



第196章

工作流程

集成

本章介绍 DaVinci Resolve 的第三方工作流程集成和编解码器插件。

内容

DaVinci Resolve 中的工作流程集成（仅限 Studio 版本）	4109
创建工作流集成插件	4109
工作流程集成插件	4109
编辑分享	4109
演播室网络解决方案 (SNS)	4110
编解码器插件（仅限 Studio 版本）	4111
主要概念	4111

工作流程集成 DaVinci Resolve（仅限 Studio 版本）

DaVinci Resolve 允许第三方使用脚本语言创建自己的自定义界面插件。这使得 DaVinci Resolve 和其他软件程序之间的直接集成成为可能,可用于多种用途。多个集成插件可以同时处于活动状态。

安装后,可以通过转至工作区 > 工作流程集成并从下拉菜单中选择您的插件来在 DaVinci Resolve 中启用插件。

创建工作流程 集成插件

用户可以使用 Resolve Javascript 的 API 以及 Python 或 Lua 脚本编写自己的工作流集成插件（一个 Electron 应用程序）。有关如何创建工作流集成插件的更多信息,请转到“帮助”>“文档”>“开发人员”,然后打开“工作流集成”文件夹以获取技术详细信息和示例代码。

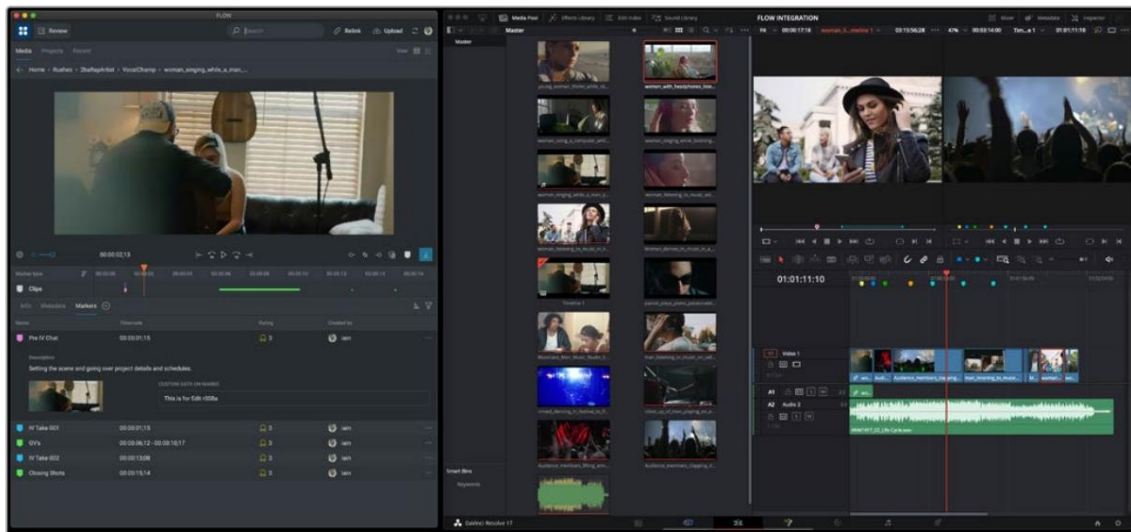
工作流程集成插件

现在可以使用工作流程集成插件通过 DaVinci Resolve 直接访问多个媒体资产管理 (MAM) 系统。

编辑分享

EditShare 创建了一个工作流程集成插件,允许 DaVinci Resolve 直接与其 FLOW 媒体管理系统连接。该插件允许您在 FLOW 中评论、搜索和预览媒体,而无需离开 DaVinci Resolve。您还可以在整个过程中上传修订版本、管理代理媒体并维护完整的元数据支持。

有关此插件以及 FLOW 如何与 DaVinci Resolve 配合使用的更多信息,请访问: <https://editshare.com/say-hello-to-flow-and-davinci-resolve-studio/>

