD
- CopyNonZeroTargetAllKeysAllFramesOffset
- CopyNonZeroTargetAllKeysAllFramesOffsetName
- CopyNonZeroTargetAllKeysAllFramesOffsetNameD
- CopyNonZeroTargetAllKeysName
- CopyNonZeroTargetAllKeysNameD
- CopyNonZeroTargetAllKeysOffset
- CopyNonZeroTargetAllKeysOffsetName

- FillNonZeroColorName
- FillNonZeroColorNameD
- FillNonZeroColorRGB
- FillNonZeroColorRGBName
- FillNonZeroColorRGBNameD
- FillRandomColors
- FillRandomColorsAllFrames
- FillRandomColorsAllFramesName
- FillRandomColorsAllFramesNameD
- FillRandomColorsBlackAndWhite
- FillRandomColorsBlackAndWhiteAllFrames
- FillRandomColorsBlackAndWhiteAllFramesName
- FillRandomColorsBlackAndWhiteAllFramesNameD
- FillRandomColorsBlackAndWhiteName
- FillRandomColorsBlackAndWhiteNameD
- FillRandomColorsName
- FillRandomColorsNameD
- FillThresholdColors
- FillThresholdColorsAllFrames
- FillThresholdColorsAllFramesName
- FillThresholdColorsAllFramesNameD
- FillThresholdColorsAllFramesRGB
- FillThresholdColorsAllFramesRGBName
- FillThresholdColorsAllFramesRGBNameD
- FillThresholdColorsMinMaxAllFramesRGB
- FillThresholdColorsMinMaxAllFramesRGBName
- FillThresholdColorsMinMaxAllFramesRGBNameD
- FillThresholdColorsMinMaxRGB
- FillThresholdColorsMinMaxRGBName
- FillThresholdColorsMinMaxRGBNameD
- FillThresholdColorsName
- FillThresholdColorsNameD
- FillThresholdColorsRGB
- FillThresholdColorsRGBName
- FillThresholdColorsRGBNameD
- FillThresholdRGBColorsAllFramesRGB
- FillThresholdRGBColorsAllFramesRGBName
- FillThresholdRGBColorsAllFramesRGBNameD
- FillThresholdRGBColorsRGB
- FillThresholdRGBColorsRGBName
- FillThresholdRGBColorsRGBNameD
- FillZeroColor
- FillZeroColorAllFrames
- FillZeroColorAllFramesName
- FillZeroColorAllFramesNameD
- FillZeroColorAllFramesRGB

- GetPlayingAnimationId
- GetRGB
- GetRGBD
- GetTotalDuration
- GetTotalDurationName
- HasAnimationLoop
- HasAnimationLoopName
- HasAnimationLoopNameD
- Init
- InitD
- InitSDK
- InsertDelay
- InsertDelayName
- InsertDelayNameD
- InsertFrame
- InsertFrameName
- InsertFrameNameD
- InvertColors
- InvertColorsAllFrames
- InvertColorsAllFramesName
- InvertColorsAllFramesNameD
- InvertColorsName
- InvertColorsNameD
- IsAnimationPaused
- IsAnimationPausedName
- IsAnimationPausedNameD
- IsDialogOpen
- IsDialogOpenD
- IsInitialized
- IsInitializedD
- IsPlatformSupported
- IsPlatformSupportedD
- IsPlaying
- IsPlayingD
- IsPlayingName
- IsPlayingNameD
- IsPlayingType
- IsPlayingTypeD
- Lerp
- LerpColor
- LoadAnimation
- LoadAnimationD
- LoadAnimationName
- LoadComposite
- MakeBlankFrames
- MakeBlankFramesName

AddColor

Return the sum of colors

```
int result = ChromaAnimationAPI.AddColor(int color1, int color2);
```

AddFrame

Adds a frame to the `Chroma` animation and sets the `duration` (in seconds). The `color` is expected to be an array of the dimensions for the `deviceType/device`. The `length` parameter is the size of the `color` array. For `EChromaSDKDevice1DEnum` the array size should be `MAX LEDS`. For `EChromaSDKDevice2DEnum` the array size should be `MAX ROW` times `MAX COLUMN`. Returns the animation id upon success. Returns negative one upon failure.

```
int result = ChromaAnimationAPI.AddFrame(int animationId, float duration, int[]
colors, int length);
```

AddNonZeroAllKeys

Add source color to target where color is not black for frame id, reference source and target by id.

```
ChromaAnimationAPI.AddNonZeroAllKeys(int sourceAnimationId, int targetAnimationId,
int frameId);
```

AddNonZeroAllKeysAllFrames

Add source color to target where color is not black for all frames, reference source and target by id.

```
ChromaAnimationAPI.AddNonZeroAllKeysAllFrames(int sourceAnimationId, int
targetAnimationId);
```

AddNonZeroAllKeysAllFramesName

Add source color to target where color is not black for all frames, reference source and target by name.

```
ChromaAnimationAPI.AddNonZeroAllKeysAllFramesName(string sourceAnimation, string
targetAnimation);
```

AddNonZeroAllKeysAllFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.AddNonZeroAllKeysAllFramesNameD(string
sourceAnimation, string targetAnimation);
```

AddNonZeroAllKeysAllFramesOffset

Add source color to target where color is not black for all frames starting at offset for the length of the source, reference source and target by id.

```
ChromaAnimationAPI.AddNonZeroAllKeysAllFramesOffset(int sourceAnimationId, int
targetAnimationId, int offset);
```

AddNonZeroAllKeysAllFramesOffsetName

Add source color to target where color is not black for all frames starting at offset for the length of the source, reference source and target by name.

```
ChromaAnimationAPI.AddNonZeroAllKeysAllFramesOffsetName(string sourceAnimation,
string targetAnimation, int offset);
```

AddNonZeroAllKeysAllFramesOffsetNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.AddNonZeroAllKeysAllFramesOffsetNameD(string
sourceAnimation, string targetAnimation, double offset);
```

AddNonZeroAllKeysName

Add source color to target where color is not black for frame id, reference source and target by name.

```
ChromaAnimationAPI.AddNonZeroAllKeysName(string sourceAnimation, string
targetAnimation, int frameId);
```

AddNonZeroAllKeysOffset

Add source color to target where color is not black for the source frame and target offset frame, reference source and target by id.

```
ChromaAnimationAPI.AddNonZeroAllKeysOffset(int sourceAnimationId, int
targetAnimationId, int frameId, int offset);
```

AddNonZeroAllKeysOffsetName

Add source color to target where color is not black for the source frame and target offset frame, reference source and target by name.

```
ChromaAnimationAPI.AddNonZeroAllKeysOffsetName(string sourceAnimation, string
targetAnimation, int frameId, int offset);
```

AddNonZeroAllKeysOffsetNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.AddNonZeroAllKeysOffsetNameD(string
sourceAnimation, string targetAnimation, double frameId, double offset);
```

AddNonZeroTargetAllKeysAllFrames

Add source color to target where the target color is not black for all frames, reference source and target by id.

```
ChromaAnimationAPI.AddNonZeroTargetAllKeysAllFrames(int sourceAnimationId, int
targetAnimationId);
```

AddNonZeroTargetAllKeysAllFramesName

Add source color to target where the target color is not black for all frames, reference source and target by name.

```
ChromaAnimationAPI.AddNonZeroTargetAllKeysAllFramesName(string sourceAnimation,
string targetAnimation);
```

AddNonZeroTargetAllKeysAllFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.AddNonZeroTargetAllKeysAllFramesNameD(string
sourceAnimation, string targetAnimation);
```

AddNonZeroTargetAllKeysAllFramesOffset

Add source color to target where the target color is not black for all frames starting at offset for the length of the source, reference source and target by id.

```
ChromaAnimationAPI.AddNonZeroTargetAllKeysAllFramesOffset(int sourceAnimationId,
int targetAnimationId, int offset);
```

AddNonZeroTargetAllKeysAllFramesOffsetName

Add source color to target where the target color is not black for all frames starting at offset for the length of the source, reference source and target by name.

```
ChromaAnimationAPI.AddNonZeroTargetAllKeysAllFramesOffsetName(string
sourceAnimation, string targetAnimation, int offset);
```

AddNonZeroTargetAllKeysAllFramesOffsetNameD

D suffix for limited data types.

```
double result =
ChromaAnimationAPI.AddNonZeroTargetAllKeysAllFramesOffsetNameD(string
sourceAnimation, string targetAnimation, double offset);
```

AddNonZeroTargetAllKeysOffset

Add source color to target where target color is not blank from the source frame to the target offset frame, reference source and target by id.

```
ChromaAnimationAPI.AddNonZeroTargetAllKeysOffset(int sourceAnimationId, int
targetAnimationId, int frameId, int offset);
```

AddNonZeroTargetAllKeysOffsetName

Add source color to target where target color is not blank from the source frame to the target offset frame, reference source and target by name.

```
ChromaAnimationAPI.AddNonZeroTargetAllKeysOffsetName(string sourceAnimation,
string targetAnimation, int frameId, int offset);
```

AddNonZeroTargetAllKeysOffsetNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.AddNonZeroTargetAllKeysOffsetNameD(string
sourceAnimation, string targetAnimation, double frameId, double offset);
```

AppendAllFrames

Append all source frames to the target animation, reference source and target by id.

```
ChromaAnimationAPI.AppendAllFrames(int sourceAnimationId, int targetAnimationId);
```

AppendAllFramesName

Append all source frames to the target animation, reference source and target by name.

```
ChromaAnimationAPI.AppendAllFramesName(string sourceAnimation, string
targetAnimation);
```

AppendAllFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.AppendAllFramesNameD(string sourceAnimation,
string targetAnimation);
```

ClearAll

`PluginClearAll` will issue a `CLEAR` effect for all devices.

```
ChromaAnimationAPI.ClearAll();
```

ClearAnimationType

`PluginClearAnimationType` will issue a `CLEAR` effect for the given device.

```
ChromaAnimationAPI.ClearAnimationType(int deviceType, int device);
```

CloseAll

`PluginCloseAll` closes all open animations so they can be reloaded from disk. The set of animations will be stopped if playing.

```
ChromaAnimationAPI.CloseAll();
```

CloseAnimation

Closes the `Chroma` animation to free up resources referenced by id. Returns the animation id upon success. Returns negative one upon failure. This might be used while authoring effects if there was a change necessitating re-opening the animation. The animation id can no longer be used once closed.

```
int result = ChromaAnimationAPI.CloseAnimation(int animationId);
```

CloseAnimationD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.CloseAnimationD(double animationId);
```

CloseAnimationName

Closes the `Chroma` animation referenced by name so that the animation can be reloaded from disk.

```
ChromaAnimationAPI.CloseAnimationName(string path);
```

CloseAnimationNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.CloseAnimationNameD(string path);
```

CloseComposite

`PluginCloseComposite` closes a set of animations so they can be reloaded from disk. The set of animations will be stopped if playing.

```
ChromaAnimationAPI.CloseComposite(string name);
```

CloseCompositeD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.CloseCompositeD(string name);
```

CopyAllKeys

Copy source animation to target animation for the given frame. Source and target are referenced by id.

```
ChromaAnimationAPI.CopyAllKeys(int sourceAnimationId, int targetAnimationId, int
frameId);
```

CopyAllKeysName

Copy source animation to target animation for the given frame. Source and target are referenced by id.

```
ChromaAnimationAPI.CopyAllKeysName(string sourceAnimation, string targetAnimation,
int frameId);
```

CopyAnimation

Copy animation to named target animation in memory. If target animation exists, close first. Source is referenced by id.

```
int result = ChromaAnimationAPI.CopyAnimation(int sourceAnimationId, string
targetAnimation);
```

CopyAnimationName

Copy animation to named target animation in memory. If target animation exists, close first. Source is referenced by name.

```
ChromaAnimationAPI.CopyAnimationName(string sourceAnimation, string
targetAnimation);
```

CopyAnimationNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.CopyAnimationNameD(string sourceAnimation,
string targetAnimation);
```

CopyBlueChannelAllFrames

Copy blue channel to other channels for all frames. Intensity range is 0.0 to 1.0. Reference the animation by id.

```
ChromaAnimationAPI.CopyBlueChannelAllFrames(int animationId, float redIntensity,
float greenIntensity);
```

CopyBlueChannelAllFramesName

Copy blue channel to other channels for all frames. Intensity range is 0.0 to 1.0. Reference the animation by name.

```
ChromaAnimationAPI.CopyBlueChannelAllFramesName(string path, float redIntensity,
float greenIntensity);
```

CopyBlueChannelAllFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.CopyBlueChannelAllFramesNameD(string path,
double redIntensity, double greenIntensity);
```

CopyGreenChannelAllFrames

Copy green channel to other channels for all frames. Intensity range is 0.0 to 1.0. Reference the animation by id.

```
ChromaAnimationAPI.CopyGreenChannelAllFrames(int animationId, float redIntensity,
float blueIntensity);
```

CopyGreenChannelAllFramesName

Copy green channel to other channels for all frames. Intensity range is 0.0 to 1.0. Reference the animation by name.

```
ChromaAnimationAPI.CopyGreenChannelAllFramesName(string path, float redIntensity,
float blueIntensity);
```

CopyGreenChannelAllFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.CopyGreenChannelAllFramesNameD(string path,
double redIntensity, double blueIntensity);
```

CopyKeyColor

Copy animation key color from the source animation to the target animation for the given frame. Reference the source and target by id.

```
ChromaAnimationAPI.CopyKeyColor(int sourceAnimationId, int targetAnimationId, int
frameId, int rzkey);
```

CopyKeyColorAllFrames

Copy animation key color from the source animation to the target animation for all frames. Reference the source and target by id.

```
ChromaAnimationAPI.CopyKeyColorAllFrames(int sourceAnimationId, int
targetAnimationId, int rzkey);
```

CopyKeyColorAllFramesName

Copy animation key color from the source animation to the target animation for all frames. Reference the source and target by name.

```
ChromaAnimationAPI.CopyKeyColorAllFramesName(string sourceAnimation, string
targetAnimation, int rzkey);
```

CopyKeyColorAllFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.CopyKeyColorAllFramesNameD(string
sourceAnimation, string targetAnimation, double rzkey);
```

CopyKeyColorAllFramesOffset

Copy animation key color from the source animation to the target animation for all frames, starting at the offset for the length of the source animation. Source and target are referenced by id.

```
ChromaAnimationAPI.CopyKeyColorAllFramesOffset(int sourceAnimationId, int
targetAnimationId, int rzkey, int offset);
```

CopyKeyColorAllFramesOffsetName

Copy animation key color from the source animation to the target animation for all frames, starting at the offset for the length of the source animation. Source and target are referenced by name.

```
ChromaAnimationAPI.CopyKeyColorAllFramesOffsetName(string sourceAnimation, string
targetAnimation, int rzkey, int offset);
```

CopyKeyColorAllFramesOffsetNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.CopyKeyColorAllFramesOffsetNameD(string
sourceAnimation, string targetAnimation, double rzkey, double offset);
```

CopyKeyColorName

Copy animation key color from the source animation to the target animation for the given frame.

```
ChromaAnimationAPI.CopyKeyColorName(string sourceAnimation, string
targetAnimation, int frameId, int rzkey);
```

CopyKeyColorNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.CopyKeyColorNameD(string sourceAnimation,
string targetAnimation, double frameId, double rzkey);
```

CopyKeysColor

Copy animation color for a set of keys from the source animation to the target animation for the given frame. Reference the source and target by id.

```
ChromaAnimationAPI.CopyKeysColor(int sourceAnimationId, int targetAnimationId, int
frameId, int[] keys, int size);
```

CopyKeysColorAllFrames

Copy animation color for a set of keys from the source animation to the target animation for all frames. Reference the source and target by id.

