D3D Player 使用指南

2026 年 6 月

法律事项

本文档中呈现的信息属于格兰菲智能科技股份有限公司和/或其附属公司（以下统称为“格兰菲”）所有，且格兰菲保留不经通知随时对本文档信息或对本文档所述任何产品和服务做出修改的权利。格兰菲不担保信息、文本、图案、链接或本文档内所含其他项目的准确性或完整性。本文档内容仅供参考，格兰菲不对本文档所述产品的可销售性、所有权、不侵犯知识产权、准确性、完整性、稳定性或特定用途适用性做任何暗示担保、保证。格兰菲可不经通知随时对本文档或本文档所述产品或服务做出更改，但不承诺因此更新本文档。

非经格兰菲事先书面许可，任何实体或个人均不得获得本文档的副本，且无权以任何方式处理本文档，包括但不限于使用、复制、修改、合并、出版、发行、销售或传播本文档的部分或全部。在任何情况下，格兰菲均不对因本文档引起的、由本文档造成的、或与之相关的任何索赔、损害或其他责任负责。格兰菲保留自行决定随时更改、修改、添加或删除本文档的部分或全部的权利。

格兰菲对本文档享有最终解释权。

目录

修订记录

版本	日期	作者	描述
A0	2026/6/10	格兰菲	初次发布

1 概述

在用户使用 D3D Logger 工具，获得一个精确记录了 D3D 应用程序与图形驱动之间所有交互的日志后（本指南称之为脚本）。Player 系列工具完全脱离原始应用程序，通过对捕获的脚本进行分析，实现在多种目标平台上，如实地复现相同的 API 调用序列与渲染结果（本指南称之为回放）。

Player 系列工具由 5 个以“Player”开头命名的 exe 组成，分别用于回放不同类型的 D3D API 的脚本，见表 1。

Player 系列工具	D3D API 支持
Player.exe	D3D9
Player10.exe	D3D10 (经典语法)
Player11.exe	D3D11(经典语法)
Player11_1.exe	D3D11.1(经典语法)
Player12.exe	D3D10(现代语法) D3D11(现代语法) D3D12

表 1

1.1 Player 系列工具窗口布局

Player 系列工具的窗口布局分为三个区域，如图 1：

（1） 绿色的菜单栏区，提供与用户交互的功能入口。

（2） 黄色的显示区，用于显示渲染结果。

（3） 红色的状态栏区，从左至右分为 5 个信息格：

- 当前状态："Initializing...", "Idle", "Rendering..."
- 当前帧号
- 渲染结果："Passed..." "Failed"，正确率，以及颜色通道允许的阈值偏差

- 当前运行的脚本行号
- 脚本名称

图 1

1.2 快速开始

本章节以 Player12.exe 回放 D3D12 脚本为示例，简要介绍 Player 系列工具回放脚本的通用方法，后续章节再分别说明各 Player 工具特有的功能。

（1）使用【File】→【Open】找到脚本路径，选择一帧要回放的脚本，如图 2。

（2）使用【Run】→【Device】配置实现脚本回放的平台。这里第一个设备【Device1】选择 hardware，代表当前的硬件环境。第二个设备【Device2】选择为 Image，并在【Image】区域指定一个 dds 文件，它将作为验证【Device1】渲染结果正确性的基准（本指南称之为 GoldenImage），如图 3。

（3）点击【Run】→【Frame Step】或快捷键 F8，Player12 开始运行脚本记录的 API 序列。至脚本解析结束时，【Device1】显示该帧渲染结果，【Device2】显示 Golden Image 图像。在状态栏中显示两个运行信息及结果信息，如图 1。当两个 Device 的对比正确率达不到 100%时，长按鼠标左键可以高亮显示对比失败的像素点，以便于用户观察。