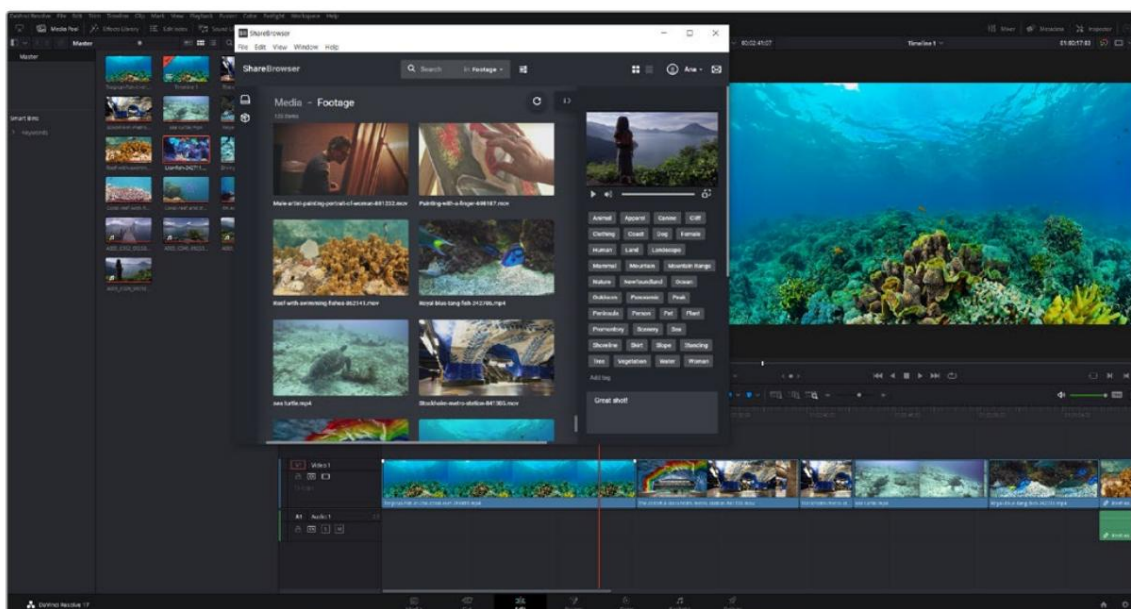


EditShare 的 FLOW 集成插件

演播室网络解决方案 (SNS)

Studio Network Solutions (SNS) 创建了 ShareBrowser 集成插件,用于连接 DaVinci Resolve 与其 ShareBrowser 媒体资产管理软件 (包含在 SNS EVO 媒体服务器中)。该插件允许您的团队搜索、标记、预览、评论、组织和导入媒体,而无需离开 DaVinci Resolve 界面。您的团队可以直接将媒体导入 DaVinci Resolve 项目,您输入的元数据会随媒体一起保留。

有关此插件以及 SNS 的高速服务器或云解决方案如何与 DaVinci Resolve 配合使用的更多信息,请访问:<https://www.studionetworksolutions.com/>。



SNS ShareBrowser 集成插件

编解码器插件（仅限 Studio 版本）

编解码器插件允许第三方安装新的编解码器以在“交付”页面中进行编码,而达芬奇 Resolve 主软件当前不支持这些编解码器。这为极其具体的交付成果打开了大门,这些交付成果通常需要通过多个程序才能交付。

主要概念

MainConcept Codec 插件允许您以多种方式渲染 DaVinci Resolve Studio 时间线
新编解码器:

AS-11 UK SD、AS-11 UK HD 以及随附的 XML 元数据文件,用于创建 AS-11 UK DPP
合规内容。

MainConcept 的软件 HEVC Main 和 Main 10 配置文件,允许 8 位/10 位的 H.265 文件
4:2:0/4:2:2,分辨率高达 8K。

MainConcept MXF 和 MP4,允许编码为索尼使用的本机相机格式
基于 XAVC/XDCAM 和 Panasonic P2 AVC 的摄像机。

有关 DaVinci Resolve 的 MainConcept 编解码器插件的更多信息,请访问:<https://www.mainconcept.com/blackmagic-plugins>



“交付”页面中的 DaVinci Resolve 选项的 MainConcept 编解码器插件

第197章

创建 DCTL LUT

本章介绍如何创建 DCTL LUT 以在 DaVinci Resolve 中执行您自己的自定义数学转换。

内容

关于DCTL	4113
DCTL语法	4113
一个简单的 DCT LUT 示例	4115
矩阵 DCT LUT 示例	4115
更复杂的 DCT LUT 示例	4116

关于DCTL

DCTL 文件实际上是 DaVinci Resolve 看到并应用的颜色转换脚本,就像任何其他 LUT 一样。其他 LUT 是使用插值近似图像变换的 1D 或 3D 值查找表,与此不同,DCTL 文件实际上由计算机代码组成,可使用您设计的数学函数组合直接变换图像。此外,DCTL 文件在工作站的 GPU 上本机运行,因此速度很快。

任何具有数学知识的人都可以制作并安装 DCTL。只需使用类似于 C 的语法 (下面将更详细地描述)输入转换代码到任何能够保存纯 ASCII 文本文件的文本编辑器中,并确保其名称以 “.dctl”结尾 (达芬奇颜色转换语言)文件扩展名。完成后,将文件移动到工作站的 LUT 目录。具体位置取决于您使用的操作系统:

在 Mac OS X 上:库/应用程序支持/Blackmagic Design/DaVinci Resolve/LUT/

在 Windows 上:C:\ProgramData\Blackmagic Design\DaVinci Resolve\Support\LUT

在 Linux 上: /home/resolve/LUT

当DaVinci Resolve启动时,假设.dctl的语法正确,它们将出现在DaVinci CTL子菜单内的“颜色”页面节点上下文菜单中。

DCTL语法

用户需要将 __DEVICE__ 放在他们编写的每个函数前面。例如:

```
__DEVICE__ float2 DoSomething()
```

主入口函数 (转换)应位于所有其他函数之后,并具有以下格式参数:

```
__DEVICE__ float3 变换 (浮点 p_R,浮点 p_G,浮点 p_B)
```

主入口函数还必须有一个 float3 返回值。

对于以下浮点数学函数,请使用所描述的语法:

浮动_fabs (浮动)	// 绝对值
浮点_powf (浮点x,浮点y)	// 计算 x 的 y 次方
浮点_logf (浮点)	// 自然对数
浮点_log2f (浮点)	// 以 2 为底的对数
浮点_log10f (浮点)	// 以 10 为底的对数
浮点_exp2f(浮点)	// 以 2 为底的指数
浮点_expf(浮点)	// 以 E 为底的指数
浮点_copysignf (浮点x,浮点y)	// 返回 x,并将符号更改为符号 y
浮点_fmaxf (浮点x,浮点y)	// 如果 x < y,则返回 y

浮点_fminf (浮点x,浮点y)	// 如果 $x > y$,则返回 y
浮点_saturatef (浮点x,浮点minVal,浮点maxVal)	// 返回 $\min(\max(x, \minVal), \maxVal)$
浮点_sqrtf (浮点)	// 平方根
int _ceil(浮点数)	// 朝 + 无穷大舍入为整数
int _floor(浮点)	// 朝 - 无穷大舍入为整数
浮点_fmod (浮点x,浮点y)	// 模数,返回 $x - y$ * 行李箱(x / y)
float _fremainder(float x, float y) // 浮点余数	
int _round(浮点x)	// 最接近 x 舍入的整数值
浮点_hypotf (浮点x,浮点y)	// $(x^2 + y^2)$ 的平方根
浮点_atan2f (浮点x)	// (y / x) 的反正切
浮点_sinf (浮点x)	// 正弦波
浮点_cosf (浮点x)	// 余弦
浮点_acosf (浮点x)	// 反余弦
浮点_asinf (浮点x)	// 反正弦
浮点_fdivide (浮点x,浮点y)	// 返回 (x/y)
浮点_frecip(浮点x)	// 返回 $(1 / x)$

以下函数支持整数类型:

最小值、最大值、绝对值、旋转

其他支持的 C 数学函数包括:

cbirt, cosh, cospi, exp10, expm1, trunc, fdim, fma, lgamma, log1p, logb, rint, 圆形, rsqrt, sincos, sinh, sinpi, tan, tanh, 帮助, tgamma

支持向量类型 float2, float3 和 float4, 数据字段是:

浮动x

浮点y

浮点z

浮动w

要生成向量值,请使用 make_floatN(), 其中 $N = 2, 3$ 或 4 。

用户可以使用 “typedef struct” 定义自己的结构。例如:

类型定义结构

```
{
    浮动 c00, c01, c02;
    浮动 c10, c11, c12;
} 矩阵;
```


要声明常量内存,请使用 `__CONSTANT__`。例如:

```
__CONSTANT__ 浮点 NORM[] = {1.0f / 3.0f, 1.0f / 3.0f, 1.0f / 3.0f};
```

要将常量内存作为函数参数传递,请使用 `__CONSTANTREF__` 限定符,例如:

```
__DEVICE__ float DoSomething(__CONSTANTREF__ float* p_Params)
```

浮点值必须在末尾带有 “f” 字符 (例如 1.2f) 。

一个简单的 DCT LUT 示例

以下代码显示了如何使用 DCT LUT 语法创建简单的颜色增益转换的示例。

```
// 演示简单颜色增益变换的示例
__DEVICE__ float3 变换 (浮点 p_R,浮点 p_G,浮点 p_B)
{
    常量浮点 r = p_R * 1.2f;
    常量浮点 g = p_G * 1.1f;
    常量浮点 b = p_B * 1.2f;
    返回 make_float3(r, g, b);
}
```

矩阵 DCT LUT 示例

以下代码显示了使用 DCT LUT 语法创建矩阵变换的示例。

// 演示如何使用用户定义的矩阵类型将 Rec 中的 RGB 转换为 YUV 的示例。 709

```
__CONSTANT__ 浮点 RGBToYUVMat[9] = { 0.2126f, 0.7152f, 0.0722f,
-0.09991f, -0.33609f, 0.436f,
0.615f, -0.55861f, -0.05639f};
```

```
__DEVICE__ float3 变换 (int p_Width,int p_Height,int p_X,int p_Y,float p_R,float p_G,float p_B)
```

```
{
    float3 结果;

    结果.x = RGBToYUVMat[0] * p_R + RGBToYUVMat[1] * p_G + RGBToYUVMat[2] * p_B;

    结果.y = RGBToYUVMat[3] * p_R + RGBToYUVMat[4] * p_G + RGBToYUVMat[5] * p_B;

    结果.z = RGBToYUVMat[6] * p_R + RGBToYUVMat[7] * p_G + RGBToYUVMat[8] * p_B;

    返回结果;
}
```