```
ChromaAnimationAPI.CopyKeysColorAllFrames(int sourceAnimationId, int
targetAnimationId, int[] keys, int size);
```

CopyKeysColorAllFramesName

Copy animation color for a set of keys from the source animation to the target animation for all frames. Reference the source and target by name.

```
ChromaAnimationAPI.CopyKeysColorAllFramesName(string sourceAnimation, string
targetAnimation, int[] keys, int size);
```

CopyKeysColorName

Copy animation color for a set of keys from the source animation to the target animation for the given frame. Reference the source and target by name.

```
ChromaAnimationAPI.CopyKeysColorName(string sourceAnimation, string
targetAnimation, int frameId, int[] keys, int size);
```

CopyKeysColorOffset

Copy animation color for a set of keys from the source animation to the target animation from the source frame to the target frame. Reference the source and target by id.

```
ChromaAnimationAPI.CopyKeysColorOffset(int sourceAnimationId, int
targetAnimationId, int sourceFrameId, int targetFrameId, int[] keys, int size);
```

CopyKeysColorOffsetName

Copy animation color for a set of keys from the source animation to the target animation from the source frame to the target frame. Reference the source and target by name.

```
ChromaAnimationAPI.CopyKeysColorOffsetName(string sourceAnimation, string
targetAnimation, int sourceFrameId, int targetFrameId, int[] keys, int size);
```

CopyNonZeroAllKeys

Copy source animation to target animation for the given frame. Source and target are referenced by id.

```
ChromaAnimationAPI.CopyNonZeroAllKeys(int sourceAnimationId, int
targetAnimationId, int frameId);
```

CopyNonZeroAllKeysAllFrames

Copy nonzero colors from a source animation to a target animation for all frames. Reference source and target by id.

```
ChromaAnimationAPI.CopyNonZeroAllKeysAllFrames(int sourceAnimationId, int
targetAnimationId);
```

CopyNonZeroAllKeysAllFramesName

Copy nonzero colors from a source animation to a target animation for all frames. Reference source and target by name.

```
ChromaAnimationAPI.CopyNonZeroAllKeysAllFramesName(string sourceAnimation, string
targetAnimation);
```

CopyNonZeroAllKeysAllFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.CopyNonZeroAllKeysAllFramesNameD(string
sourceAnimation, string targetAnimation);
```

CopyNonZeroAllKeysAllFramesOffset

Copy nonzero colors from a source animation to a target animation for all frames starting at the offset for the length of the source animation. The source and target are referenced by id.

```
ChromaAnimationAPI.CopyNonZeroAllKeysAllFramesOffset(int sourceAnimationId, int
targetAnimationId, int offset);
```

CopyNonZeroAllKeysAllFramesOffsetName

Copy nonzero colors from a source animation to a target animation for all frames starting at the offset for the length of the source animation. The source and target are referenced by name.

```
ChromaAnimationAPI.CopyNonZeroAllKeysAllFramesOffsetName(string sourceAnimation,
string targetAnimation, int offset);
```

CopyNonZeroAllKeysAllFramesOffsetNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.CopyNonZeroAllKeysAllFramesOffsetNameD(string
sourceAnimation, string targetAnimation, double offset);
```

CopyNonZeroAllKeysName

Copy nonzero colors from source animation to target animation for the specified frame. Source and target are referenced by id.

```
ChromaAnimationAPI.CopyNonZeroAllKeysName(string sourceAnimation, string
targetAnimation, int frameId);
```

CopyNonZeroAllKeysNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.CopyNonZeroAllKeysNameD(string sourceAnimation,
string targetAnimation, double frameId);
```

CopyNonZeroAllKeysOffset

Copy nonzero colors from the source animation to the target animation from the source frame to the target offset frame. Source and target are referenced by id.

```
ChromaAnimationAPI.CopyNonZeroAllKeysOffset(int sourceAnimationId, int
targetAnimationId, int frameId, int offset);
```

CopyNonZeroAllKeysOffsetName

Copy nonzero colors from the source animation to the target animation from the source frame to the target offset frame. Source and target are referenced by name.

```
ChromaAnimationAPI.CopyNonZeroAllKeysOffsetName(string sourceAnimation, string
targetAnimation, int frameId, int offset);
```

CopyNonZeroAllKeysOffsetNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.CopyNonZeroAllKeysOffsetNameD(string
sourceAnimation, string targetAnimation, double frameId, double offset);
```

CopyNonZeroKeyColor

Copy animation key color from the source animation to the target animation for the given frame where color is not zero.

```
ChromaAnimationAPI.CopyNonZeroKeyColor(int sourceAnimationId, int
targetAnimationId, int frameId, int rzkey);
```

CopyNonZeroKeyColorName

Copy animation key color from the source animation to the target animation for the given frame where color is not zero.

```
ChromaAnimationAPI.CopyNonZeroKeyColorName(string sourceAnimation, string
targetAnimation, int frameId, int rzkey);
```

CopyNonZeroKeyColorNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.CopyNonZeroKeyColorNameD(string
sourceAnimation, string targetAnimation, double frameId, double rzkey);
```

CopyNonZeroTargetAllKeys

Copy nonzero colors from the source animation to the target animation where the target color is nonzero for the specified frame. Source and target are referenced by id.

```
ChromaAnimationAPI.CopyNonZeroTargetAllKeys(int sourceAnimationId, int
targetAnimationId, int frameId);
```

CopyNonZeroTargetAllKeysAllFrames

Copy nonzero colors from the source animation to the target animation where the target color is nonzero for all frames. Source and target are referenced by id.

```
ChromaAnimationAPI.CopyNonZeroTargetAllKeysAllFrames(int sourceAnimationId, int
targetAnimationId);
```

CopyNonZeroTargetAllKeysAllFramesName

Copy nonzero colors from the source animation to the target animation where the target color is nonzero for all frames. Source and target are referenced by name.

```
ChromaAnimationAPI.CopyNonZeroTargetAllKeysAllFramesName(string sourceAnimation,
string targetAnimation);
```

CopyNonZeroTargetAllKeysAllFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.CopyNonZeroTargetAllKeysAllFramesNameD(string
sourceAnimation, string targetAnimation);
```

CopyNonZeroTargetAllKeysAllFramesOffset

Copy nonzero colors from the source animation to the target animation where the target color is nonzero for all frames. Source and target are referenced by name.

```
ChromaAnimationAPI.CopyNonZeroTargetAllKeysAllFramesOffset(int sourceAnimationId,
int targetAnimationId, int offset);
```

CopyNonZeroTargetAllKeysAllFramesOffsetName

Copy nonzero colors from the source animation to the target animation where the target color is nonzero for all frames starting at the target offset for the length of the source animation. Source and target animations are referenced by name.

```
ChromaAnimationAPI.CopyNonZeroTargetAllKeysAllFramesOffsetName(string
sourceAnimation, string targetAnimation, int offset);
```

CopyNonZeroTargetAllKeysAllFramesOffsetNameD

D suffix for limited data types.

```
double result =
ChromaAnimationAPI.CopyNonZeroTargetAllKeysAllFramesOffsetNameD(string
sourceAnimation, string targetAnimation, double offset);
```

CopyNonZeroTargetAllKeysName

Copy nonzero colors from the source animation to the target animation where the target color is nonzero for the specified frame. The source and target are referenced by name.

```
ChromaAnimationAPI.CopyNonZeroTargetAllKeysName(string sourceAnimation, string
targetAnimation, int frameId);
```

CopyNonZeroTargetAllKeysNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.CopyNonZeroTargetAllKeysNameD(string
sourceAnimation, string targetAnimation, double frameId);
```

CopyNonZeroTargetAllKeysOffset

Copy nonzero colors from the source animation to the target animation where the target color is nonzero for the specified source frame and target offset frame. The source and target are referenced by id.

```
ChromaAnimationAPI.CopyNonZeroTargetAllKeysOffset(int sourceAnimationId, int
targetAnimationId, int frameId, int offset);
```

CopyNonZeroTargetAllKeysOffsetName

Copy nonzero colors from the source animation to the target animation where the target color is nonzero for the specified source frame and target offset frame. The source and target are referenced by name.

```
ChromaAnimationAPI.CopyNonZeroTargetAllKeysOffsetName(string sourceAnimation,
string targetAnimation, int frameId, int offset);
```

CopyNonZeroTargetAllKeysOffsetNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.CopyNonZeroTargetAllKeysOffsetNameD(string
sourceAnimation, string targetAnimation, double frameId, double offset);
```

CopyNonZeroTargetZeroAllKeysAllFrames

Copy nonzero colors from the source animation to the target animation where the target color is zero for all frames. Source and target are referenced by id.

```
ChromaAnimationAPI.CopyNonZeroTargetZeroAllKeysAllFrames(int sourceAnimationId,
int targetAnimationId);
```

CopyNonZeroTargetZeroAllKeysAllFramesName

Copy nonzero colors from the source animation to the target animation where the target color is zero for all frames. Source and target are referenced by name.

```
ChromaAnimationAPI.CopyNonZeroTargetZeroAllKeysAllFramesName(string
sourceAnimation, string targetAnimation);
```

CopyNonZeroTargetZeroAllKeysAllFramesNameD

D suffix for limited data types.

```
double result =
ChromaAnimationAPI.CopyNonZeroTargetZeroAllKeysAllFramesNameD(string
sourceAnimation, string targetAnimation);
```

CopyRedChannelAllFrames

Copy red channel to other channels for all frames. Intensity range is 0.0 to 1.0. Reference the animation by id.

```
ChromaAnimationAPI.CopyRedChannelAllFrames(int animationId, float greenIntensity,
float blueIntensity);
```

CopyRedChannelAllFramesName

Copy green channel to other channels for all frames. Intensity range is 0.0 to 1.0. Reference the animation by name.

```
ChromaAnimationAPI.CopyRedChannelAllFramesName(string path, float greenIntensity,
float blueIntensity);
```

CopyRedChannelAllFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.CopyRedChannelAllFramesNameD(string path,
double greenIntensity, double blueIntensity);
```

CopyZeroAllKeys

Copy zero colors from source animation to target animation for the frame. Source and target are referenced by id.

```
ChromaAnimationAPI.CopyZeroAllKeys(int sourceAnimationId, int targetAnimationId,
int frameId);
```

CopyZeroAllKeysAllFrames

Copy zero colors from source animation to target animation for all frames. Source and target are referenced by id.

```
ChromaAnimationAPI.CopyZeroAllKeysAllFrames(int sourceAnimationId, int
targetAnimationId);
```

CopyZeroAllKeysAllFramesName

Copy zero colors from source animation to target animation for all frames. Source and target are referenced by name.

```
ChromaAnimationAPI.CopyZeroAllKeysAllFramesName(string sourceAnimation, string
targetAnimation);
```

CopyZeroAllKeysAllFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.CopyZeroAllKeysAllFramesNameD(string
sourceAnimation, string targetAnimation);
```

CopyZeroAllKeysAllFramesOffset

Copy zero colors from source animation to target animation for all frames starting at the target offset for the length of the source animation. Source and target are referenced by id.

```
ChromaAnimationAPI.CopyZeroAllKeysAllFramesOffset(int sourceAnimationId, int
targetAnimationId, int offset);
```

CopyZeroAllKeysAllFramesOffsetName

Copy zero colors from source animation to target animation for all frames starting at the target offset for the length of the source animation. Source and target are referenced by name.

```
ChromaAnimationAPI.CopyZeroAllKeysAllFramesOffsetName(string sourceAnimation,
string targetAnimation, int offset);
```

CopyZeroAllKeysAllFramesOffsetNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.CopyZeroAllKeysAllFramesOffsetNameD(string
sourceAnimation, string targetAnimation, double offset);
```

CopyZeroAllKeysName

Copy zero colors from source animation to target animation for the frame. Source and target are referenced by name.

```
ChromaAnimationAPI.CopyZeroAllKeysName(string sourceAnimation, string
targetAnimation, int frameId);
```

CopyZeroAllKeysOffset

Copy zero colors from source animation to target animation for the frame id starting at the target offset for the length of the source animation. Source and target are referenced by id.

```
ChromaAnimationAPI.CopyZeroAllKeysOffset(int sourceAnimationId, int
targetAnimationId, int frameId, int offset);
```

CopyZeroAllKeysOffsetName

Copy zero colors from source animation to target animation for the frame id starting at the target offset for the length of the source animation. Source and target are referenced by name.

```
ChromaAnimationAPI.CopyZeroAllKeysOffsetName(string sourceAnimation, string
targetAnimation, int frameId, int offset);
```

CopyZeroKeyColor

Copy zero key color from source animation to target animation for the specified frame. Source and target are referenced by id.

```
ChromaAnimationAPI.CopyZeroKeyColor(int sourceAnimationId, int targetAnimationId,
int frameId, int rzkey);
```

CopyZeroKeyColorName

Copy zero key color from source animation to target animation for the specified frame. Source and target are referenced by name.

```
ChromaAnimationAPI.CopyZeroKeyColorName(string sourceAnimation, string
targetAnimation, int frameId, int rzkey);
```

CopyZeroKeyColorNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.CopyZeroKeyColorNameD(string sourceAnimation,
string targetAnimation, double frameId, double rzkey);
```

CopyZeroTargetAllKeys

Copy nonzero color from source animation to target animation where target is zero for the frame. Source and target are referenced by id.

```
ChromaAnimationAPI.CopyZeroTargetAllKeys(int sourceAnimationId, int
targetAnimationId, int frameId);
```

CopyZeroTargetAllKeysAllFrames

Copy nonzero color from source animation to target animation where target is zero for all frames. Source and target are referenced by id.

```
ChromaAnimationAPI.CopyZeroTargetAllKeysAllFrames(int sourceAnimationId, int
targetAnimationId);
```

CopyZeroTargetAllKeysAllFramesName

Copy nonzero color from source animation to target animation where target is zero for all frames. Source and target are referenced by name.