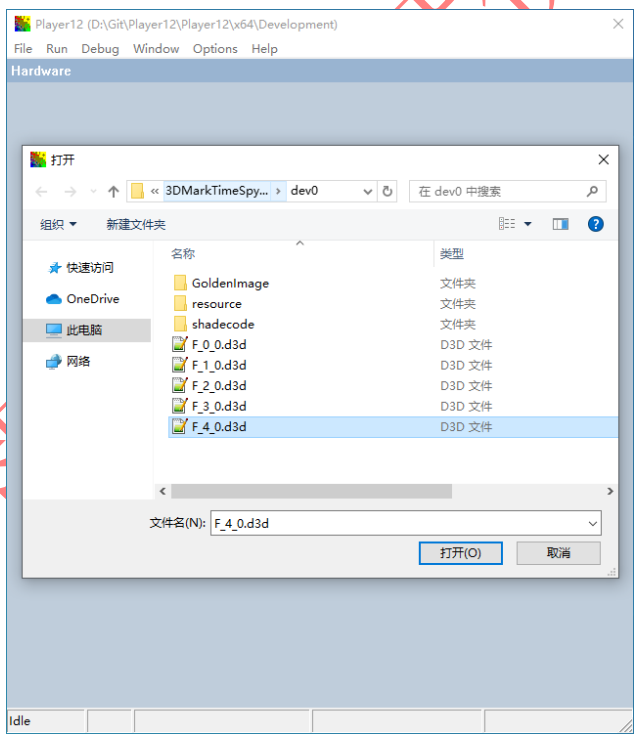

图 2

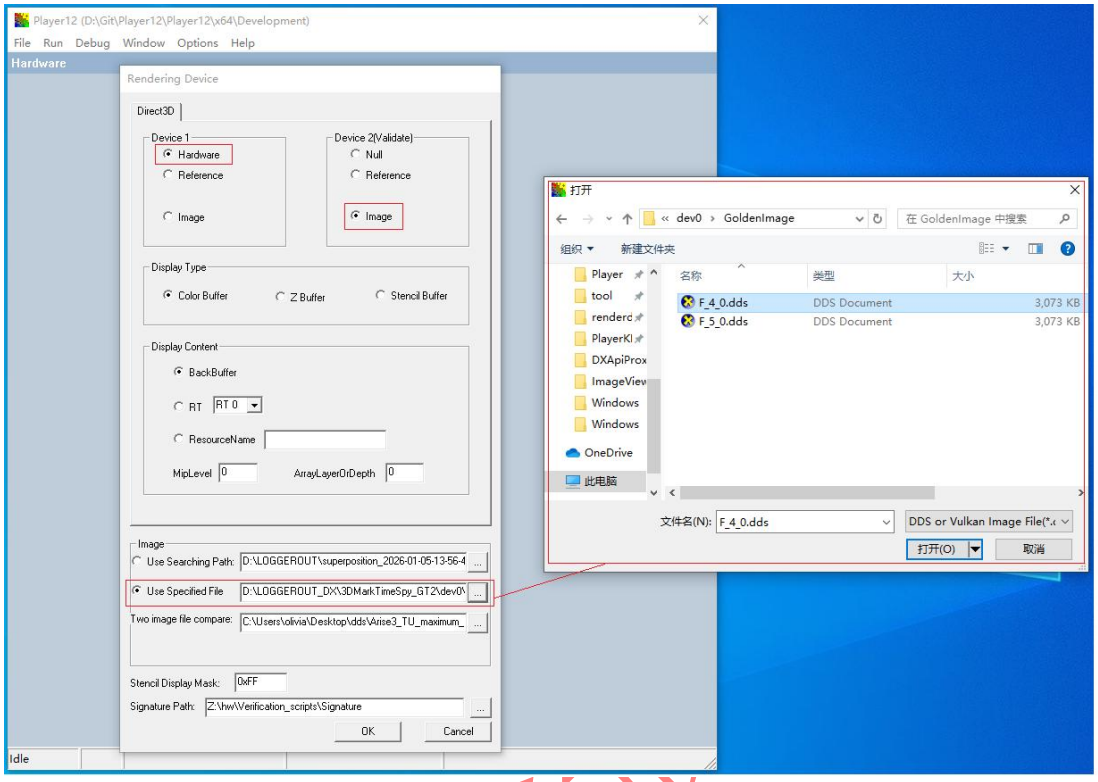

图 3

2 Player9、Player10、Player11、Player11_1

Player 系列工具中 Player.exe、Player10.exe、Player11.exe、Player11_1.exe 的 UI 界面是完全一致的，但是请注意，选定某个版本的 D3D 脚本后，依然要使用对应的 Player 版本。Player.exe 用于回放 D3D9 脚本，Player10.exe、Player11.exe、Player11_1.exe 分别用于回放 D3D10、D3D11、D3D11.1 的经典语法脚本。

2.1 UI 功能介绍

2.1.1 【菜单栏 File】

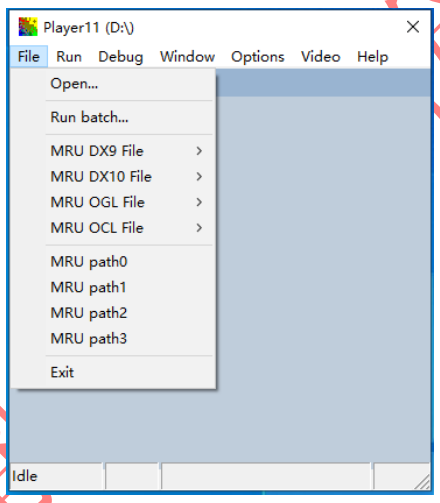

图 4

【File】→【Open】

打开待回放脚本。

选择一个独立的单帧脚本（称之为 1keylog 脚本），或全记录脚本的第一帧（称之为 fulllog 脚本），两种脚本的获得可以参考《D3D logger 使用指南》。当选择回放 fulllog 脚本时，通过脚本命名和【Options】→【Misc】→“Auto pickup next file only if it starts with F_”来控制 Player 是否自动回放下一帧。

【File】→【Run batch…】

打开一个 batch 文件，并且按顺序回放 batch 中的脚本。batch 文件中至少包含一个脚本。如果需要回放一批脚本，可以创捷一个 batch 文件，列出带完整路径的脚本名称。

【File】→【MRU DX9/DX10 File】

显示最近运行的 N 个脚本文件。用户可以直接选择一个脚本文件开始回放。

【File】→【MRU Path】

显示最近常用的访问路径。用户可以在【Options】→【General】→“MRU Script Path”中指定常用路径。

2.1.2 【菜单栏 Run】

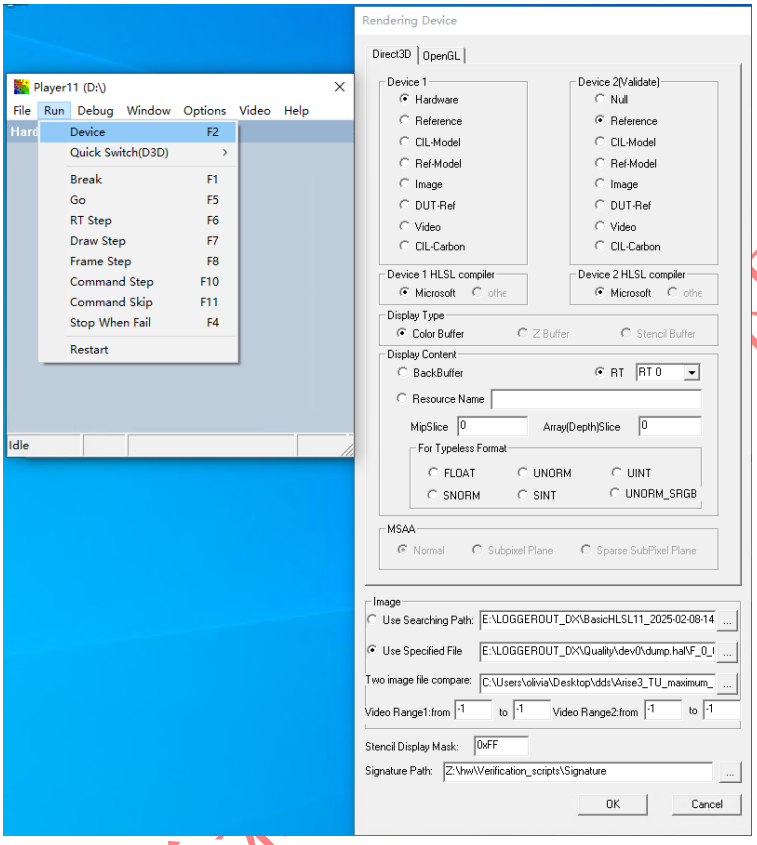

图 5

【Run】→【Device】

允许同时配置两个渲染设备来回放同一个脚本，并且分别绘制场景，同时在窗口中显示渲染结果。【Rendering Device】对话框如图 5。

- **Device1/Device2**

Device1 是主用设备，Device2 是备用设备。当两个 device 有发生改变时，重视当前的回放，Player 重置为 Idle 模式。 Device2 用来验证，如果不是必须运行，可设置为 NULL。