更复杂的 DCT LUT 示例

以下代码显示了创建镜像效果的示例,说明了如何在空间上访问像素。

// 镜像效果的空间访问示例

```
__DEVICE__ float3 变换(int p_Width, int p_Height, int p_X, int p_Y, __  
TEXTURE__ p_TexR, __TEXTURE__ p_TexG, __TEXTURE__ p_TexB) {  
  
    const bool isMirror = (p_X < (p_Width / 2));  
    const float r = (isMirror) ? _tex2D(p_TexR, p_X, p_Y) : _tex2D(p_TexR, p_宽度 - 1 - p_X, p_Y);  
  
    const float g = (isMirror) ? _tex2D(p_TexG, p_X, p_Y) : _tex2D(p_TexG, p_宽度 - 1 - p_X, p_Y);  
  
    const float b = (isMirror) ? _tex2D(p_TexB, p_X, p_Y) : _tex2D(p_TexB, p_宽度 - 1 - p_X, p_Y); 返回 make_float3(r, g, b);  
  
}
```

第198章

TCP 协议用于

达芬奇决心

运输控制

本章介绍如何创建将传输控制与 DaVinci Resolve 结合使用的第三方实用程序。

内容

关于 TCP 协议版本 1.2	4118
数据类型	4118
命令格式	4118
响应格式	4118
通讯延迟	4118
状态响应值	4119
TCP协议流	4119
连接	4119
去	4119
玩	4119
获取TC	4119
获取帧率	4119

关于 TCP 协议版本 1.2

该协议定义了第三方应用程序（“客户端”）和DaVinci Resolve（“服务器”）之间使用TCP协议的通信标准。

服务器将使用端口号 9060。此协议中不会使用 SSL。通信采用请求-响应消息的形式，其中客户端发起命令，服务器做出适当响应。

要使用此协议，您必须首先在高级面板中键入以下字符串
达芬奇解决系统偏好设置：

系统.远程.控制 = 1

数据类型

该协议使用以下数据类型：

float (f):4 字节 IEEE 754 单精度浮点

int (i):4 字节有符号 int

unsigned char (uc):1 字节无符号字符 (0-255)

string (s):UTF-8 编码的字符串。没有指定终止符。字符串是复合类型，
作为单个 int (i) 传输，指定字符串中的字符数 (N)，后跟包含字符串字母的 N 个无符号字符 (uc)。

注意:float 和 int 类型的字节以小端顺序传输。

命令格式

命令作为单个字符串传输（仅使用字符 a-z (0x61 - 0x7A)），后跟定义中命令所需的任何附加负载。

响应格式

对任何命令的响应均由状态字节（无符号字符）组成，后跟响应所需的任何附加负载。

通讯延迟

一旦命令字符串的第一个字节被发送，命令字符串的其余部分和有效负载数据必须立即跟随。COMMAND 结束后，服务器必须立即响应。如果在此过程中延迟超过 5 秒，则等待数据的一方可能会认为对方已无响应而断开连接。

目前，两个连续命令之间的延迟没有限制。

注意:或者,可以定义最大允许延迟,在这种情况下,客户端可以定期发出“连接”命令以保持连接处于活动状态。

状态响应值

状态值的含义如下:

0x00:命令执行成功。任何额外的有效负载都会按预期发送。

0xFF:命令无法成功执行。随后不会有额外的有效负载。

TCP协议流

可以通过协议流发送以下命令。

连接

客户端通过发送连接命令字符串来启动流。没有有效负载。服务器以状态值 0x00 进行响应。

去

客户端发送一个 goto 命令字符串,后跟四个无符号字符,分别代表时间码的小时、分钟、秒和帧。

服务器根据命令的执行情况以适当的状态字节进行响应。

玩

客户端发送一个播放命令字符串,后跟一个浮点值。实时播放为1.0,停止为0.0,反转为-1.0,2x为2.0等。

服务器根据命令的执行情况以适当的状态字节进行响应。

获取TC

客户端发送 gettc 命令字符串。

服务器以适当的状态字节进行响应(例如,如果不存在时间线,则状态字节可能为 0xFF)。如果状态字节为 0x00,则后面跟着四个无符号字符,分别代表时间码的小时、分钟、秒和帧。

获取帧率

客户端发送 getframerate 命令字符串。

服务器以适当的状态字节进行响应。如果状态字节为 0x00,则后面跟着帧速率的浮点值。