```
ChromaAnimationAPI.CopyZeroTargetAllKeysAllFramesName(string sourceAnimation,
string targetAnimation);
```

CopyZeroTargetAllKeysAllFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.CopyZeroTargetAllKeysAllFramesNameD(string
sourceAnimation, string targetAnimation);
```

CopyZeroTargetAllKeysName

Copy nonzero color from source animation to target animation where target is zero for the frame. Source and target are referenced by name.

```
ChromaAnimationAPI.CopyZeroTargetAllKeysName(string sourceAnimation, string
targetAnimation, int frameId);
```

CoreCreateChromaLinkEffect

Direct access to low level API.

```
int result = ChromaAnimationAPI.CoreCreateChromaLinkEffect(int effect, IntPtr
pParam, out Guid pEffectId);
```

CoreCreateEffect

Direct access to low level API.

```
int result = ChromaAnimationAPI.CoreCreateEffect(Guid deviceId, EFFECT_TYPE
effect, IntPtr pParam, out Guid pEffectId);
```

CoreCreateHeadsetEffect

Direct access to low level API.

```
int result = ChromaAnimationAPI.CoreCreateHeadsetEffect(int effect, IntPtr pParam,
out Guid pEffectId);
```

CoreCreateKeyboardEffect

Direct access to low level API.

```
int result = ChromaAnimationAPI.CoreCreateKeyboardEffect(int effect, IntPtr
pParam, out Guid pEffectId);
```

CoreCreateKeypadEffect

Direct access to low level API.

```
int result = ChromaAnimationAPI.CoreCreateKeypadEffect(int effect, IntPtr pParam,
out Guid pEffectId);
```

CoreCreateMouseEffect

Direct access to low level API.

```
int result = ChromaAnimationAPI.CoreCreateMouseEffect(int effect, IntPtr pParam,
out Guid pEffectId);
```

CoreCreateMousepadEffect

Direct access to low level API.

```
int result = ChromaAnimationAPI.CoreCreateMousepadEffect(int effect, IntPtr
pParam, out Guid pEffectId);
```

CoreDeleteEffect

Direct access to low level API.

```
int result = ChromaAnimationAPI.CoreDeleteEffect(Guid effectId);
```

CoreInit

Direct access to low level API.

```
int result = ChromaAnimationAPI.CoreInit();
```

CoreInitSDK

Direct access to low level API.

```
int result = ChromaAnimationAPI.CoreInitSDK(ref ChromaSDK.APPINFOTYPE appInfo);
```

CoreIsActive

Direct access to low level API.

```
int result = ChromaAnimationAPI.CoreIsActive(BOOL& active);
```

CoreIsConnected

Direct access to low level API.

```
int result = ChromaAnimationAPI.CoreIsConnected(ChromaSDK::DEVICE_INFO_TYPE&
deviceInfo);
```

CoreQueryDevice

Direct access to low level API.

```
int result = ChromaAnimationAPI.CoreQueryDevice(Guid deviceId, out
DEVICE_INFO_TYPE deviceInfo);
```

CoreSetEffect

Direct access to low level API.

```
int result = ChromaAnimationAPI.CoreSetEffect(Guid effectId);
```

CoreSetEventName

Direct access to low level API.

```
int result = ChromaAnimationAPI.CoreSetEventName(LPCTSTR name);
```

CoreStreamBroadcast

Begin broadcasting Chroma RGB data using the stored stream key as the endpoint. Intended for Cloud Gaming Platforms, restore the streaming key when the game instance is launched to continue streaming. streamId is a null terminated string streamKey is a null terminated string StreamGetStatus() should return the READY status to use this method.

```
bool result = ChromaAnimationAPI.CoreStreamBroadcast(string streamId, string
streamKey);
```

CoreStreamBroadcastEnd

End broadcasting Chroma RGB data. StreamGetStatus() should return the BROADCASTING status to use this method.

```
bool result = ChromaAnimationAPI.CoreStreamBroadcastEnd();
```

CoreStreamGetAuthShortcode

shortcode: Pass the address of a preallocated character buffer to get the streaming auth code. The buffer should have a minimum length of 6. length: Length will return as zero if the streaming auth code could not be obtained. If length is greater than zero, it will be the length of the returned streaming auth code. Once you have the shortcode, it should be shown to the user so they can associate the stream with their Razer ID StreamGetStatus() should return the READY status before invoking this method. platform: is the null terminated string that identifies the source of the stream: { GEFORCE_NOW, LUNA, STADIA, GAME_PASS } title: is the null terminated string that identifies the application or game.

```
ChromaAnimationAPI.CoreStreamGetAuthShortcode(ref string shortcode, out byte
length, string platform, string title);
```

CoreStreamGetFocus

focus: Pass the address of a preallocated character buffer to get the stream focus. The buffer should have a length of 48 length: Length will return as zero if the stream focus could not be obtained. If length is greater than zero, it will be the length of the returned stream focus.

```
bool result = ChromaAnimationAPI.CoreStreamGetFocus(ref string focus, out byte
length);
```

CoreStreamGetId

Intended for Cloud Gaming Platforms, store the stream id to persist in user preferences to continue streaming if the game is suspended or closed. shortcode: The shortcode is a null terminated string. Use the shortcode that authorized the stream to obtain the stream id. streamId should be a preallocated buffer to get the stream key. The buffer should have a length of 48. length: Length will return zero if the key could not be obtained. If the length is greater than zero, it will be the length of the returned streaming id. Retrieve the stream id after authorizing the shortcode. The authorization window will expire in 5 minutes. Be sure to save the stream key before the window expires. StreamGetStatus() should return the READY status to use this method.

```
ChromaAnimationAPI.CoreStreamGetId(string shortcode, ref string streamId, out byte
length);
```

CoreStreamGetKey

Intended for Cloud Gaming Platforms, store the streaming key to persist in user preferences to continue streaming if the game is suspended or closed. shortcode: The shortcode is a null terminated string. Use the shortcode that authorized the stream to obtain the stream key. If the status is in the BROADCASTING or WATCHING state, passing a NULL shortcode will return the active streamId. streamKey should be a preallocated buffer to get the stream key. The buffer should have a length of 48. length: Length will return zero if the key could not be obtained. If the length is greater than zero, it will be the length of the returned streaming key. Retrieve the stream key after authorizing the shortcode. The authorization window will expire in 5 minutes. Be sure to save the stream key before the window expires. StreamGetStatus() should return the READY status to use this method.

```
ChromaAnimationAPI.CoreStreamGetKey(string shortcode, ref string streamKey, out
byte length);
```

CoreStreamGetStatus

Returns StreamStatus, the current status of the service

```
ChromaSDK.Stream.StreamStatusType result =
ChromaAnimationAPI.CoreStreamGetStatus();
```

CoreStreamGetStatusString

Convert StreamStatusType to a printable string

```
string result =
ChromaAnimationAPI.CoreStreamGetStatusString(ChromaSDK.Stream.StreamStatusType
status);
```

CoreStreamReleaseShortcode

This prevents the stream id and stream key from being obtained through the shortcode. This closes the auth window. shortcode is a null terminated string. StreamGetStatus() should return the READY status to use this method. returns success when shortcode has been released

```
bool result = ChromaAnimationAPI.CoreStreamReleaseShortcode(string shortcode);
```

CoreStreamSetFocus

The focus is a null terminated string. Set the focus identifer for the application designated to automatically change the streaming state. Returns true on success.

```
bool result = ChromaAnimationAPI.CoreStreamSetFocus(string focus);
```

CoreStreamSupportsStreaming

Returns true if the Chroma streaming is supported. If false is returned, avoid calling stream methods.

```
bool result = ChromaAnimationAPI.CoreStreamSupportsStreaming();
```

CoreStreamWatch

Begin watching the Chroma RGB data using streamID parameter. streamId is a null terminated string. StreamGetStatus() should return the READY status to use this method.

```
bool result = ChromaAnimationAPI.CoreStreamWatch(string streamId, ulong
timestamp);
```

CoreStreamWatchEnd

End watching Chroma RGB data stream. StreamGetStatus() should return the WATCHING status to use this method.

```
bool result = ChromaAnimationAPI.CoreStreamWatchEnd();
```

CoreUnInit

Direct access to low level API.

```
int result = ChromaAnimationAPI.CoreUnInit();
```

CreateAnimation

Creates a `Chroma` animation at the given path. The `deviceType` parameter uses `EChromaSDKDeviceTypeEnum` as an integer. The `device` parameter uses `EChromaSDKDevice1DEnum` or `EChromaSDKDevice2DEnum` as an integer, respective to the `deviceType`. Returns the animation id upon success. Returns negative one upon failure. Saves a `Chroma` animation file with the `.chroma` extension at the given path. Returns the animation id upon success. Returns negative one upon failure.

```
int result = ChromaAnimationAPI.CreateAnimation(string path, int deviceType, int
device);
```

CreateAnimationInMemory

Creates a `Chroma` animation in memory without creating a file. The `deviceType` parameter uses `EChromaSDKDeviceTypeEnum` as an integer. The `device` parameter uses `EChromaSDKDevice1DEnum` or `EChromaSDKDevice2DEnum` as an integer, respective to the `deviceType`. Returns the animation id upon success. Returns negative one upon failure. Returns the animation id upon success. Returns negative one upon failure.

```
int result = ChromaAnimationAPI.CreateAnimationInMemory(int deviceType, int
device);
```

CreateEffect

Create a device specific effect.

```
int result = ChromaAnimationAPI.CreateEffect(Guid deviceId, EFFECT_TYPE effect,
int[] colors, int size, out FChromaSDKGuid effectId);
```

DeleteEffect

Delete an effect given the effect id.

```
int result = ChromaAnimationAPI.DeleteEffect(Guid effectId);
```

DuplicateFirstFrame

Duplicate the first animation frame so that the animation length matches the frame count. Animation is referenced by id.

```
ChromaAnimationAPI.DuplicateFirstFrame(int animationId, int frameCount);
```

DuplicateFirstFrameName

Duplicate the first animation frame so that the animation length matches the frame count. Animation is referenced by name.

```
ChromaAnimationAPI.DuplicateFirstFrameName(string path, int frameCount);
```

DuplicateFirstFrameNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.DuplicateFirstFrameNameD(string path, double
frameCount);
```

DuplicateFrames

Duplicate all the frames of the animation to double the animation length. Frame 1 becomes frame 1 and 2. Frame 2 becomes frame 3 and 4. And so on. The animation is referenced by id.

```
ChromaAnimationAPI.DuplicateFrames(int animationId);
```

DuplicateFramesName

Duplicate all the frames of the animation to double the animation length. Frame 1 becomes frame 1 and 2. Frame 2 becomes frame 3 and 4. And so on. The animation is referenced by name.

```
ChromaAnimationAPI.DuplicateFramesName(string path);
```

DuplicateFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.DuplicateFramesNameD(string path);
```

DuplicateMirrorFrames

Duplicate all the animation frames in reverse so that the animation plays forwards and backwards. Animation is referenced by id.

```
ChromaAnimationAPI.DuplicateMirrorFrames(int animationId);
```

DuplicateMirrorFramesName

Duplicate all the animation frames in reverse so that the animation plays forwards and backwards. Animation is referenced by name.

```
ChromaAnimationAPI.DuplicateMirrorFramesName(string path);
```

DuplicateMirrorFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.DuplicateMirrorFramesNameD(string path);
```

FadeEndFrames

Fade the animation to black starting at the fade frame index to the end of the animation. Animation is referenced by id.

```
ChromaAnimationAPI.FadeEndFrames(int animationId, int fade);
```

FadeEndFramesName

Fade the animation to black starting at the fade frame index to the end of the animation. Animation is referenced by name.

```
ChromaAnimationAPI.FadeEndFramesName(string path, int fade);
```

FadeEndFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.FadeEndFramesNameD(string path, double fade);
```

FadeStartFrames

Fade the animation from black to full color starting at 0 to the fade frame index. Animation is referenced by id.

```
ChromaAnimationAPI.FadeStartFrames(int animationId, int fade);
```

FadeStartFramesName

Fade the animation from black to full color starting at 0 to the fade frame index. Animation is referenced by name.

```
ChromaAnimationAPI.FadeStartFramesName(string path, int fade);
```

FadeStartFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.FadeStartFramesNameD(string path, double fade);
```

FillColor

Set the RGB value for all colors in the specified frame. Animation is referenced by id.

```
ChromaAnimationAPI.FillColor(int animationId, int frameId, int color);
```

FillColorAllFrames

Set the RGB value for all colors for all frames. Animation is referenced by id.

```
ChromaAnimationAPI.FillColorAllFrames(int animationId, int color);
```

FillColorAllFramesName

Set the RGB value for all colors for all frames. Animation is referenced by name.

```
ChromaAnimationAPI.FillColorAllFramesName(string path, int color);
```

FillColorAllFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.FillColorAllFramesNameD(string path, double
color);
```

FillColorAllFramesRGB

Set the RGB value for all colors for all frames. Use the range of 0 to 255 for red, green, and blue parameters. Animation is referenced by id.

```
ChromaAnimationAPI.FillColorAllFramesRGB(int animationId, int red, int green, int
blue);
```

FillColorAllFramesRGBName

Set the RGB value for all colors for all frames. Use the range of 0 to 255 for red, green, and blue parameters. Animation is referenced by name.

```
ChromaAnimationAPI.FillColorAllFramesRGBName(string path, int red, int green, int
blue);
```

FillColorAllFramesRGBNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.FillColorAllFramesRGBNameD(string path, double
red, double green, double blue);
```

FillColorName

Set the RGB value for all colors in the specified frame. Animation is referenced by name.

```
ChromaAnimationAPI.FillColorName(string path, int frameId, int color);
```

FillColorNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.FillColorNameD(string path, double frameId,
double color);
```

FillColorRGB

Set the RGB value for all colors in the specified frame. Animation is referenced by id.

```
ChromaAnimationAPI.FillColorRGB(int animationId, int frameId, int red, int green,
int blue);
```

FillColorRGBName

Set the RGB value for all colors in the specified frame. Animation is referenced by name.

```
ChromaAnimationAPI.FillColorRGBName(string path, int frameId, int red, int green,
int blue);
```

FillColorRGBNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.FillColorRGBNameD(string path, double frameId,
double red, double green, double blue);
```

FillNonZeroColor

This method will only update colors in the animation that are not already set to black. Set the RGB value for a subset of colors in the specified frame. Animation is referenced by id.

```
ChromaAnimationAPI.FillNonZeroColor(int animationId, int frameId, int color);
```

FillNonZeroColorAllFrames

This method will only update colors in the animation that are not already set to black. Set the RGB value for a subset of colors for all frames. Animation is referenced by id.

```
ChromaAnimationAPI.FillNonZeroColorAllFrames(int animationId, int color);
```

FillNonZeroColorAllFramesName

This method will only update colors in the animation that are not already set to black. Set the RGB value for a subset of colors for all frames. Animation is referenced by name.

```
ChromaAnimationAPI.FillNonZeroColorAllFramesName(string path, int color);
```

FillNonZeroColorAllFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.FillNonZeroColorAllFramesNameD(string path,
double color);
```

FillNonZeroColorAllFramesRGB

This method will only update colors in the animation that are not already set to black. Set the RGB value for a subset of colors for all frames. Use the range of 0 to 255 for red, green, and blue parameters. Animation is referenced by id.

```
ChromaAnimationAPI.FillNonZeroColorAllFramesRGB(int animationId, int red, int
green, int blue);
```

FillNonZeroColorAllFramesRGBName

This method will only update colors in the animation that are not already set to black. Set the RGB value for a subset of colors for all frames. Use the range of 0 to 255 for red, green, and blue parameters. Animation is referenced by name.

```
ChromaAnimationAPI.FillNonZeroColorAllFramesRGBName(string path, int red, int
green, int blue);
```

FillNonZeroColorAllFramesRGBNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.FillNonZeroColorAllFramesRGBNameD(string path,
double red, double green, double blue);
```

FillNonZeroColorName

This method will only update colors in the animation that are not already set to black. Set the RGB value for a subset of colors in the specified frame. Animation is referenced by name.

```
ChromaAnimationAPI.FillNonZeroColorName(string path, int frameId, int color);
```

FillNonZeroColorNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.FillNonZeroColorNameD(string path, double
frameId, double color);
```

FillNonZeroColorRGB

This method will only update colors in the animation that are not already set to black. Set the RGB value for a subset of colors in the specified frame. Use the range of 0 to 255 for red, green, and blue parameters. Animation is referenced by id.

```
ChromaAnimationAPI.FillNonZeroColorRGB(int animationId, int frameId, int red, int
green, int blue);
```

FillNonZeroColorRGBName

This method will only update colors in the animation that are not already set to black. Set the RGB value for a subset of colors in the specified frame. Use the range of 0 to 255 for red, green, and blue parameters. Animation is referenced by name.

```
ChromaAnimationAPI.FillNonZeroColorRGBName(string path, int frameId, int red, int
green, int blue);
```

FillNonZeroColorRGBNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.FillNonZeroColorRGBNameD(string path, double
frameId, double red, double green, double blue);
```

FillRandomColors

Fill the frame with random RGB values for the given frame. Animation is referenced by id.

```
ChromaAnimationAPI.FillRandomColors(int animationId, int frameId);
```

FillRandomColorsAllFrames

Fill the frame with random RGB values for all frames. Animation is referenced by id.

```
ChromaAnimationAPI.FillRandomColorsAllFrames(int animationId);
```

FillRandomColorsAllFramesName

Fill the frame with random RGB values for all frames. Animation is referenced by name.