- **Device Type**

Hardware：

在硬件设备上创建 D3D Device, API 的实现会经过 Player，D3D Runtime, Driver。Player 运行机制等同与一个 D3D 应用程序。

Reference：

在 reference 上创建 D3D Device, API 的实现会经过 Player，D3D Runtime, Reference DLL（例如 d3d10ref.dll、d3d11ref.dll）。Player 运行机制等同与一个 D3D 应用程序。

Image:

加载一个指定的 Image dds 文件，用于验证渲染结果。

CIL-Model（内部开发使用）：

在用户模式下模拟所有必须的组件，Player 的运行机制等同于常规应用程序。使用 PIF_CP 配置 Playerint.dll，Driver 和 CModel。

Ref-Model（内部开发使用）：

该设备是修改过的 reference dll，在正式 Driver 准备就绪前，用于验证目标芯片的部分 feature。运行需要以来特定的 D3D reference 环境。

- **Display Type**

Player 可以在窗口显示的纹理类型：Color Buffer、Z Buffer 或 Stencil Buffer。用户可通过切换 display tpye 来 debug 脚本。正常地，渲染设备只能显示 Color bufer；对于内部开发环境 CModel 或修改版的 reference，也可以显示 Z Buffer 和 Stencil Buffer。

- **Display content**

Player 可以显示应用程序绘制的所有 render target。对于基于纹理/多渲染目标（MRT）的渲染场景中，单帧可能包含多个 RT，且每个 RT 包含多个 Subresource，通过切换 RT0-RT7 并设置各 RT 的 MipSlice 和 ArraySlice,可以便捷显示可渲染的纹理和多渲染目标（MRT）。更改 Display content 后，无需重启 player 即可实时查看渲染结果。

- **HLSL Compiler**

为脚本中的 HLSL 着色器选择编译器。

- **MSAA**

当要显示的 surface 是 msaa texture 时：Normal 模式显示 downSampled surface； Subpixel Plane 和 Sparsh subpixel Plane 模式需要使用 Ref-model 或 CModel 的内部开发 rendering device 环境。

- **Image**

指定 Rendering Device 要加载的 Image 或 Image 的路径。当选择“Use Searching Path”时，Player 自动加载路径中脚本的同名 dds 文件。

【Run】→【Break】

暂停正在运行的脚本。

【Run】→【Go】

Idle 状态时，开始运行，持续运行直到遇到 breakpoint 或者没有脚本可以运行，或者出现错误。当遇到 breakpoint 时，会从【Go】模式切换到【Command Step】模式。当用户使用【Frame Step】、【Command Step】或【Command Skip】模式时，player 会在下一帧切回到这些模式。当脚本运行结束，Player 会停止，变回 Idel 状态。

【Run】→【RT Step】

Idle 状态时，开始运行，持续运行直到遇到 breakpoint 或者 Present 命令，或者出现错误。当遇到 breakpoint 时，会从【Go】模式切换到【Command Step】模式。当用户使用【Frame Step】、【Command Step】或【Command Skip】模式时，player 会在下一帧切回到这些模式。当脚本运行结束，Player 会停止，变回 Idel 状态。

【Run】→【Draw Step】

Idle 状态时，开始运行，持续运行直到遇到 breakpoint 或者渲染命令，或者出现错误。当遇到 breakpoint 时，会从【Go】模式切换到【Command Step】模式。当用户使用【Frame Step】、【Command Step】或【Command Skip】模式时，player 会在下一帧切回到这些模式。当脚本运行结束，Player 会停止，变回 Idel 状态。

【Run】→【Frame Step】

Idle 状态时，开始运行。执行一帧后停下来。遇到 Breakpoint 会使 Player 切换到【Command Step】模式。当脚本运行结束，Player 会停止，变回 Idel 状态。

【Run】→【Command Step】

Idle 状态时，开始运行。打开 Command 窗口并且单步执行脚本。当脚本运行结束，Player 会停止，变回 Idel 状态。Rendering Device 窗口会显示当前的 BackBuffer。可以在 Command 窗口中选中一行命令，用 F9 设置 BreakPoint，并且该 BreakPoint 会保留在脚本的数据

包中持续生效。

【Run】→【Command Skip】

Idle 状态时，开始运行。打开 Command 窗口并且跳过一个 command 后停下来，跳过的命令以灰色标注。

【Run】→【Stop when Fail】

每执行绘制命令之后对比 RT，当对比 Fail 时停下来。

2.1.3 【菜单栏 Debug】

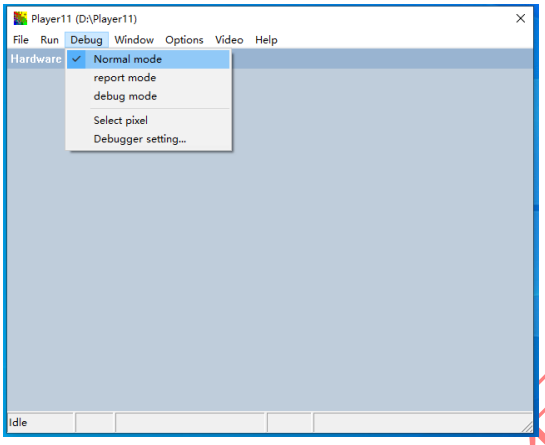

图 6

本菜单栏的【report mode】和用于内部 CModel 开发平台的设置， 此处简要介绍。

【Debug】→【Normal mode】

标准回放模式，执行脚本 API 命令直至结束，并在状态栏显示对比结果。

【Debug】→【report mode】

当对比结果是 Failed 时，可以选择 Report 模式来获得更多的信息。首先，选择一个像素点。打开【菜单栏 Debug】→【Select pixel】窗口，鼠标右键选中一个像素点，或者使用【菜单栏 Debug】→【Debugger Setting…】窗口直接输入像素点坐标。然后，选中【report mode】回放脚本，在第一次出现 Fail 时，会停下来，Dump 该像素相关的 texture 和信息，并标记无关的 Draw。