```
ChromaAnimationAPI.FillRandomColorsAllFramesName(string path);
```

FillRandomColorsAllFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.FillRandomColorsAllFramesNameD(string path);
```

FillRandomColorsBlackAndWhite

Fill the frame with random black and white values for the specified frame. Animation is referenced by id.

```
ChromaAnimationAPI.FillRandomColorsBlackAndWhite(int animationId, int frameId);
```

FillRandomColorsBlackAndWhiteAllFrames

Fill the frame with random black and white values for all frames. Animation is referenced by id.

```
ChromaAnimationAPI.FillRandomColorsBlackAndWhiteAllFrames(int animationId);
```

FillRandomColorsBlackAndWhiteAllFramesName

Fill the frame with random black and white values for all frames. Animation is referenced by name.

```
ChromaAnimationAPI.FillRandomColorsBlackAndWhiteAllFramesName(string path);
```

FillRandomColorsBlackAndWhiteAllFramesNameD

D suffix for limited data types.

```
double result =
ChromaAnimationAPI.FillRandomColorsBlackAndWhiteAllFramesNameD(string path);
```

FillRandomColorsBlackAndWhiteName

Fill the frame with random black and white values for the specified frame. Animation is referenced by name.

```
ChromaAnimationAPI.FillRandomColorsBlackAndWhiteName(string path, int frameId);
```

FillRandomColorsBlackAndWhiteNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.FillRandomColorsBlackAndWhiteNameD(string path,
double frameId);
```

FillRandomColorsName

Fill the frame with random RGB values for the given frame. Animation is referenced by name.

```
ChromaAnimationAPI.FillRandomColorsName(string path, int frameId);
```

FillRandomColorsNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.FillRandomColorsNameD(string path, double
frameId);
```

FillThresholdColors

Fill the specified frame with RGB color where the animation color is less than the RGB threshold. Animation is referenced by id.

```
ChromaAnimationAPI.FillThresholdColors(int animationId, int frameId, int
threshold, int color);
```

FillThresholdColorsAllFrames

Fill all frames with RGB color where the animation color is less than the RGB threshold. Animation is referenced by id.

```
ChromaAnimationAPI.FillThresholdColorsAllFrames(int animationId, int threshold,
int color);
```

FillThresholdColorsAllFramesName

Fill all frames with RGB color where the animation color is less than the RGB threshold. Animation is referenced by name.

```
ChromaAnimationAPI.FillThresholdColorsAllFramesName(string path, int threshold,
int color);
```

FillThresholdColorsAllFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.FillThresholdColorsAllFramesNameD(string path,
double threshold, double color);
```

FillThresholdColorsAllFramesRGB

Fill all frames with RGB color where the animation color is less than the threshold. Animation is referenced by id.

```
ChromaAnimationAPI.FillThresholdColorsAllFramesRGB(int animationId, int threshold,
int red, int green, int blue);
```

FillThresholdColorsAllFramesRGBName

Fill all frames with RGB color where the animation color is less than the threshold. Animation is referenced by name.

```
ChromaAnimationAPI.FillThresholdColorsAllFramesRGBName(string path, int threshold,
int red, int green, int blue);
```

FillThresholdColorsAllFramesRGBNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.FillThresholdColorsAllFramesRGBNameD(string
path, double threshold, double red, double green, double blue);
```

FillThresholdColorsMinMaxAllFramesRGB

Fill all frames with the min RGB color where the animation color is less than the min threshold AND with the max RGB color where the animation is more than the max threshold. Animation is referenced by id.

```
ChromaAnimationAPI.FillThresholdColorsMinMaxAllFramesRGB(int animationId, int
minThreshold, int minRed, int minGreen, int minBlue, int maxThreshold, int maxRed,
int maxGreen, int maxBlue);
```

FillThresholdColorsMinMaxAllFramesRGBName

Fill all frames with the min RGB color where the animation color is less than the min threshold AND with the max RGB color where the animation is more than the max threshold. Animation is referenced by name.

```
ChromaAnimationAPI.FillThresholdColorsMinMaxAllFramesRGBName(string path, int
minThreshold, int minRed, int minGreen, int minBlue, int maxThreshold, int maxRed,
int maxGreen, int maxBlue);
```

FillThresholdColorsMinMaxAllFramesRGBNameD

D suffix for limited data types.

```
double result =
ChromaAnimationAPI.FillThresholdColorsMinMaxAllFramesRGBNameD(string path, double
minThreshold, double minRed, double minGreen, double minBlue, double maxThreshold,
double maxRed, double maxGreen, double maxBlue);
```

FillThresholdColorsMinMaxRGB

Fill the specified frame with the min RGB color where the animation color is less than the min threshold AND with the max RGB color where the animation is more than the max threshold. Animation is referenced by id.

```
ChromaAnimationAPI.FillThresholdColorsMinMaxRGB(int animationId, int frameId, int
minThreshold, int minRed, int minGreen, int minBlue, int maxThreshold, int maxRed,
int maxGreen, int maxBlue);
```

FillThresholdColorsMinMaxRGBName

Fill the specified frame with the min RGB color where the animation color is less than the min threshold AND with the max RGB color where the animation is more than the max threshold. Animation is referenced by name.

```
ChromaAnimationAPI.FillThresholdColorsMinMaxRGBName(string path, int frameId, int
minThreshold, int minRed, int minGreen, int minBlue, int maxThreshold, int maxRed,
int maxGreen, int maxBlue);
```

FillThresholdColorsMinMaxRGBNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.FillThresholdColorsMinMaxRGBNameD(string path,
double frameId, double minThreshold, double minRed, double minGreen, double
minBlue, double maxThreshold, double maxRed, double maxGreen, double maxBlue);
```

FillThresholdColorsName

Fill the specified frame with RGB color where the animation color is less than the RGB threshold. Animation is referenced by name.

```
ChromaAnimationAPI.FillThresholdColorsName(string path, int frameId, int
threshold, int color);
```

FillThresholdColorsNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.FillThresholdColorsNameD(string path, double
frameId, double threshold, double color);
```

FillThresholdColorsRGB

Fill the specified frame with RGB color where the animation color is less than the RGB threshold. Animation is referenced by id.

```
ChromaAnimationAPI.FillThresholdColorsRGB(int animationId, int frameId, int
threshold, int red, int green, int blue);
```

FillThresholdColorsRGBName

Fill the specified frame with RGB color where the animation color is less than the RGB threshold. Animation is referenced by name.

```
ChromaAnimationAPI.FillThresholdColorsRGBName(string path, int frameId, int
threshold, int red, int green, int blue);
```

FillThresholdColorsRGBNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.FillThresholdColorsRGBNameD(string path, double
frameId, double threshold, double red, double green, double blue);
```

FillThresholdRGBColorsAllFramesRGB

Fill all frames with RGB color where the animation color is less than the RGB threshold. Animation is referenced by id.

```
ChromaAnimationAPI.FillThresholdRGBColorsAllFramesRGB(int animationId, int
redThreshold, int greenThreshold, int blueThreshold, int red, int green, int
blue);
```

FillThresholdRGBColorsAllFramesRGBName

Fill all frames with RGB color where the animation color is less than the RGB threshold. Animation is referenced by name.

```
ChromaAnimationAPI.FillThresholdRGBColorsAllFramesRGBName(string path, int
redThreshold, int greenThreshold, int blueThreshold, int red, int green, int
blue);
```

FillThresholdRGBColorsAllFramesRGBNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.FillThresholdRGBColorsAllFramesRGBNameD(string
path, double redThreshold, double greenThreshold, double blueThreshold, double
red, double green, double blue);
```

FillThresholdRGBColorsRGB

Fill the specified frame with RGB color where the animation color is less than the RGB threshold. Animation is referenced by id.

```
ChromaAnimationAPI.FillThresholdRGBColorsRGB(int animationId, int frameId, int
redThreshold, int greenThreshold, int blueThreshold, int red, int green, int
blue);
```

FillThresholdRGBColorsRGBName

Fill the specified frame with RGB color where the animation color is less than the RGB threshold. Animation is referenced by name.

```
ChromaAnimationAPI.FillThresholdRGBColorsRGBName(string path, int frameId, int
redThreshold, int greenThreshold, int blueThreshold, int red, int green, int
blue);
```

FillThresholdRGBColorsRGBNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.FillThresholdRGBColorsRGBNameD(string path,
double frameId, double redThreshold, double greenThreshold, double blueThreshold,
double red, double green, double blue);
```

FillZeroColor

Fill the specified frame with RGB color where the animation color is zero. Animation is referenced by id.

```
ChromaAnimationAPI.FillZeroColor(int animationId, int frameId, int color);
```

FillZeroColorAllFrames

Fill all frames with RGB color where the animation color is zero. Animation is referenced by id.

```
ChromaAnimationAPI.FillZeroColorAllFrames(int animationId, int color);
```

FillZeroColorAllFramesName

Fill all frames with RGB color where the animation color is zero. Animation is referenced by name.

```
ChromaAnimationAPI.FillZeroColorAllFramesName(string path, int color);
```

FillZeroColorAllFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.FillZeroColorAllFramesNameD(string path, double
color);
```

FillZeroColorAllFramesRGB

Fill all frames with RGB color where the animation color is zero. Animation is referenced by id.

```
ChromaAnimationAPI.FillZeroColorAllFramesRGB(int animationId, int red, int green,
int blue);
```

FillZeroColorAllFramesRGBName

Fill all frames with RGB color where the animation color is zero. Animation is referenced by name.

```
ChromaAnimationAPI.FillZeroColorAllFramesRGBName(string path, int red, int green,
int blue);
```

FillZeroColorAllFramesRGBNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.FillZeroColorAllFramesRGBNameD(string path,
double red, double green, double blue);
```

FillZeroColorName

Fill the specified frame with RGB color where the animation color is zero. Animation is referenced by name.

```
ChromaAnimationAPI.FillZeroColorName(string path, int frameId, int color);
```

FillZeroColorNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.FillZeroColorNameD(string path, double frameId,
double color);
```

FillZeroColorRGB

Fill the specified frame with RGB color where the animation color is zero. Animation is referenced by id.

```
ChromaAnimationAPI.FillZeroColorRGB(int animationId, int frameId, int red, int
green, int blue);
```

FillZeroColorRGBName

Fill the specified frame with RGB color where the animation color is zero. Animation is referenced by name.

```
ChromaAnimationAPI.FillZeroColorRGBName(string path, int frameId, int red, int
green, int blue);
```

FillZeroColorRGBNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.FillZeroColorRGBNameD(string path, double
frameId, double red, double green, double blue);
```

Get1DColor

Get the animation color for a frame given the `1D` `led`. The `led` should be greater than or equal to 0 and less than the `MaxLeds`. Animation is referenced by id.

```
int result = ChromaAnimationAPI.Get1DColor(int animationId, int frameId, int led);
```

Get1DColorName

Get the animation color for a frame given the `1D` `led`. The `led` should be greater than or equal to 0 and less than the `MaxLeds`. Animation is referenced by name.

```
int result = ChromaAnimationAPI.Get1DColorName(string path, int frameId, int led);
```

Get1DColorNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.Get1DColorNameD(string path, double frameId,
double led);
```

Get2DColor

Get the animation color for a frame given the `2D` `row` and `column`. The `row` should be greater than or equal to 0 and less than the `MaxRow`. The `column` should be greater than or equal to 0 and less than the `MaxColumn`. Animation is referenced by id.

```
int result = ChromaAnimationAPI.Get2DColor(int animationId, int frameId, int row,
int column);
```

Get2DColorName

Get the animation color for a frame given the `2D` `row` and `column`. The `row` should be greater than or equal to 0 and less than the `MaxRow`. The `column` should be greater than or equal to 0 and less than the `MaxColumn`. Animation is referenced by name.

```
int result = ChromaAnimationAPI.Get2DColorName(string path, int frameId, int row,
int column);
```

Get2DColorNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.Get2DColorNameD(string path, double frameId,
double row, double column);
```

GetAnimation

Get the animation id for the named animation.

```
int result = ChromaAnimationAPI.GetAnimation(string name);
```

GetAnimationCount

`PluginGetAnimationCount` will return the number of loaded animations.

```
int result = ChromaAnimationAPI.GetAnimationCount();
```

GetAnimationD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.GetAnimationD(string name);
```

GetAnimationId

`PluginGetAnimationId` will return the `animationId` given the `index` of the loaded animation. The `index` is zero-based and less than the number returned by `PluginGetAnimationCount`. Use `PluginGetAnimationName` to get the name of the animation.

```
int result = ChromaAnimationAPI.GetAnimationId(int index);
```

GetAnimationName

`PluginGetAnimationName` takes an `animationId` and returns the name of the animation of the `.chroma` animation file. If a name is not available then an empty string will be returned.

```
string result = ChromaAnimationAPI.GetAnimationName(int animationId);
```

GetCurrentFrame

Get the current frame of the animation referenced by id.

```
int result = ChromaAnimationAPI.GetCurrentFrame(int animationId);
```

GetCurrentFrameName

Get the current frame of the animation referenced by name.

```
int result = ChromaAnimationAPI.GetCurrentFrameName(string path);
```

GetCurrentFrameNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.GetCurrentFrameNameD(string path);
```

GetDevice

Returns the `EChromaSDKDevice1DEnum` or `EChromaSDKDevice2DEnum` of a `Chroma` animation respective to the `deviceType`, as an integer upon success. Returns negative one upon failure.

```
int result = ChromaAnimationAPI.GetDevice(int animationId);
```

GetDeviceName

Returns the `EChromaSDKDevice1DEnum` or `EChromaSDKDevice2DEnum` of a `Chroma` animation respective to the `deviceType`, as an integer upon success. Returns negative one upon failure.

```
int result = ChromaAnimationAPI.GetDeviceName(string path);
```

GetDeviceNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.GetDeviceNameD(string path);
```

GetDeviceType

Returns the `EChromaSDKDeviceTypeEnum` of a `Chroma` animation as an integer upon success. Returns negative one upon failure.

```
int result = ChromaAnimationAPI.GetDeviceType(int animationId);
```

GetDeviceTypeName

Returns the `EChromaSDKDeviceTypeEnum` of a `Chroma` animation as an integer upon success. Returns negative one upon failure.

```
int result = ChromaAnimationAPI.GetDeviceTypeName(string path);
```

GetDeviceTypeNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.GetDeviceTypeNameD(string path);
```

GetFrame

Get the frame colors and duration (in seconds) for a `Chroma` animation referenced by id. The `color` is expected to be an array of the expected dimensions for the `deviceType/device`. The `length` parameter is the size of the `color` array. For `EChromaSDKDevice1DEnum` the array size should be `MAX_LEDS`. For `EChromaSDKDevice2DEnum` the array size should be `MAX_ROW` times `MAX_COLUMN`. Keys are populated only for EChromaSDKDevice2DEnum::DE_Keyboard and EChromaSDKDevice2DEnum::DE_KeyboardExtended. Keys will only use the EChromaSDKDevice2DEnum::DE_Keyboard `MAX_ROW` times `MAX_COLUMN` keysLength. Returns the animation id upon success. Returns negative one upon failure.

```
int result = ChromaAnimationAPI.GetFrame(int animationId, int frameId, out float
duration, int[] colors, int length, int[] keys, int keysLength);
```

GetFrameCount

Returns the frame count of a `Chroma` animation upon success. Returns negative one upon failure.

```
int result = ChromaAnimationAPI.GetFrameCount(int animationId);
```

GetFrameCountName

Returns the frame count of a `Chroma` animation upon success. Returns negative one upon failure.

```
int result = ChromaAnimationAPI.GetFrameCountName(string path);
```

GetFrameCountNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.GetFrameCountNameD(string path);
```

GetFrameDuration

Returns the duration of an animation frame in seconds upon success. Returns zero upon failure.

```
float result = ChromaAnimationAPI.GetFrameDuration(int animationId, int frameId);
```

GetFrameDurationName

Returns the duration of an animation frame in seconds upon success. Returns zero upon failure.

```
float result = ChromaAnimationAPI.GetFrameDurationName(string path, int frameId);
```

GetFrameName

Get the frame colors and duration (in seconds) for a `Chroma` animation referenced by name. The `color` is expected to be an array of the expected dimensions for the `deviceType/device`. The `length` parameter is the size of the `color` array. For `EChromaSDKDevice1DEnum` the array size should be `MAX LEDS`. For `EChromaSDKDevice2DEnum` the array size should be `MAX ROW` times `MAX COLUMN`. Keys are populated only for EChromaSDKDevice2DEnum::DE_Keyboard and EChromaSDKDevice2DEnum::DE_KeyboardExtended. Keys will only use the EChromaSDKDevice2DEnum::DE_Keyboard `MAX_ROW` times `MAX_COLUMN` keysLength. Returns the animation id upon success. Returns negative one upon failure.

```
int result = ChromaAnimationAPI.GetFrameName(string path, int frameId, out float
duration, int[] colors, int length, int[] keys, int keysLength);
```

GetKeyColor

Get the color of an animation key for the given frame referenced by id.

```
int result = ChromaAnimationAPI.GetKeyColor(int animationId, int frameId, int
rzkey);
```

GetKeyColorD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.GetKeyColorD(string path, double frameId,
double rzkey);
```

GetKeyColorName

Get the color of an animation key for the given frame referenced by name.

```
int result = ChromaAnimationAPI.GetKeyColorName(string path, int frameId, int
rzkey);
```

GetLibraryLoadedState

Returns `RZRESULT_SUCCESS` if the plugin has been initialized successfully. Returns `RZRESULT_DLL_NOT_FOUND` if core Chroma library is not found. Returns `RZRESULT_DLL_INVALID_SIGNATURE` if core Chroma library has an invalid signature.

```
int result = ChromaAnimationAPI.GetLibraryLoadedState();
```

GetLibraryLoadedStateD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.GetLibraryLoadedStateD();
```

GetMaxColumn

Returns the `MAX COLUMN` given the `EChromaSDKDevice2DEnum` device as an integer upon success. Returns negative one upon failure.

```
int result = ChromaAnimationAPI.GetMaxColumn(Device2D device);
```

GetMaxColumnD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.GetMaxColumnD(double device);
```

GetMaxLeds

Returns the MAX LEDS given the `EChromaSDKDevice1DEnum` device as an integer upon success. Returns negative one upon failure.

```
int result = ChromaAnimationAPI.GetMaxLeds(Device1D device);
```

GetMaxLedsD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.GetMaxLedsD(double device);
```

GetMaxRow

Returns the `MAX ROW` given the `EChromaSDKDevice2DEnum` device as an integer upon success. Returns negative one upon failure.

```
int result = ChromaAnimationAPI.GetMaxRow(Device2D device);
```

GetMaxRowD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.GetMaxRowD(double device);
```

GetPlayingAnimationCount

`PluginGetPlayingAnimationCount` will return the number of playing animations.

```
int result = ChromaAnimationAPI.GetPlayingAnimationCount();
```

GetPlayingAnimationId

`PluginGetPlayingAnimationId` will return the `animationId` given the `index` of the playing animation. The `index` is zero-based and less than the number returned by `PluginGetPlayingAnimationCount`. Use `PluginGetAnimationName` to get the name of the animation.

```
int result = ChromaAnimationAPI.GetPlayingAnimationId(int index);
```

GetRGB

Get the RGB color given red, green, and blue.

```
int result = ChromaAnimationAPI.GetRGB(int red, int green, int blue);
```

GetRGBD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.GetRGBD(double red, double green, double blue);
```

GetTotalDuration

Returns the total duration of an animation in seconds upon success. Returns zero upon failure.

```
float result = ChromaAnimationAPI.GetTotalDuration(int animationId);
```

GetTotalDurationName

Returns the total duration of an animation in seconds upon success. Returns zero upon failure.

```
float result = ChromaAnimationAPI.GetTotalDurationName(string path);
```

HasAnimationLoop

Check if the animation has loop enabled referenced by id.

```
bool result = ChromaAnimationAPI.HasAnimationLoop(int animationId);
```

HasAnimationLoopName

Check if the animation has loop enabled referenced by name.

```
bool result = ChromaAnimationAPI.HasAnimationLoopName(string path);
```

HasAnimationLoopNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.HasAnimationLoopNameD(string path);
```

Init

Initialize the ChromaSDK. Zero indicates success, otherwise failure. Many API methods auto initialize the ChromaSDK if not already initialized.

```
int result = ChromaAnimationAPI.Init();
```

InitD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.InitD();
```

InitSDK

Initialize the ChromaSDK. AppInfo populates the details in Synapse. Zero indicates success, otherwise failure. Many API methods auto initialize the ChromaSDK if not already initialized.

```
int result = ChromaAnimationAPI.InitSDK(ref ChromaSDK.APPINFOTYPE appInfo);
```

InsertDelay

Insert an animation delay by duplicating the frame by the delay number of times. Animation is referenced by id.

```
ChromaAnimationAPI.InsertDelay(int animationId, int frameId, int delay);
```

InsertDelayName

Insert an animation delay by duplicating the frame by the delay number of times. Animation is referenced by name.

```
ChromaAnimationAPI.InsertDelayName(string path, int frameId, int delay);
```

InsertDelayNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.InsertDelayNameD(string path, double frameId,
double delay);
```

InsertFrame

Duplicate the source frame index at the target frame index. Animation is referenced by id.

```
ChromaAnimationAPI.InsertFrame(int animationId, int sourceFrame, int targetFrame);
```

InsertFrameName

Duplicate the source frame index at the target frame index. Animation is referenced by name.

```
ChromaAnimationAPI.InsertFrameName(string path, int sourceFrame, int targetFrame);
```

InsertFrameNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.InsertFrameNameD(string path, double
sourceFrame, double targetFrame);
```

InvertColors

Invert all the colors at the specified frame. Animation is referenced by id.

```
ChromaAnimationAPI.InvertColors(int animationId, int frameId);
```

InvertColorsAllFrames

Invert all the colors for all frames. Animation is referenced by id.

```
ChromaAnimationAPI.InvertColorsAllFrames(int animationId);
```

InvertColorsAllFramesName

Invert all the colors for all frames. Animation is referenced by name.

```
ChromaAnimationAPI.InvertColorsAllFramesName(string path);
```

InvertColorsAllFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.InvertColorsAllFramesNameD(string path);
```

InvertColorsName

Invert all the colors at the specified frame. Animation is referenced by name.

```
ChromaAnimationAPI.InvertColorsName(string path, int frameId);
```

InvertColorsNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.InvertColorsNameD(string path, double frameId);
```

IsAnimationPaused

Check if the animation is paused referenced by id.

```
bool result = ChromaAnimationAPI.IsAnimationPaused(int animationId);
```

IsAnimationPausedName

Check if the animation is paused referenced by name.

```
bool result = ChromaAnimationAPI.IsAnimationPausedName(string path);
```

IsAnimationPausedNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.IsAnimationPausedNameD(string path);
```

IsDialogOpen

The editor dialog is a non-blocking modal window, this method returns true if the modal window is open, otherwise false.

```
bool result = ChromaAnimationAPI.IsDialogOpen();
```

IsDialogOpenD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.IsDialogOpenD();
```

IsInitialized

Returns true if the plugin has been initialized. Returns false if the plugin is uninitialized.

```
bool result = ChromaAnimationAPI.IsInitialized();
```

IsInitializedD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.IsInitializedD();
```

IsPlatformSupported

If the method can be invoked the method returns true.

```
bool result = ChromaAnimationAPI.IsPlatformSupported();
```

IsPlatformSupportedD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.IsPlatformSupportedD();
```

IsPlaying

`PluginIsPlayingName` automatically handles initializing the `ChromaSDK`. The named `.chroma` animation file will be automatically opened. The method will return whether the animation is playing or not. Animation is referenced by id.

```
bool result = ChromaAnimationAPI.IsPlaying(int animationId);
```

IsPlayingD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.IsPlayingD(double animationId);
```

IsPlayingName

`PluginIsPlayingName` automatically handles initializing the `ChromaSDK`. The named `.chroma` animation file will be automatically opened. The method will return whether the animation is playing or not. Animation is referenced by name.

```
bool result = ChromaAnimationAPI.IsPlayingName(string path);
```

IsPlayingNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.IsPlayingNameD(string path);
```

IsPlayingType

`PluginIsPlayingType` automatically handles initializing the `ChromaSDK`. If any animation is playing for the `deviceType` and `device` combination, the method will return true, otherwise false.

```
bool result = ChromaAnimationAPI.IsPlayingType(int deviceType, int device);
```

IsPlayingTypeD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.IsPlayingTypeD(double deviceType, double
device);
```

Lerp

Do a lerp math operation on a float.

```
float result = ChromaAnimationAPI.Lerp(float start, float end, float amt);
```

LerpColor

Lerp from one color to another given t in the range 0.0 to 1.0.

```
int result = ChromaAnimationAPI.LerpColor(int from, int to, float t);
```

LoadAnimation

Loads `Chroma` effects so that the animation can be played immediately. Returns the animation id upon success. Returns negative one upon failure.

```
int result = ChromaAnimationAPI.LoadAnimation(int animationId);
```

LoadAnimationD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.LoadAnimationD(double animationId);
```

LoadAnimationName

Load the named animation.

```
ChromaAnimationAPI.LoadAnimationName(string path);
```

LoadComposite

Load a composite set of animations.

```
ChromaAnimationAPI.LoadComposite(string name);
```

MakeBlankFrames

Make a blank animation for the length of the frame count. Frame duration defaults to the duration. The frame color defaults to color. Animation is referenced by id.

```
ChromaAnimationAPI.MakeBlankFrames(int animationId, int frameCount, float
duration, int color);
```

MakeBlankFramesName

Make a blank animation for the length of the frame count. Frame duration defaults to the duration. The frame color defaults to color. Animation is referenced by name.

```
ChromaAnimationAPI.MakeBlankFramesName(string path, int frameCount, float
duration, int color);
```

MakeBlankFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.MakeBlankFramesNameD(string path, double
frameCount, double duration, double color);
```

MakeBlankFramesRandom

Make a blank animation for the length of the frame count. Frame duration defaults to the duration. The frame color is random. Animation is referenced by id.

```
ChromaAnimationAPI.MakeBlankFramesRandom(int animationId, int frameCount, float
duration);
```

MakeBlankFramesRandomBlackAndWhite

Make a blank animation for the length of the frame count. Frame duration defaults to the duration. The frame color is random black and white. Animation is referenced by id.

```
ChromaAnimationAPI.MakeBlankFramesRandomBlackAndWhite(int animationId, int
frameCount, float duration);
```

MakeBlankFramesRandomBlackAndWhiteName

Make a blank animation for the length of the frame count. Frame duration defaults to the duration. The frame color is random black and white. Animation is referenced by name.

```
ChromaAnimationAPI.MakeBlankFramesRandomBlackAndWhiteName(string path, int
frameCount, float duration);
```

MakeBlankFramesRandomBlackAndWhiteNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.MakeBlankFramesRandomBlackAndWhiteNameD(string
path, double frameCount, double duration);
```

MakeBlankFramesRandomName

Make a blank animation for the length of the frame count. Frame duration defaults to the duration. The frame color is random. Animation is referenced by name.

```
ChromaAnimationAPI.MakeBlankFramesRandomName(string path, int frameCount, float
duration);
```

MakeBlankFramesRandomNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.MakeBlankFramesRandomNameD(string path, double
frameCount, double duration);
```

MakeBlankFramesRGB

Make a blank animation for the length of the frame count. Frame duration defaults to the duration. The frame color defaults to color. Animation is referenced by id.

```
ChromaAnimationAPI.MakeBlankFramesRGB(int animationId, int frameCount, float
duration, int red, int green, int blue);
```

MakeBlankFramesRGBName

Make a blank animation for the length of the frame count. Frame duration defaults to the duration. The frame color defaults to color. Animation is referenced by name.

```
ChromaAnimationAPI.MakeBlankFramesRGBName(string path, int frameCount, float
duration, int red, int green, int blue);
```

MakeBlankFramesRGBNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.MakeBlankFramesRGBNameD(string path, double
frameCount, double duration, double red, double green, double blue);
```

MirrorHorizontally

Flips the color grid horizontally for all `Chroma` animation frames. Returns the animation id upon success. Returns negative one upon failure.

```
int result = ChromaAnimationAPI.MirrorHorizontally(int animationId);
```

MirrorVertically

Flips the color grid vertically for all `Chroma` animation frames. This method has no effect for `EChromaSDKDevice1DEnum` devices. Returns the animation id upon success. Returns negative one upon failure.

```
int result = ChromaAnimationAPI.MirrorVertically(int animationId);
```

MultiplyColorLerpAllFrames

Multiply the color intensity with the lerp result from color 1 to color 2 using the frame index divided by the frame count for the `t` parameter. Animation is referenced in id.

```
ChromaAnimationAPI.MultiplyColorLerpAllFrames(int animationId, int color1, int
color2);
```

MultiplyColorLerpAllFramesName

Multiply the color intensity with the lerp result from color 1 to color 2 using the frame index divided by the frame count for the `t` parameter. Animation is referenced in name.

```
ChromaAnimationAPI.MultiplyColorLerpAllFramesName(string path, int color1, int
color2);
```

MultiplyColorLerpAllFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.MultiplyColorLerpAllFramesNameD(string path,
double color1, double color2);
```

MultiplyIntensity

Multiply all the colors in the frame by the intensity value. The valid the intensity range is from 0.0 to 255.0. RGB components are multiplied equally. An intensity of 0.5 would half the color value. Black colors in the frame will not be affected by this method.

```
ChromaAnimationAPI.MultiplyIntensity(int animationId, int frameId, float
intensity);
```

MultiplyIntensityAllFrames

Multiply all the colors for all frames by the intensity value. The valid the intensity range is from 0.0 to 255.0. RGB components are multiplied equally. An intensity of 0.5 would half the color value. Black colors in the frame will not be affected by this method.

```
ChromaAnimationAPI.MultiplyIntensityAllFrames(int animationId, float intensity);
```

MultiplyIntensityAllFramesName

Multiply all the colors for all frames by the intensity value. The valid the intensity range is from 0.0 to 255.0. RGB components are multiplied equally. An intensity of 0.5 would half the color value. Black colors in the frame will not be affected by this method.

```
ChromaAnimationAPI.MultiplyIntensityAllFramesName(string path, float intensity);
```

MultiplyIntensityAllFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.MultiplyIntensityAllFramesNameD(string path,
double intensity);
```

MultiplyIntensityAllFramesRGB

Multiply all frames by the RBG color intensity. Animation is referenced by id.

```
ChromaAnimationAPI.MultiplyIntensityAllFramesRGB(int animationId, int red, int
green, int blue);
```

MultiplyIntensityAllFramesRGBName

Multiply all frames by the RBG color intensity. Animation is referenced by name.

```
ChromaAnimationAPI.MultiplyIntensityAllFramesRGBName(string path, int red, int
green, int blue);
```

MultiplyIntensityAllFramesRGBNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.MultiplyIntensityAllFramesRGBNameD(string path,
double red, double green, double blue);
```

MultiplyIntensityColor

Multiply the specific frame by the RBG color intensity. Animation is referenced by id.