【Debug】→【debug mode】

选中小窗口中“Skip useless renderings in debug mode”，选中【debug mode】回放脚本，对于结果正确的渲染操作，自动 Load RT Surface 和 DS Surface，然后回放过程会 break 在相关渲染操作的 Failed 的 Primitive 上。

2.1.4【菜单栏 Window】

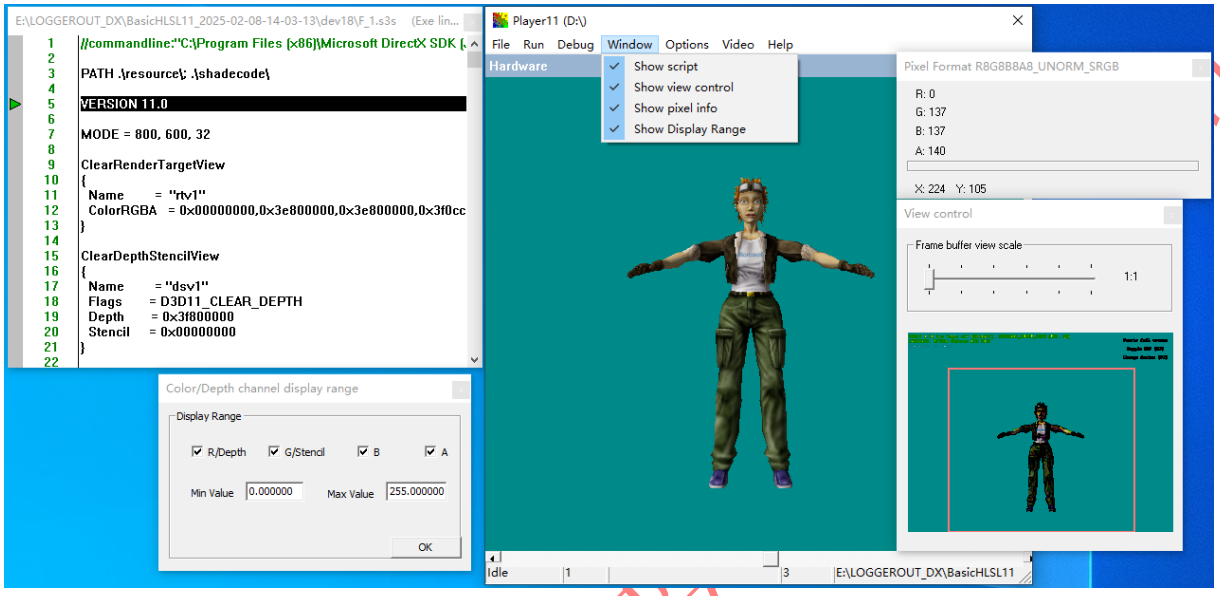

图 7

【Window】→【Show Script】

启动窗口显示脚本。

【Window】→【Show View Control】

启动视图缩放窗口。可以放大渲染场景，也可以缩小还原，放大功能对两个 Rendering Device 均生效。放大后，渲染窗口会出现横向和纵向滚动条。你可以滚动窗口查看渲染结果的全部内容。在视图控制窗口中，有一幅 Device1 渲染结果的全景缩略图，上面有一个白色方框，标示当前渲染窗口所显示的区域范围。你可以在白色方框中间点击左键并拖动，将视图移动到任意位置。渲染窗口会随之更新。

【Window】→【Show Pixel Info】

启动像素信息窗口。像素信息窗口显示当前鼠标所指像素（在渲染区域内）的 RGB/Z/Stencil 值及位置信息。

【Window】→【Show Display Range】

启动 Color Channel 显示范围窗口。可以指定要显示的 Color Channel，并且对选定的 Color Channel 规定一个显示的范围值，过滤掉在指定范围之外的像素点，只显示在指定范围内的像素点。

2.1.5 【菜单栏 Options】

2.1.5.1 【Options】→【General 页】

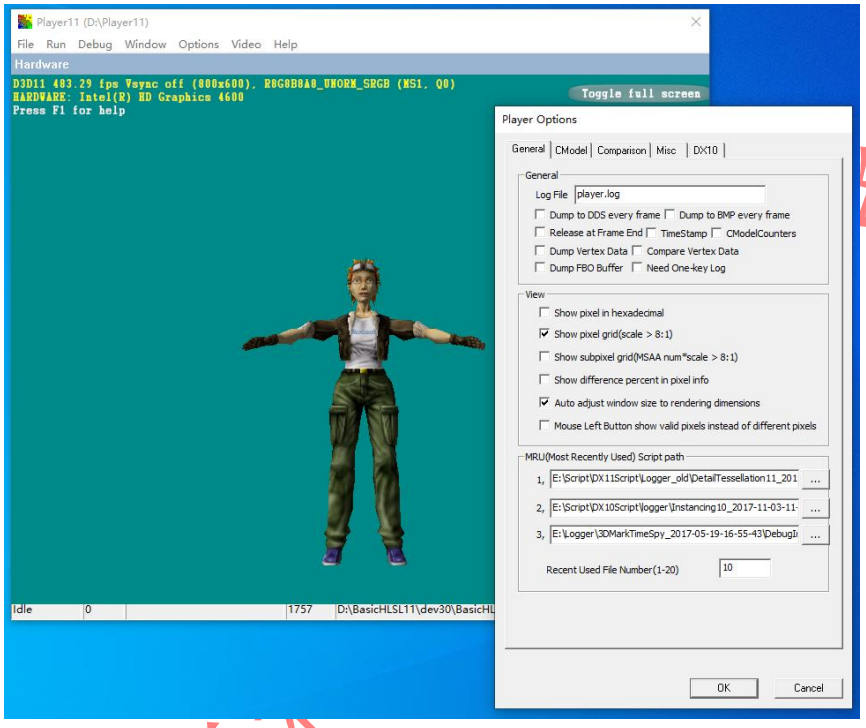

图 8

【General 区域】

Log File:

指定日志文件命名。

Dump to DDS every frame:

每帧的结尾 dump 当前 RT 绘制结果保存至 DDS 文件。存放于执行脚本目录下的 dump.ref（REF Device）、dump.mdl（CModel Device）或 dump.hal（Hardware Device）子目录中。Dump 文件以脚本名命名，为 DDS 格式，可使用 DirectX 纹理工具打开。

Dump to BMP every frame:

每帧的结尾 dump 当前 RT 绘制结果保存至 DDS 文件。存放于执行脚本目录下的 dump.ref（REF Device）、dump.mdl（CModel Device）或 dump.hal（Hardware Device）子目录中。Dump 文件以脚本名命名，为 BMP 格式。

Release at Frame End:

每帧播放结束释放 Player 中创建的所有对象。此功能仅适用于单个 Rendering Device，因为释放后不允许进行比较（Player 会对两个 Rendering Device 进行比对）。

TimeStamp:

在 Draw 命令前后插入时间戳。

CmodelCounters/ Dump Vertex Data/ Compare Vertex Data: 作用于内部 CModel 开发环境。

【View 区域】

Show pixel in hexadecimal: 16 进制格式显示像素信息。

Show pixel grid: 当纹理图像放大超过 8 倍时显示像素网格。

Show sub pixel grid: 当 MSAA 图像采样个数*缩放比例>8 时，显示像素网格。

Show difference percent in pixel info: 当开启两个设备时，显示像素点的差异百分比。

Auto adjust window size to rendering dimensions: 自动调整窗口大小到渲染的维度大小。

Mouse Left button show valid pixel instead of different pixels: 当开启两个设备时，长按鼠标显示匹配的像素点（默认长按鼠标显示不匹配的像素点，以便发现错误的像素点）。

【MRU Script path 区域】

设置 3 个固定常用的脚本路径。

Recent Used File Number:

最近使用文件的显示个数。

2.1.5.2 【Options】→【CModel 页】

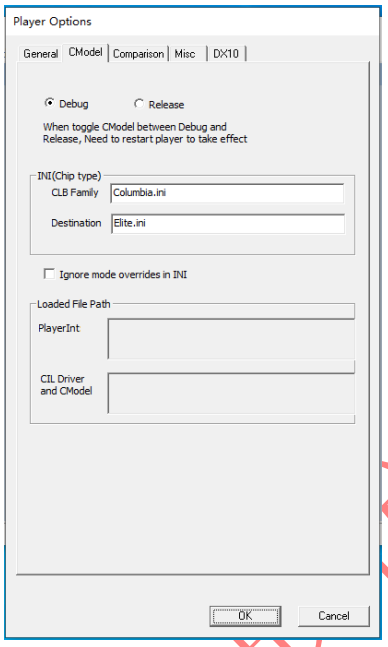

图 9

本 Tab 页用于内部 CModel 开发平台的相关设置。

Debug/Release：

选择 CModel 平台是 Debug 或 Release。

CLB Family/Destination:

针对 chiptype 设置 Model 平台的配置文件。

Ignore mode overrides in INI:

忽略多个 INI 时，包含的各项的重复设置覆盖。

Loaded File Path:

加载 Player ini 文件和 CIL Driver&CModel 的路径。

2.1.5.3 【Options】→【Comparison 页】

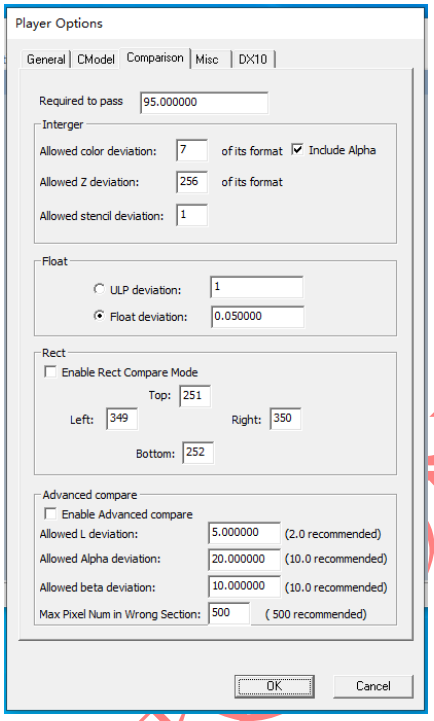

图 10

Required to pass:

在比较模式下，Player 会在状态栏中显示两个 Rendering Device 渲染结果的相似度百分比。若该百分比高于当前设置，播放器将显示"Pass(xx%)"，否则显示"Failed(xx%)"。

【Interger 区域】

Allowed Color Deviation:

每个 Color Channel 允许的颜色偏差 x（以其格式为单位）：比较两个 Device 渲染结果的 Color Buffer 时，若同一像素的 R（或 G、B）值差值低于此设置，则该通道的相似度百分比视为 100%。

Allowed Z Deviation:

允许 Z 偏差 x（以其格式为单位）：比较两个 Device 渲染结果的深度缓冲区时，若同一像素的 Z 值差值低于此设置，则该像素的相似度百分比视为 100%。

Allowed Stencil Deviation:

比较两个 Device 渲染结果的模板缓冲区时，若同一像素的模板值差值低于此设置，则该像素的相似度百分比视为 100%。

Include Alpha：比较两个 Device 渲染结果时，若勾选此项，则比较将包含 Alpha 通道。

【Float 区域】

Float Deviation:

比较两个 float 格式缓冲区时，若同一像素的浮点值差值低于此设置，则该像素的相似度百分比视为 100%。

ULP Deviation：

比较两个 float 格式缓冲区时，若二进制位的差值低于此设置，则该像素的相似度百分比视为 100%。

【Rect 区域】

若选择启用 Rect 比较模式，则左窗口中将显示一个矩形，比较结果仅计算矩形内的像素。左、下、右、上四个参数定义了矩形的大小和位置。用户也可以使用鼠标右键指定该矩形区域。