```
ChromaAnimationAPI.MultiplyIntensityColor(int animationId, int frameId, int
color);
```

MultiplyIntensityColorAllFrames

Multiply all frames by the RBG color intensity. Animation is referenced by id.

```
ChromaAnimationAPI.MultiplyIntensityColorAllFrames(int animationId, int color);
```

MultiplyIntensityColorAllFramesName

Multiply all frames by the RBG color intensity. Animation is referenced by name.

```
ChromaAnimationAPI.MultiplyIntensityColorAllFramesName(string path, int color);
```

MultiplyIntensityColorAllFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.MultiplyIntensityColorAllFramesNameD(string
path, double color);
```

MultiplyIntensityColorName

Multiply the specific frame by the RBG color intensity. Animation is referenced by name.

```
ChromaAnimationAPI.MultiplyIntensityColorName(string path, int frameId, int
color);
```

MultiplyIntensityColorNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.MultiplyIntensityColorNameD(string path, double
frameId, double color);
```

MultiplyIntensityName

Multiply all the colors in the frame by the intensity value. The valid the intensity range is from 0.0 to 255.0. RGB components are multiplied equally. An intensity of 0.5 would half the color value. Black colors in the frame will not be affected by this method.

```
ChromaAnimationAPI.MultiplyIntensityName(string path, int frameId, float
intensity);
```

MultiplyIntensityNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.MultiplyIntensityNameD(string path, double
frameId, double intensity);
```

MultiplyIntensityRGB

Multiply the specific frame by the RBG color intensity. Animation is referenced by id.

```
ChromaAnimationAPI.MultiplyIntensityRGB(int animationId, int frameId, int red, int
green, int blue);
```

MultiplyIntensityRGBName

Multiply the specific frame by the RBG color intensity. Animation is referenced by name.

```
ChromaAnimationAPI.MultiplyIntensityRGBName(string path, int frameId, int red, int
green, int blue);
```

MultiplyIntensityRGBNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.MultiplyIntensityRGBNameD(string path, double
frameId, double red, double green, double blue);
```

MultiplyNonZeroTargetColorLerp

Multiply the specific frame by the color lerp result between color 1 and 2 using the frame color value as the `t` value. Animation is referenced by id.

```
ChromaAnimationAPI.MultiplyNonZeroTargetColorLerp(int animationId, int frameId,
int color1, int color2);
```

MultiplyNonZeroTargetColorLerpAllFrames

Multiply all frames by the color lerp result between color 1 and 2 using the frame color value as the `t` value. Animation is referenced by id.

```
ChromaAnimationAPI.MultiplyNonZeroTargetColorLerpAllFrames(int animationId, int
color1, int color2);
```

MultiplyNonZeroTargetColorLerpAllFramesName

Multiply all frames by the color lerp result between color 1 and 2 using the frame color value as the `t` value. Animation is referenced by name.

```
ChromaAnimationAPI.MultiplyNonZeroTargetColorLerpAllFramesName(string path, int
color1, int color2);
```

MultiplyNonZeroTargetColorLerpAllFramesNameD

D suffix for limited data types.

```
double result =
ChromaAnimationAPI.MultiplyNonZeroTargetColorLerpAllFramesNameD(string path,
double color1, double color2);
```

MultiplyNonZeroTargetColorLerpAllFramesRGB

Multiply the specific frame by the color lerp result between RGB 1 and 2 using the frame color value as the `t` value. Animation is referenced by id.

```
ChromaAnimationAPI.MultiplyNonZeroTargetColorLerpAllFramesRGB(int animationId, int
red1, int green1, int blue1, int red2, int green2, int blue2);
```

MultiplyNonZeroTargetColorLerpAllFramesRGBName

Multiply the specific frame by the color lerp result between RGB 1 and 2 using the frame color value as the `t` value. Animation is referenced by name.

```
ChromaAnimationAPI.MultiplyNonZeroTargetColorLerpAllFramesRGBName(string path, int
red1, int green1, int blue1, int red2, int green2, int blue2);
```

MultiplyNonZeroTargetColorLerpAllFramesRGBNameD

D suffix for limited data types.

```
double result =
ChromaAnimationAPI.MultiplyNonZeroTargetColorLerpAllFramesRGBNameD(string path,
double red1, double green1, double blue1, double red2, double green2, double
blue2);
```

MultiplyTargetColorLerp

Multiply the specific frame by the color lerp result between color 1 and 2 using the frame color value as the `t` value. Animation is referenced by id.

```
ChromaAnimationAPI.MultiplyTargetColorLerp(int animationId, int frameId, int
color1, int color2);
```

MultiplyTargetColorLerpAllFrames

Multiply all frames by the color lerp result between color 1 and 2 using the frame color value as the `t` value. Animation is referenced by id.

```
ChromaAnimationAPI.MultiplyTargetColorLerpAllFrames(int animationId, int color1,
int color2);
```

MultiplyTargetColorLerpAllFramesName

Multiply all frames by the color lerp result between color 1 and 2 using the frame color value as the `t` value. Animation is referenced by name.

```
ChromaAnimationAPI.MultiplyTargetColorLerpAllFramesName(string path, int color1,
int color2);
```

MultiplyTargetColorLerpAllFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.MultiplyTargetColorLerpAllFramesNameD(string
path, double color1, double color2);
```

MultiplyTargetColorLerpAllFramesRGB

Multiply all frames by the color lerp result between RGB 1 and 2 using the frame color value as the `t` value. Animation is referenced by id.

```
ChromaAnimationAPI.MultiplyTargetColorLerpAllFramesRGB(int animationId, int red1,
int green1, int blue1, int red2, int green2, int blue2);
```

MultiplyTargetColorLerpAllFramesRGBName

Multiply all frames by the color lerp result between RGB 1 and 2 using the frame color value as the `t` value. Animation is referenced by name.

```
ChromaAnimationAPI.MultiplyTargetColorLerpAllFramesRGBName(string path, int red1,
int green1, int blue1, int red2, int green2, int blue2);
```

MultiplyTargetColorLerpAllFramesRGBNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.MultiplyTargetColorLerpAllFramesRGBNameD(string
path, double red1, double green1, double blue1, double red2, double green2, double
blue2);
```

MultiplyTargetColorLerpName

Multiply the specific frame by the color lerp result between color 1 and 2 using the frame color value as the `t` value. Animation is referenced by name.

```
ChromaAnimationAPI.MultiplyTargetColorLerpName(string path, int frameId, int
color1, int color2);
```

OffsetColors

Offset all colors in the frame using the RGB offset. Use the range of -255 to 255 for red, green, and blue parameters. Negative values remove color. Positive values add color.

```
ChromaAnimationAPI.OffsetColors(int animationId, int frameId, int red, int green,
int blue);
```

OffsetColorsAllFrames

Offset all colors for all frames using the RGB offset. Use the range of -255 to 255 for red, green, and blue parameters. Negative values remove color. Positive values add color.

```
ChromaAnimationAPI.OffsetColorsAllFrames(int animationId, int red, int green, int
blue);
```

OffsetColorsAllFramesName

Offset all colors for all frames using the RGB offset. Use the range of -255 to 255 for red, green, and blue parameters. Negative values remove color. Positive values add color.

```
ChromaAnimationAPI.OffsetColorsAllFramesName(string path, int red, int green, int
blue);
```

OffsetColorsAllFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.OffsetColorsAllFramesNameD(string path, double
red, double green, double blue);
```

OffsetColorsName

Offset all colors in the frame using the RGB offset. Use the range of -255 to 255 for red, green, and blue parameters. Negative values remove color. Positive values add color.

```
ChromaAnimationAPI.OffsetColorsName(string path, int frameId, int red, int green,
int blue);
```

OffsetColorsNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.OffsetColorsNameD(string path, double frameId,
double red, double green, double blue);
```

OffsetNonZeroColors

This method will only update colors in the animation that are not already set to black. Offset a subset of colors in the frame using the RGB offset. Use the range of -255 to 255 for red, green, and blue parameters. Negative values remove color. Positive values add color.

```
ChromaAnimationAPI.OffsetNonZeroColors(int animationId, int frameId, int red, int
green, int blue);
```

OffsetNonZeroColorsAllFrames

This method will only update colors in the animation that are not already set to black. Offset a subset of colors for all frames using the RGB offset. Use the range of -255 to 255 for red, green, and blue parameters. Negative values remove color. Positive values add color.

```
ChromaAnimationAPI.OffsetNonZeroColorsAllFrames(int animationId, int red, int
green, int blue);
```

OffsetNonZeroColorsAllFramesName

This method will only update colors in the animation that are not already set to black. Offset a subset of colors for all frames using the RGB offset. Use the range of -255 to 255 for red, green, and blue parameters. Negative values remove color. Positive values add color.

```
ChromaAnimationAPI.OffsetNonZeroColorsAllFramesName(string path, int red, int
green, int blue);
```

OffsetNonZeroColorsAllFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.OffsetNonZeroColorsAllFramesNameD(string path,
double red, double green, double blue);
```

OffsetNonZeroColorsName

This method will only update colors in the animation that are not already set to black. Offset a subset of colors in the frame using the RGB offset. Use the range of -255 to 255 for red, green, and blue parameters. Negative values remove color. Positive values add color.

```
ChromaAnimationAPI.OffsetNonZeroColorsName(string path, int frameId, int red, int
green, int blue);
```

OffsetNonZeroColorsNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.OffsetNonZeroColorsNameD(string path, double
frameId, double red, double green, double blue);
```

OpenAnimation

Opens a `Chroma` animation file so that it can be played. Returns an animation id >= 0 upon success. Returns negative one if there was a failure. The animation id is used in most of the API methods.

```
int result = ChromaAnimationAPI.OpenAnimation(string path);
```

OpenAnimationD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.OpenAnimationD(string path);
```

OpenAnimationFromMemory

Opens a `Chroma` animation data from memory so that it can be played. `Data` is a pointer to BYTE array of the loaded animation in memory. `Name` will be assigned to the animation when loaded. Returns an animation id >= 0 upon success. Returns negative one if there was a failure. The animation id is used in most of the API methods.

```
int result = ChromaAnimationAPI.OpenAnimationFromMemory(byte[] data, string name);
```

OpenEditorDialog

Opens a `Chroma` animation file with the `.chroma` extension. Returns zero upon success. Returns negative one if there was a failure.

```
int result = ChromaAnimationAPI.OpenEditorDialog(string path);
```

OpenEditorDialogAndPlay

Open the named animation in the editor dialog and play the animation at start.

```
int result = ChromaAnimationAPI.OpenEditorDialogAndPlay(string path);
```

OpenEditorDialogAndPlayD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.OpenEditorDialogAndPlayD(string path);
```

OpenEditorDialogD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.OpenEditorDialogD(string path);
```

OverrideFrameDuration

Sets the `duration` for all grames in the `Chroma` animation to the `duration` parameter. Returns the animation id upon success. Returns negative one upon failure.

```
int result = ChromaAnimationAPI.OverrideFrameDuration(int animationId, float
duration);
```

OverrideFrameDurationD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.OverrideFrameDurationD(double animationId,
double duration);
```

OverrideFrameDurationName

Override the duration of all frames with the `duration` value. Animation is referenced by name.

```
ChromaAnimationAPI.OverrideFrameDurationName(string path, float duration);
```

PauseAnimation

Pause the current animation referenced by id.

```
ChromaAnimationAPI.PauseAnimation(int animationId);
```

PauseAnimationName

Pause the current animation referenced by name.

```
ChromaAnimationAPI.PauseAnimationName(string path);
```

PauseAnimationNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.PauseAnimationNameD(string path);
```

PlayAnimation

Plays the `Chroma` animation. This will load the animation, if not loaded previously. Returns the animation id upon success. Returns negative one upon failure.

```
int result = ChromaAnimationAPI.PlayAnimation(int animationId);
```

PlayAnimationD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.PlayAnimationD(double animationId);
```

PlayAnimationFrame

`PluginPlayAnimationFrame` automatically handles initializing the `ChromaSDK`. The method will play the animation given the `animationId` with looping `on` or `off` starting at the `frameId`.

```
ChromaAnimationAPI.PlayAnimationFrame(int animationId, int frameId, bool loop);
```

PlayAnimationFrameName

`PluginPlayAnimationFrameName` automatically handles initializing the `ChromaSDK`. The named `.chroma` animation file will be automatically opened. The animation will play with looping `on` or `off` starting at the `frameId`.

```
ChromaAnimationAPI.PlayAnimationFrameName(string path, int frameId, bool loop);
```

PlayAnimationFrameNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.PlayAnimationFrameNameD(string path, double
frameId, double loop);
```

PlayAnimationLoop

`PluginPlayAnimationLoop` automatically handles initializing the `ChromaSDK`. The method will play the animation given the `animationId` with looping `on` or `off`.

```
ChromaAnimationAPI.PlayAnimationLoop(int animationId, bool loop);
```

PlayAnimationName

`PluginPlayAnimationName` automatically handles initializing the `ChromaSDK`. The named `.chroma` animation file will be automatically opened. The animation will play with looping `on` or `off`.

```
ChromaAnimationAPI.PlayAnimationName(string path, bool loop);
```

PlayAnimationNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.PlayAnimationNameD(string path, double loop);
```

PlayComposite

`PluginPlayComposite` automatically handles initializing the `ChromaSDK`. The named animation files for the `.chroma` set will be automatically opened. The set of animations will play with looping `on` or `off`.

```
ChromaAnimationAPI.PlayComposite(string name, bool loop);
```

PlayCompositeD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.PlayCompositeD(string name, double loop);
```

PreviewFrame

Displays the `Chroma` animation frame on `Chroma` hardware given the `frameId`. Returns the animation id upon success. Returns negative one upon failure.

```
int result = ChromaAnimationAPI.PreviewFrame(int animationId, int frameId);
```

PreviewFrameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.PreviewFrameD(double animationId, double
frameId);
```

PreviewFrameName

Displays the `Chroma` animation frame on `Chroma` hardware given the `frameId`. Animaton is referenced by name.

```
ChromaAnimationAPI.PreviewFrameName(string path, int frameId);
```

ReduceFrames

Reduce the frames of the animation by removing every nth element. Animation is referenced by id.

```
ChromaAnimationAPI.ReduceFrames(int animationId, int n);
```

ReduceFramesName

Reduce the frames of the animation by removing every nth element. Animation is referenced by name.

```
ChromaAnimationAPI.ReduceFramesName(string path, int n);
```

ReduceFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.ReduceFramesNameD(string path, double n);
```

ResetAnimation

Resets the `Chroma` animation to 1 blank frame. Returns the animation id upon success. Returns negative one upon failure.

```
int result = ChromaAnimationAPI.ResetAnimation(int animationId);
```

ResumeAnimation

Resume the animation with loop `ON` or `OFF` referenced by id.

```
ChromaAnimationAPI.ResumeAnimation(int animationId, bool loop);
```

ResumeAnimationName

Resume the animation with loop `ON` or `OFF` referenced by name.

```
ChromaAnimationAPI.ResumeAnimationName(string path, bool loop);
```

ResumeAnimationNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.ResumeAnimationNameD(string path, double loop);
```

Reverse

Reverse the animation frame order of the `Chroma` animation. Returns the animation id upon success. Returns negative one upon failure. Animation is referenced by id.

```
int result = ChromaAnimationAPI.Reverse(int animationId);
```

ReverseAllFrames

Reverse the animation frame order of the `Chroma` animation. Animation is referenced by id.

```
ChromaAnimationAPI.ReverseAllFrames(int animationId);
```

ReverseAllFramesName

Reverse the animation frame order of the `Chroma` animation. Animation is referenced by name.

```
ChromaAnimationAPI.ReverseAllFramesName(string path);
```

ReverseAllFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.ReverseAllFramesNameD(string path);
```

SaveAnimation

Save the animation referenced by id to the path specified.

```
int result = ChromaAnimationAPI.SaveAnimation(int animationId, string path);
```

SaveAnimationName

Save the named animation to the target path specified.

```
int result = ChromaAnimationAPI.SaveAnimationName(string sourceAnimation, string
targetAnimation);
```

Set1DColor

Set the animation color for a frame given the `1D led`. The `led` should be greater than or equal to 0 and less than the `MaxLeds`. The animation is referenced by id.

```
ChromaAnimationAPI.Set1DColor(int animationId, int frameId, int led, int color);
```

Set1DColorName

Set the animation color for a frame given the `1D led`. The `led` should be greater than or equal to 0 and less than the `MaxLeds`. The animation is referenced by name.

```
ChromaAnimationAPI.Set1DColorName(string path, int frameId, int led, int color);
```

Set1DColorNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.Set1DColorNameD(string path, double frameId,
double led, double color);
```

Set2DColor

Set the animation color for a frame given the `2D` `row` and `column`. The `row` should be greater than or equal to 0 and less than the `MaxRow`. The `column` should be greater than or equal to 0 and less than the `MaxColumn`. The animation is referenced by id.

```
ChromaAnimationAPI.Set2DColor(int animationId, int frameId, int row, int column,
int color);
```

Set2DColorName

Set the animation color for a frame given the `2D` `row` and `column`. The `row` should be greater than or equal to 0 and less than the `MaxRow`. The `column` should be greater than or equal to 0 and less than the `MaxColumn`. The animation is referenced by name.

```
ChromaAnimationAPI.Set2DColorName(string path, int frameId, int row, int column,
int color);
```

Set2DColorNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.Set2DColorNameD(string path, double frameId,
double rowColumnIndex, double color);
```

SetChromaCustomColorAllFrames

When custom color is set, the custom key mode will be used. The animation is referenced by id.

```
ChromaAnimationAPI.SetChromaCustomColorAllFrames(int animationId);
```

SetChromaCustomColorAllFramesName

When custom color is set, the custom key mode will be used. The animation is referenced by name.

```
ChromaAnimationAPI.SetChromaCustomColorAllFramesName(string path);
```

SetChromaCustomColorAllFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.SetChromaCustomColorAllFramesNameD(string
path);
```

SetChromaCustomFlag

Set the Chroma custom key color flag on all frames. `True` changes the layout from grid to key. `True` changes the layout from key to grid. Animation is referenced by id.

```
ChromaAnimationAPI.SetChromaCustomFlag(int animationId, bool flag);
```

SetChromaCustomFlagName

Set the Chroma custom key color flag on all frames. `True` changes the layout from grid to key. `True` changes the layout from key to grid. Animation is referenced by name.

```
ChromaAnimationAPI.SetChromaCustomFlagName(string path, bool flag);
```

SetChromaCustomFlagNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.SetChromaCustomFlagNameD(string path, double
flag);
```

SetCurrentFrame

Set the current frame of the animation referenced by id.

```
ChromaAnimationAPI.SetCurrentFrame(int animationId, int frameId);
```

SetCurrentFrameName

Set the current frame of the animation referenced by name.

```
ChromaAnimationAPI.SetCurrentFrameName(string path, int frameId);
```

SetCurrentFrameNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.SetCurrentFrameNameD(string path, double
frameId);
```

SetCustomColorFlag2D

Set the custom alpha flag on the color array

```
int result = ChromaAnimationAPI.SetCustomColorFlag2D(int device, int[] colors);
```

SetDevice

Changes the `deviceType` and `device` of a `Chroma` animation. If the device is changed, the `Chroma` animation will be reset with 1 blank frame. Returns the animation id upon success. Returns negative one upon failure.

```
int result = ChromaAnimationAPI.SetDevice(int animationId, int deviceType, int
device);
```

SetEffect

SetEffect will display the referenced effect id.

```
int result = ChromaAnimationAPI.SetEffect(Guid effectId);
```

SetEffectCustom1D

SetEffectCustom1D will display the referenced colors immediately

```
int result = ChromaAnimationAPI.SetEffectCustom1D(int device, int[] colors);
```

SetEffectCustom2D

SetEffectCustom2D will display the referenced colors immediately.

```
int result = ChromaAnimationAPI.SetEffectCustom2D(int device, int[] colors);
```

SetEffectKeyboardCustom2D

SetEffectKeyboardCustom2D will display the referenced custom keyboard colors immediately. Colors represent a visual grid layout. Keys represent the hotkeys for any layout.

```
int result = ChromaAnimationAPI.SetEffectKeyboardCustom2D(int device, int[]
colors, int[] keys);
```

SetIdleAnimation

When the idle animation is used, the named animation will play when no other animations are playing. Reference the animation by id.

```
ChromaAnimationAPI.SetIdleAnimation(int animationId);
```

SetIdleAnimationName

When the idle animation is used, the named animation will play when no other animations are playing. Reference the animation by name.

```
ChromaAnimationAPI.SetIdleAnimationName(string path);
```

SetKeyColor

Set animation key to a static color for the given frame.

```
ChromaAnimationAPI.SetKeyColor(int animationId, int frameId, int rzkey, int
color);
```

SetKeyColorAllFrames

Set the key to the specified key color for all frames. Animation is referenced by id.

```
ChromaAnimationAPI.SetKeyColorAllFrames(int animationId, int rzkey, int color);
```

SetKeyColorAllFramesName

Set the key to the specified key color for all frames. Animation is referenced by name.

```
ChromaAnimationAPI.SetKeyColorAllFramesName(string path, int rzkey, int color);
```

SetKeyColorAllFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.SetKeyColorAllFramesNameD(string path, double
rzkey, double color);
```

SetKeyColorAllFramesRGB

Set the key to the specified key color for all frames. Animation is referenced by id.

```
ChromaAnimationAPI.SetKeyColorAllFramesRGB(int animationId, int rzkey, int red,
int green, int blue);
```

SetKeyColorAllFramesRGBName

Set the key to the specified key color for all frames. Animation is referenced by name.

```
ChromaAnimationAPI.SetKeyColorAllFramesRGBName(string path, int rzkey, int red,
int green, int blue);
```

SetKeyColorAllFramesRGBNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.SetKeyColorAllFramesRGBNameD(string path,
double rzkey, double red, double green, double blue);
```

SetKeyColorName

Set animation key to a static color for the given frame.

```
ChromaAnimationAPI.SetKeyColorName(string path, int frameId, int rzkey, int
color);
```

SetKeyColorNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.SetKeyColorNameD(string path, double frameId,
double rzkey, double color);
```

SetKeyColorRGB

Set the key to the specified key color for the specified frame. Animation is referenced by id.

```
ChromaAnimationAPI.SetKeyColorRGB(int animationId, int frameId, int rzkey, int
red, int green, int blue);
```

SetKeyColorRGBName

Set the key to the specified key color for the specified frame. Animation is referenced by name.

```
ChromaAnimationAPI.SetKeyColorRGBName(string path, int frameId, int rzkey, int
red, int green, int blue);
```

SetKeyColorRGBNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.SetKeyColorRGBNameD(string path, double
frameId, double rzkey, double red, double green, double blue);
```

SetKeyNonZeroColor

Set animation key to a static color for the given frame if the existing color is not already black.

```
ChromaAnimationAPI.SetKeyNonZeroColor(int animationId, int frameId, int rzkey, int
color);
```

SetKeyNonZeroColorName

Set animation key to a static color for the given frame if the existing color is not already black.

```
ChromaAnimationAPI.SetKeyNonZeroColorName(string path, int frameId, int rzkey, int
color);
```

SetKeyNonZeroColorNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.SetKeyNonZeroColorNameD(string path, double
frameId, double rzkey, double color);
```

SetKeyNonZeroColorRGB

Set the key to the specified key color for the specified frame where color is not black. Animation is referenced by id.

```
ChromaAnimationAPI.SetKeyNonZeroColorRGB(int animationId, int frameId, int rzkey,
int red, int green, int blue);
```

SetKeyNonZeroColorRGBName

Set the key to the specified key color for the specified frame where color is not black. Animation is referenced by name.

```
ChromaAnimationAPI.SetKeyNonZeroColorRGBName(string path, int frameId, int rzkey,
int red, int green, int blue);
```

SetKeyNonZeroColorRGBNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.SetKeyNonZeroColorRGBNameD(string path, double
frameId, double rzkey, double red, double green, double blue);
```

SetKeyRowColumnColorName

Set animation key by row and column to a static color for the given frame.

```
ChromaAnimationAPI.SetKeyRowColumnColorName(string path, int frameId, int row, int
column, int color);
```

SetKeysColor

Set an array of animation keys to a static color for the given frame. Animation is referenced by id.

```
ChromaAnimationAPI.SetKeysColor(int animationId, int frameId, int[] rzkeys, int
keyCount, int color);
```

SetKeysColorAllFrames

Set an array of animation keys to a static color for all frames. Animation is referenced by id.

```
ChromaAnimationAPI.SetKeysColorAllFrames(int animationId, int[] rzkeys, int
keyCount, int color);
```

SetKeysColorAllFramesName

Set an array of animation keys to a static color for all frames. Animation is referenced by name.

```
ChromaAnimationAPI.SetKeysColorAllFramesName(string path, int[] rzkeys, int
keyCount, int color);
```

SetKeysColorAllFramesRGB

Set an array of animation keys to a static color for all frames. Animation is referenced by id.

```
ChromaAnimationAPI.SetKeysColorAllFramesRGB(int animationId, int[] rzkeys, int
keyCount, int red, int green, int blue);
```

SetKeysColorAllFramesRGBName

Set an array of animation keys to a static color for all frames. Animation is referenced by name.

```
ChromaAnimationAPI.SetKeysColorAllFramesRGBName(string path, int[] rzkeys, int
keyCount, int red, int green, int blue);
```

SetKeysColorName

Set an array of animation keys to a static color for the given frame.

```
ChromaAnimationAPI.SetKeysColorName(string path, int frameId, int[] rzkeys, int
keyCount, int color);
```

SetKeysColorRGB

Set an array of animation keys to a static color for the given frame. Animation is referenced by id.

```
ChromaAnimationAPI.SetKeysColorRGB(int animationId, int frameId, int[] rzkeys, int
keyCount, int red, int green, int blue);
```

SetKeysColorRGBName

Set an array of animation keys to a static color for the given frame. Animation is referenced by name.

```
ChromaAnimationAPI.SetKeysColorRGBName(string path, int frameId, int[] rzkeys, int
keyCount, int red, int green, int blue);
```

SetKeysNonZeroColor

Set an array of animation keys to a static color for the given frame if the existing color is not already black.

```
ChromaAnimationAPI.SetKeysNonZeroColor(int animationId, int frameId, int[] rzkeys,
int keyCount, int color);
```

SetKeysNonZeroColorAllFrames

Set an array of animation keys to a static color for the given frame where the color is not black. Animation is referenced by id.

```
ChromaAnimationAPI.SetKeysNonZeroColorAllFrames(int animationId, int[] rzkeys, int
keyCount, int color);
```

SetKeysNonZeroColorAllFramesName

Set an array of animation keys to a static color for all frames if the existing color is not already black. Reference animation by name.

```
ChromaAnimationAPI.SetKeysNonZeroColorAllFramesName(string path, int[] rzkeys, int
keyCount, int color);
```

SetKeysNonZeroColorName

Set an array of animation keys to a static color for the given frame if the existing color is not already black. Reference animation by name.

```
ChromaAnimationAPI.SetKeysNonZeroColorName(string path, int frameId, int[] rzkeys,
int keyCount, int color);
```

SetKeysNonZeroColorRGB

Set an array of animation keys to a static color for the given frame where the color is not black. Animation is referenced by id.

```
ChromaAnimationAPI.SetKeysNonZeroColorRGB(int animationId, int frameId, int[]
rzkeys, int keyCount, int red, int green, int blue);
```

SetKeysNonZeroColorRGBName

Set an array of animation keys to a static color for the given frame where the color is not black. Animation is referenced by name.

```
ChromaAnimationAPI.SetKeysNonZeroColorRGBName(string path, int frameId, int[]
rzkeys, int keyCount, int red, int green, int blue);
```

SetKeysZeroColor

Set an array of animation keys to a static color for the given frame where the color is black. Animation is referenced by id.

```
ChromaAnimationAPI.SetKeysZeroColor(int animationId, int frameId, int[] rzkeys,
int keyCount, int color);
```

SetKeysZeroColorAllFrames

Set an array of animation keys to a static color for all frames where the color is black. Animation is referenced by id.

```
ChromaAnimationAPI.SetKeysZeroColorAllFrames(int animationId, int[] rzkeys, int
keyCount, int color);
```

SetKeysZeroColorAllFramesName

Set an array of animation keys to a static color for all frames where the color is black. Animation is referenced by name.

```
ChromaAnimationAPI.SetKeysZeroColorAllFramesName(string path, int[] rzkeys, int
keyCount, int color);
```

SetKeysZeroColorAllFramesRGB

Set an array of animation keys to a static color for all frames where the color is black. Animation is referenced by id.

```
ChromaAnimationAPI.SetKeysZeroColorAllFramesRGB(int animationId, int[] rzkeys, int
keyCount, int red, int green, int blue);
```

SetKeysZeroColorAllFramesRGBName

Set an array of animation keys to a static color for all frames where the color is black. Animation is referenced by name.

```
ChromaAnimationAPI.SetKeysZeroColorAllFramesRGBName(string path, int[] rzkeys, int
keyCount, int red, int green, int blue);
```

SetKeysZeroColorName

Set an array of animation keys to a static color for the given frame where the color is black. Animation is referenced by name.

```
ChromaAnimationAPI.SetKeysZeroColorName(string path, int frameId, int[] rzkeys,
int keyCount, int color);
```

SetKeysZeroColorRGB

Set an array of animation keys to a static color for the given frame where the color is black. Animation is referenced by id.

```
ChromaAnimationAPI.SetKeysZeroColorRGB(int animationId, int frameId, int[] rzkeys,
int keyCount, int red, int green, int blue);
```

SetKeysZeroColorRGBName

Set an array of animation keys to a static color for the given frame where the color is black. Animation is referenced by name.

```
ChromaAnimationAPI.SetKeysZeroColorRGBName(string path, int frameId, int[] rzkeys,
int keyCount, int red, int green, int blue);
```

SetKeyZeroColor

Set animation key to a static color for the given frame where the color is black. Animation is referenced by id.

```
ChromaAnimationAPI.SetKeyZeroColor(int animationId, int frameId, int rzkey, int
color);
```

SetKeyZeroColorName

Set animation key to a static color for the given frame where the color is black. Animation is referenced by name.

```
ChromaAnimationAPI.SetKeyZeroColorName(string path, int frameId, int rzkey, int
color);
```

SetKeyZeroColorNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.SetKeyZeroColorNameD(string path, double
frameId, double rzkey, double color);
```

SetKeyZeroColorRGB

Set animation key to a static color for the given frame where the color is black. Animation is referenced by id.

```
ChromaAnimationAPI.SetKeyZeroColorRGB(int animationId, int frameId, int rzkey, int
red, int green, int blue);
```

SetKeyZeroColorRGBName

Set animation key to a static color for the given frame where the color is black. Animation is referenced by name.

```
ChromaAnimationAPI.SetKeyZeroColorRGBName(string path, int frameId, int rzkey, int
red, int green, int blue);
```

SetKeyZeroColorRGBNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.SetKeyZeroColorRGBNameD(string path, double
frameId, double rzkey, double red, double green, double blue);
```

SetLogDelegate

Invokes the setup for a debug logging callback so that `stdout` is redirected to the callback. This is used by `Unity` so that debug messages can appear in the console window.