【Advanced compare 区域】

此区域针对 D3D9 的 Player.exe 生效。若勾选"Enable Advanced Compare"，Player.exe 会使用下方设置的参数实现 Compare。

Allowed L/Alpha/beat deviation:

该通道允许的偏差阈值。

Max Pixel Num in Wrong Section:

区域内允许的错误像素点个数。

2.1.5.4 【Options】→【Misc 页】

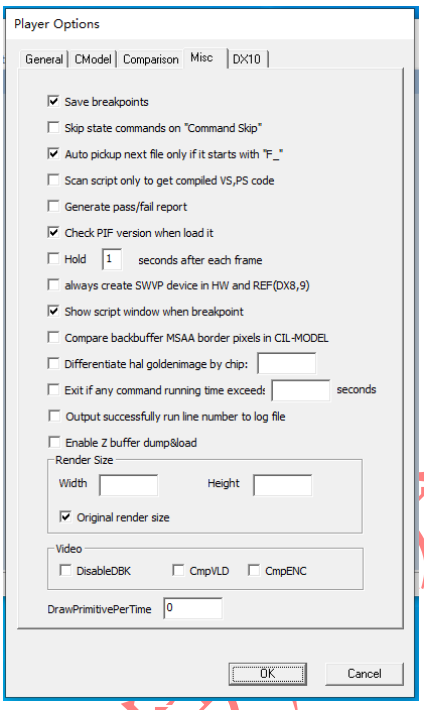

图 11

Save breakpoints:

保存设置的脚本断点。

Skip state commands on "Command Skip":

在使用【Command Skip】启动运行时，跳过状态设置类命令。

Auto pickup next file only if it starts with "F_":

对于 fulllog 脚本，选中此项时，仅自动回放以“F_”开头命名的下一帧脚本。若此项未勾选，Player 会自动回放“命名相同且以递增数字结尾的”下一帧脚本。

Scan script only to get compiled VS,PS code:

“浏览式”执行脚本，得到编译后的着色器。

Generate pass/fail report:

将比较结果写入 report.csv 文件。

Check PIF version when load it:

检查 PIF 工具（内部开发使用工具）版本。

Hold x seconds after each frame:

连续多帧播放时，在每帧结束后暂停 n 秒，并在状态栏中显示倒计时。

Show script window when breakpoint:

遇到断点显示脚本窗口。

Compare backbuffer MSAA border pixels in CIL-MODEL:

在 CIL-MODEL 平台对比 back buffer MSAA 边界像素。

Differentiate hal goldenimage by chip x:

使用指定 Chip 的 goldenimage。

Exit if and command running exceed x seconds:

每个 API 指令允许执行的最长时间，若执行时间超过 n 秒，Player 退出。

Output successfully run line number to log file:

输出执行成功的脚本行号。

Enable Z buffer dump&load:

开启 Z buffer 的 dump&load。

Render Size:

渲染区域。

2.1.5.5 【Options】→【DX10 页】

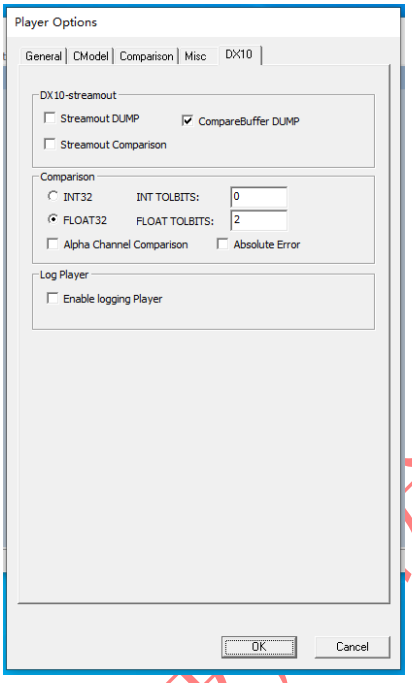

图 12

此 tab 页针对 Player10/11/11_1.exe 生效。

【DX10 streamout 区域】

Streamout DUMP:

dump streamout buffer 到.so 文件。

ComapareBuffer:

当设置了两个 Rendering Device 时，在遇到脚本命令"CompareBuffer"时，将 buffer 转储为 .txt 文件。

Streamout Comparison:

当设置了两个 Rendering Device 时，比较 Device1 和 Device2 之间所有 streamout buffer（设置为 SO 阶段）。

【Comparison 区域】

INT32 & INT TOLBITS:

比较缓冲区时将数据设为整型格式，并设置整型精度，默认值为 0。

FLOAT32 & FLOAT TOLBITS:

比较缓冲区时将数据设为浮点格式，并设置浮点精度，默认值为 2。

Absolute Error:

比较两个浮点类型资源时使用绝对差值。

Alpha Channel Comparison:

比较两个渲染结果时，若包含 Alpha 通道，则一并比较其 Alpha 通道。

【Log Player 区域】

允许使用 D3D Logger 捕获 Player 的渲染行为。

2.1.6 命令行参数

运行 Player 系列工具进行批处理脚本测试时，使用命令行可以更方便。使用“Player/10/11/11_1.exe -help”可以查看所有的命令行参数。

图 13

2.2 脚本命令实现具体任务

对于 Player10.exe、Player11.exe 和 Player11_1.exe, D3D10、D3D11、D3D11.1 的经典语法脚本中除了标准 API 渲染指令外，还支持插入自定义脚本命令，来辅助调试完成具体任务。有下面 3 类自定义的脚本命令：