```
ChromaAnimationAPI.SetLogDelegate(IntPtr fp);
```

SetStaticColor

Sets the target device to the static color.

```
ChromaAnimationAPI.SetStaticColor(int deviceType, int device, int color);
```

SetStaticColorAll

Sets all devices to the static color.

```
ChromaAnimationAPI.SetStaticColorAll(int color);
```

StaticColor

Sets the target device to the static color.

```
ChromaAnimationAPI.StaticColor(int deviceType, int device, int color);
```

StaticColorAll

Sets all devices to the static color.

```
ChromaAnimationAPI.StaticColorAll(int color);
```

StaticColorD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.StaticColorD(double deviceType, double device,
double color);
```

StopAll

`PluginStopAll` will automatically stop all animations that are playing.

```
ChromaAnimationAPI.StopAll();
```

StopAnimation

Stops animation playback if in progress. Returns the animation id upon success. Returns negative one upon failure.

```
int result = ChromaAnimationAPI.StopAnimation(int animationId);
```

StopAnimationD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.StopAnimationD(double animationId);
```

StopAnimationName

`PluginStopAnimationName` automatically handles initializing the `ChromaSDK`. The named `.chroma` animation file will be automatically opened. The animation will stop if playing.

```
ChromaAnimationAPI.StopAnimationName(string path);
```

StopAnimationNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.StopAnimationNameD(string path);
```

StopAnimationType

`PluginStopAnimationType` automatically handles initializing the `ChromaSDK`. If any animation is playing for the `deviceType` and `device` combination, it will be stopped.

```
ChromaAnimationAPI.StopAnimationType(int deviceType, int device);
```

StopAnimationTypeD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.StopAnimationTypeD(double deviceType, double
device);
```

StopComposite

`PluginStopComposite` automatically handles initializing the `ChromaSDK`. The named animation files for the `.chroma` set will be automatically opened. The set of animations will be stopped if playing.

```
ChromaAnimationAPI.StopComposite(string name);
```

StopCompositeD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.StopCompositeD(string name);
```

SubtractColor

Return color1 - color2

```
int result = ChromaAnimationAPI.SubtractColor(int color1, int color2);
```

SubtractNonZeroAllKeys

Subtract the source color from the target color for the frame where the target color is not black. Source and target are referenced by id.

```
ChromaAnimationAPI.SubtractNonZeroAllKeys(int sourceAnimationId, int
targetAnimationId, int frameId);
```

SubtractNonZeroAllKeysAllFrames

Subtract the source color from the target color for all frames where the target color is not black. Source and target are referenced by id.

```
ChromaAnimationAPI.SubtractNonZeroAllKeysAllFrames(int sourceAnimationId, int
targetAnimationId);
```

SubtractNonZeroAllKeysAllFramesName

Subtract the source color from the target color for all frames where the target color is not black. Source and target are referenced by name.

```
ChromaAnimationAPI.SubtractNonZeroAllKeysAllFramesName(string sourceAnimation,
string targetAnimation);
```

SubtractNonZeroAllKeysAllFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.SubtractNonZeroAllKeysAllFramesNameD(string
sourceAnimation, string targetAnimation);
```

SubtractNonZeroAllKeysAllFramesOffset

Subtract the source color from the target color for all frames where the target color is not black starting at offset for the length of the source. Source and target are referenced by id.

```
ChromaAnimationAPI.SubtractNonZeroAllKeysAllFramesOffset(int sourceAnimationId,
int targetAnimationId, int offset);
```

SubtractNonZeroAllKeysAllFramesOffsetName

Subtract the source color from the target color for all frames where the target color is not black starting at offset for the length of the source. Source and target are referenced by name.

```
ChromaAnimationAPI.SubtractNonZeroAllKeysAllFramesOffsetName(string
sourceAnimation, string targetAnimation, int offset);
```

SubtractNonZeroAllKeysAllFramesOffsetNameD

D suffix for limited data types.

```
double result =
ChromaAnimationAPI.SubtractNonZeroAllKeysAllFramesOffsetNameD(string
sourceAnimation, string targetAnimation, double offset);
```

SubtractNonZeroAllKeysName

Subtract the source color from the target color for the frame where the target color is not black. Source and target are referenced by name.

```
ChromaAnimationAPI.SubtractNonZeroAllKeysName(string sourceAnimation, string
targetAnimation, int frameId);
```

SubtractNonZeroAllKeysOffset

Subtract the source color from the target where color is not black for the source frame and target offset frame, reference source and target by id.

```
ChromaAnimationAPI.SubtractNonZeroAllKeysOffset(int sourceAnimationId, int
targetAnimationId, int frameId, int offset);
```

SubtractNonZeroAllKeysOffsetName

Subtract the source color from the target where color is not black for the source frame and target offset frame, reference source and target by name.

```
ChromaAnimationAPI.SubtractNonZeroAllKeysOffsetName(string sourceAnimation, string
targetAnimation, int frameId, int offset);
```

SubtractNonZeroAllKeysOffsetNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.SubtractNonZeroAllKeysOffsetNameD(string
sourceAnimation, string targetAnimation, double frameId, double offset);
```

SubtractNonZeroTargetAllKeysAllFrames

Subtract the source color from the target color where the target color is not black for all frames. Reference source and target by id.

```
ChromaAnimationAPI.SubtractNonZeroTargetAllKeysAllFrames(int sourceAnimationId,
int targetAnimationId);
```

SubtractNonZeroTargetAllKeysAllFramesName

Subtract the source color from the target color where the target color is not black for all frames. Reference source and target by name.

```
ChromaAnimationAPI.SubtractNonZeroTargetAllKeysAllFramesName(string
sourceAnimation, string targetAnimation);
```

SubtractNonZeroTargetAllKeysAllFramesNameD

D suffix for limited data types.

```
double result =
ChromaAnimationAPI.SubtractNonZeroTargetAllKeysAllFramesNameD(string
sourceAnimation, string targetAnimation);
```

SubtractNonZeroTargetAllKeysAllFramesOffset

Subtract the source color from the target color where the target color is not black for all frames starting at the target offset for the length of the source. Reference source and target by id.

```
ChromaAnimationAPI.SubtractNonZeroTargetAllKeysAllFramesOffset(int
sourceAnimationId, int targetAnimationId, int offset);
```

SubtractNonZeroTargetAllKeysAllFramesOffsetName

Subtract the source color from the target color where the target color is not black for all frames starting at the target offset for the length of the source. Reference source and target by name.

```
ChromaAnimationAPI.SubtractNonZeroTargetAllKeysAllFramesOffsetName(string
sourceAnimation, string targetAnimation, int offset);
```

SubtractNonZeroTargetAllKeysAllFramesOffsetNameD

D suffix for limited data types.

```
double result =
ChromaAnimationAPI.SubtractNonZeroTargetAllKeysAllFramesOffsetNameD(string
sourceAnimation, string targetAnimation, double offset);
```

SubtractNonZeroTargetAllKeysOffset

Subtract the source color from the target color where the target color is not black from the source frame to the target offset frame. Reference source and target by id.

```
ChromaAnimationAPI.SubtractNonZeroTargetAllKeysOffset(int sourceAnimationId, int
targetAnimationId, int frameId, int offset);
```

SubtractNonZeroTargetAllKeysOffsetName

Subtract the source color from the target color where the target color is not black from the source frame to the target offset frame. Reference source and target by name.

```
ChromaAnimationAPI.SubtractNonZeroTargetAllKeysOffsetName(string sourceAnimation,
string targetAnimation, int frameId, int offset);
```

SubtractNonZeroTargetAllKeysOffsetNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.SubtractNonZeroTargetAllKeysOffsetNameD(string
sourceAnimation, string targetAnimation, double frameId, double offset);
```

SubtractThresholdColorsMinMaxAllFramesRGB

Subtract all frames with the min RGB color where the animation color is less than the min threshold AND with the max RGB color where the animation is more than the max threshold. Animation is referenced by id.

```
ChromaAnimationAPI.SubtractThresholdColorsMinMaxAllFramesRGB(int animationId, int
minThreshold, int minRed, int minGreen, int minBlue, int maxThreshold, int maxRed,
int maxGreen, int maxBlue);
```

SubtractThresholdColorsMinMaxAllFramesRGBName

Subtract all frames with the min RGB color where the animation color is less than the min threshold AND with the max RGB color where the animation is more than the max threshold. Animation is referenced by name.

```
ChromaAnimationAPI.SubtractThresholdColorsMinMaxAllFramesRGBName(string path, int
minThreshold, int minRed, int minGreen, int minBlue, int maxThreshold, int maxRed,
int maxGreen, int maxBlue);
```

SubtractThresholdColorsMinMaxAllFramesRGBNameD

D suffix for limited data types.

```
double result =
ChromaAnimationAPI.SubtractThresholdColorsMinMaxAllFramesRGBNameD(string path,
double minThreshold, double minRed, double minGreen, double minBlue, double
maxThreshold, double maxRed, double maxGreen, double maxBlue);
```

SubtractThresholdColorsMinMaxRGB

Subtract the specified frame with the min RGB color where the animation color is less than the min threshold AND with the max RGB color where the animation is more than the max threshold. Animation is referenced by id.

```
ChromaAnimationAPI.SubtractThresholdColorsMinMaxRGB(int animationId, int frameId,
int minThreshold, int minRed, int minGreen, int minBlue, int maxThreshold, int
maxRed, int maxGreen, int maxBlue);
```

SubtractThresholdColorsMinMaxRGBName

Subtract the specified frame with the min RGB color where the animation color is less than the min threshold AND with the max RGB color where the animation is more than the max threshold. Animation is referenced by name.

```
ChromaAnimationAPI.SubtractThresholdColorsMinMaxRGBName(string path, int frameId,
int minThreshold, int minRed, int minGreen, int minBlue, int maxThreshold, int
maxRed, int maxGreen, int maxBlue);
```

SubtractThresholdColorsMinMaxRGBNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.SubtractThresholdColorsMinMaxRGBNameD(string
path, int frameId, int minThreshold, int minRed, int minGreen, int minBlue, int
maxThreshold, int maxRed, int maxGreen, int maxBlue);
```

TrimEndFrames

Trim the end of the animation. The length of the animation will be the lastFrameId plus one. Reference the animation by id.

```
ChromaAnimationAPI.TrimEndFrames(int animationId, int lastFrameId);
```

TrimEndFramesName

Trim the end of the animation. The length of the animation will be the lastFrameId plus one. Reference the animation by name.

```
ChromaAnimationAPI.TrimEndFramesName(string path, int lastFrameId);
```

TrimEndFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.TrimEndFramesNameD(string path, double
lastFrameId);
```

TrimFrame

Remove the frame from the animation. Reference animation by id.

```
ChromaAnimationAPI.TrimFrame(int animationId, int frameId);
```

TrimFrameName

Remove the frame from the animation. Reference animation by name.

```
ChromaAnimationAPI.TrimFrameName(string path, int frameId);
```

TrimFrameNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.TrimFrameNameD(string path, double frameId);
```

TrimStartFrames

Trim the start of the animation starting at frame 0 for the number of frames. Reference the animation by id.

```
ChromaAnimationAPI.TrimStartFrames(int animationId, int numberOfFrames);
```

TrimStartFramesName

Trim the start of the animation starting at frame 0 for the number of frames. Reference the animation by name.

```
ChromaAnimationAPI.TrimStartFramesName(string path, int numberOfFrames);
```

TrimStartFramesNameD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.TrimStartFramesNameD(string path, double
numberOfFrames);
```

Uninit

Uninitializes the `ChromaSDK`. Returns 0 upon success. Returns negative one upon failure.

```
int result = ChromaAnimationAPI.Uninit();
```

UninitD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.UninitD();
```

UnloadAnimation

Unloads `Chroma` effects to free up resources. Returns the animation id upon success. Returns negative one upon failure. Reference the animation by id.

```
int result = ChromaAnimationAPI.UnloadAnimation(int animationId);
```

UnloadAnimationD

D suffix for limited data types.

```
double result = ChromaAnimationAPI.UnloadAnimationD(double animationId);
```

UnloadAnimationName

Unload the animation effects. Reference the animation by name.

```
ChromaAnimationAPI.UnloadAnimationName(string path);
```

UnloadComposite

Unload the the composite set of animation effects. Reference the animation by name.

```
ChromaAnimationAPI.UnloadComposite(string name);
```

UnloadLibrarySDK

Unload the Razer Chroma SDK Library before exiting the application.

```
ChromaAnimationAPI.UnloadLibrarySDK();
```

UnloadLibraryStreamingPlugin

Unload the Razer Chroma Streaming Plugin Library before exiting the application.

```
ChromaAnimationAPI.UnloadLibraryStreamingPlugin();
```

UpdateFrame

Updates the `frameId` of the `Chroma` animation referenced by id and sets the `duration` (in seconds). The `color` is expected to be an array of the dimensions for the `deviceType/device`. The `length` parameter is the size of the `color` array. For `EChromaSDKDevice1DEnum` the array size should be `MAX_LEDS`. For `EChromaSDKDevice2DEnum` the array size should be `MAX_ROW` times `MAX_COLUMN`. Keys are populated only for EChromaSDKDevice2DEnum::DE_Keyboard and EChromaSDKDevice2DEnum::DE_KeyboardExtended. Keys will only use the EChromaSDKDevice2DEnum::DE_Keyboard `MAX_ROW` times `MAX_COLUMN` keysLength.

```
int result = ChromaAnimationAPI.UpdateFrame(int animationId, int frameId, float
duration, int[] colors, int length, int[] keys, int keysLength);
```

UpdateFrameName

Update the `frameId` of the `Chroma` animation referenced by name and sets the `duration` (in seconds). The `color` is expected to be an array of the dimensions for the `deviceType/device`. The `length` parameter is the size of the `color` array. For `EChromaSDKDevice1DEnum` the array size should be `MAX_LEDS`. For `EChromaSDKDevice2DEnum` the array size should be `MAX_ROW` times `MAX_COLUMN`. Keys are populated only for EChromaSDKDevice2DEnum::DE_Keyboard and EChromaSDKDevice2DEnum::DE_KeyboardExtended. Keys will only use the EChromaSDKDevice2DEnum::DE_Keyboard `MAX_ROW` times `MAX_COLUMN` keysLength. Returns the animation id upon success. Returns negative one upon failure.

```
int result = ChromaAnimationAPI.UpdateFrameName(string path, int frameId, float
duration, int[] colors, int length, int[] keys, int keysLength);
```

UseForwardChromaEvents

On by default, `UseForwardChromaEvents` sends the animation name to `CoreSetEventName` automatically when `PlayAnimationName` is called.

```
ChromaAnimationAPI.UseForwardChromaEvents(bool flag);
```

UseIdleAnimation

When the idle animation flag is true, when no other animations are playing, the idle animation will be used. The idle animation will not be affected by the API calls to PluginIsPlaying, PluginStopAnimationType, PluginGetPlayingAnimationId, and PluginGetPlayingAnimationCount. Then the idle animation flag is false, the idle animation is disabled. `Device` uses `EChromaSDKDeviceEnum` enums.

```
ChromaAnimationAPI.UseIdleAnimation(int device, bool flag);
```

UseIdleAnimations

Set idle animation flag for all devices.

```
ChromaAnimationAPI.UseIdleAnimations(bool flag);
```

UsePreloading

Set preloading animation flag, which is set to true by default. Reference animation by id.

```
ChromaAnimationAPI.UsePreloading(int animationId, bool flag);
```

UsePreloadingName

Set preloading animation flag, which is set to true by default. Reference animation by name.

```
ChromaAnimationAPI.UsePreloadingName(string path, bool flag);
```