Compare*命令
命令语法： CompareResource “ResourceName”, MinMipLevels, MaxMipLevels, MinArrayIndex, MaxArrayIndex **功能及详情：** 对比一个 Resource 在两个 Rendering Device 中的渲染结果。 MinMipLevels、MaxMipLevels 需在 [0, MipLevels-1]之间。 MinArrayIndex、MaxArrayIndex 需在 [0, ArraySize-1]之间。
命令语法： （1）CompareTexture “TextureName” （2）CompareTexture “TextureName”,Subresourceid （3）CompareTexture “TextureName1”,Subresourceid1, “TextureName2”,Subresourceid2, **功能及详情：** 命令（1）对比一个 Texture 在两个 Rendering Device 中的渲染结果，对比所有的 subresouce。 命令（2）对比一个 Texture 在两个 Rendering Device 中的渲染结果，对比指定的 subreouce。 命令（3）在一个 Rendering Device 内部对比两个 Texture 的各自指定的 subresource。需要确保两个 texture 在当前的 subresourceid 上长宽相等。如果 subresourceid1 和 subresourceid2 设为 -1,对比整个 textture。
命令语法： CompareTexture2D “textureName” **功能及详情：** 对比一个 Texture 在两个 Rendering Device 中的渲染结果。会把 texture format 转换到 DXGI_FORMAT_R8G8B8A8_UNORM 再做对比。

命令语法：

CompareTextureWithFile “TextureName”,”FileName.DDS”

功能及详情：

将 Texture1D/2D/3D 与指定的 dds 文件对比，对比所有的 subresouce。dds 文件中的 texture 需要和该 texture 的大小一致。

命令语法：

（1）CompareBuffer “BufferName”

（2）CompareBuffer “BufferName1”,offset1, “BufferName2”,offset2

（3）CompareBuffer “BufferName1”,offset1, “BufferName2”,offset2,bytelength

功能及详情：

可以用来对比 VertexBuffer， IndexBuffer 等。

命令（1）对比一个 Buffer 在两个 Rendering Device 中的数据。

命令（2）在一个 Rendering Device 内部对比两个 Buffer 在指定 offset 开始的所有数据。

命令（3）在一个 Rendering Device 内部对比两个 Buffer 在指定 offset 开始的 bytelenght 数据。

命令语法：

(1) CompareRTBufferwithRTFormat “BufferName”

(2) CompareRTBufferwithRTFormat “BufferName1”,offset1, “BufferName2”,offset2

(3) CompareRTBufferwithRTFormat “BufferName1”,offset1, “BufferName2”,offset2,bytelength

功能及详情：

可用来对比作为 RTV 的 buffer（以 RTV 的 format 来对比）。

命令（1）对比 RTV Buffer 在两个 Rendering Device 中的数据。

命令（2）在一个 Rendering Device 内部对比两个 RTV Buffer 在指定 offset 开始的所有数据。

命令（3）在一个 Rendering Device 内部对比两个 RTV Buffer 在指定 offset 开始的 bytelenght 数据。

命令语法：

CompareBufferWithFile “BufferName” ,offset1, FileName ,offset2, bytelength , format

功能及详情：

将 Buffer 和文件数据对比。如果 bytelength 设为 0，对比所有数据。Format 参数可设为一个 DXGI_FORMANT 或不设。

命令语法：

CompareFile "FileName1","bin",offset1, "FileName2","bin",offset2,bytelength

功能及详情：

直接对比两个 bin 文件。

命令语法：

CompareBufferInBytes "BufferName", ByteOffset,ByteLength

功能及详情：

对比一个 Buffer 在两个 Rendering Device 的数据。

命令语法：

CompareTextureInBox "TextureName", subResource,xOffset,xLength,yOffset,yLength,zOffset,zLength

功能及详情：

对比一个 Texture 的指定区域在两个 Rendering Device 的数据。

命令语法：

COMPARE_EVERY_RENDER_TARGET ON/OFF

功能及详情：

切换 RT 前，对比当前 RT 在两个 Rendering Device 的渲染结果。

命令语法：

COMPARE_EVERY_RENDER_TARGET_AFTER_EVERY_DRAW ON/OFF

功能及详情：

每个 Draw 命令执行完成后，对比当前 RT 在两个 Rendering Device 的渲染结果。

命令语法：

COMPARE_MRT_GOLDEN ON/OFF

功能及详情： 将 MRT 和 GoldenImage 文件做对比。

表 2

Dump*命令
命令语法： DUMP “TextureName”,filename.DSS DUMP “BufferName”,filename.bin **功能及详情：** Dump 指定 resource 到指定文件。
命令语法： DUMPTEXT “Name”, filename **功能及详情：** Dump 指定 resource 到指定 text 文件。Filename 不包含扩展名，得到 text 文件“filename_Sub#.txt”， #是 resource 的 subresourceid。
命令语法： DUMP2D_BIN “Name”,filename **功能及详情：** Dump 指定 texture2D 到指定 bin 文件。
命令语法： DUMPCUBEMAP_BIN “Name”,filename **功能及详情：** Dump 指定 textureCube 到指定 bin 文件。
命令语法：

DUMP_BMP "filename.bmp","Texture2DName" **功能及详情：** Dump 指定 texture2D 到指定 bmp 文件。
命令语法： DUMP_SPECTOGRAM_BMP "filename.bmp","Texture2DName" **功能及详情：** Dump 指定 texture2D 到指定 bmp 文件。
命令语法： DUMP_EVERY_RENDER_TARGET ON/OFF **功能及详情：** 切换 RT 前，dump 当前 RT 到 DDS 文件。
命令语法： DUMP_EVERY_RENDER_TARGET_AFTER_EVERY_DRAW ON/OFF **功能及详情：** 每个 Draw 命令执行完成后，dump 当前 RT 到 DDS 文件。

表 3

暂停、恢复命令
命令语法： BREAKPOINT **功能及详情：** Player 暂停。
命令语法：

SUSPEND_RENDERING **功能及详情：** 停止 render 类命令，比如 draw, drawIndexedInstanced, dispatch。
命令语法： **SUSPEND_RENDERING** **功能及详情：** 恢复 render 类命令，比如 draw, drawIndexedInstanced, dispatch。

表 4

3 Player12

Player12.exe 用于回放 D3D10、D3D11 的现代语法脚本，以及 D3D12 脚本。

3.1 UI 功能介绍

Player12.exe 的 UI 窗口布局以及各功能入口与 2.1 章节介绍的 Player 系列工具高度一致，同时也具有 Player12.exe 的一些特有功能入口。

3.1.1 【菜单栏 File】

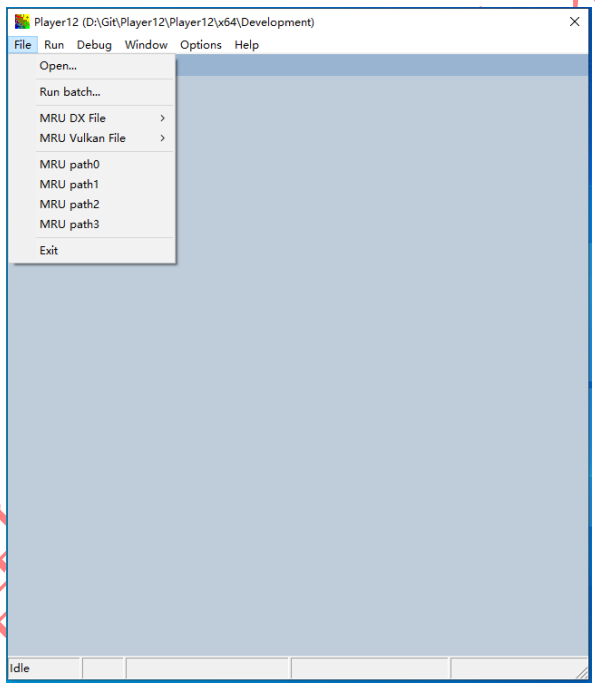

图 14

【File】→【Open】

打开脚本文件。以“.d3d”为后缀的脚本。

【File】→【Run batch…】

保留功能，暂未实现。

【File】→【MRU DX File】

显示最近使用的 DX 脚本列表。

【File】→【MRU Vulkan File】

Vulkan 相关功能，此处不涉及。

【File】→【MRU path0/1/2/3】

这是最近使用脚本路径(MRU Path)， 可在 Options 菜单中的 General→MRU(Most Recently Used)Script Path 中进行配置，该功能有助于快速打开目标路径。当点击【File】→【Open】时，用户能从预先设定好的 MRU Path 中快速定位并打开目标脚本文件。

【File】→【Exit】

退出 Player。

3.1.2 【菜单栏 Run】

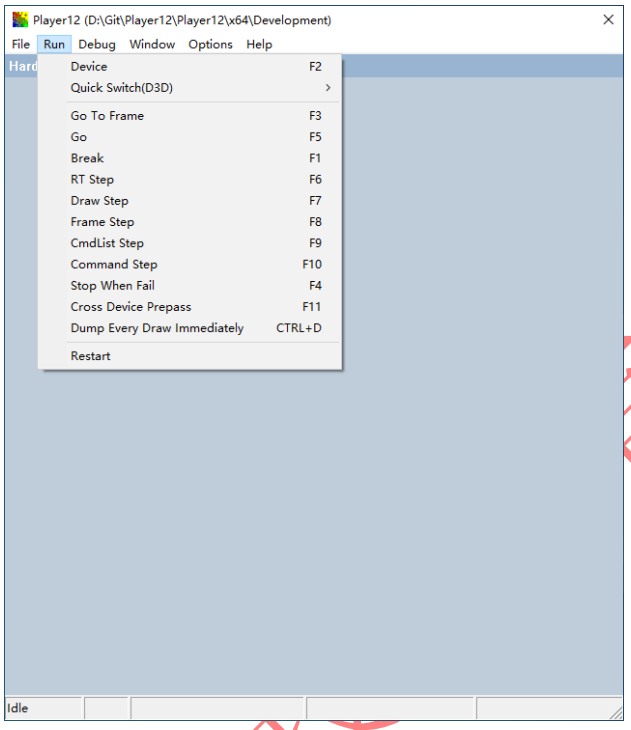

图 15

【Run】→【Device】

允许同时配置两个渲染设备来回放同一个脚本，并且分别绘制场景，同时在窗口中显示渲染结果。【Rendering Device】对话框如图 16。

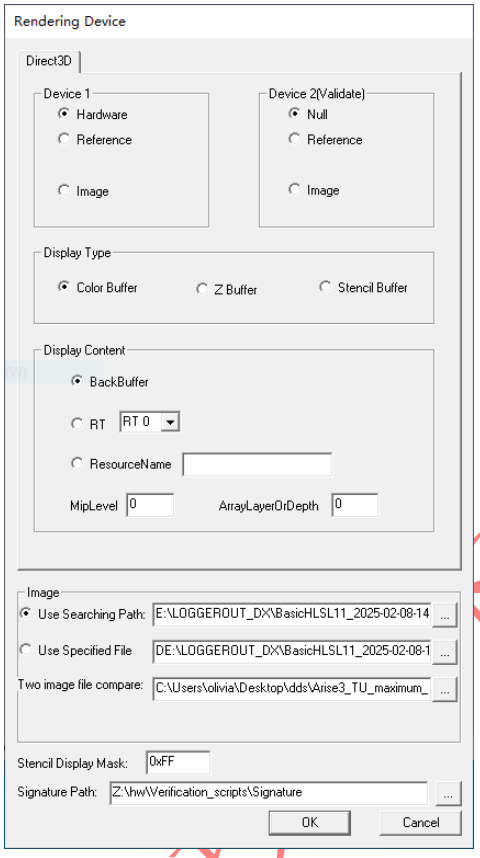

图 16

- **Device 1/2(Validate)**

对于 Device 1，可以选择 Hardware 或 Image 或 Reference。对于 Device 2，可以选择 NULL 或 Image 或 Reference（注：Reference 仅适用于 DX）。如果 Device2 不为空，Player12.exe 将比较 Device1 与 Device2 的渲染结果。

- **Display Type**

Color Buffer：

显示当前渲染目标或后备缓冲的颜色缓冲区内容。

Z Buffer：

显示当前渲染目标的深度缓冲区内容。

Stencil Buffer：

显示当前渲染目标的模板缓冲区内容。

- **Display Content**

BackBuffer：

显示交换链缓冲区。

RT：

显示当前渲染目标；RT0-RT7：在多重渲染目标之间切换。

ResourceName：

显示指定的资源。

MipLevel：

显示资源的指定 Mip 层级。

ArrayLayerOrDepth：

显示纹理数组或 3D 纹理的指定层。

- **Image**

Use Searching Path

若选中此选项，且 Device2 设置为 Image 模式，则将在指定路径中搜索并加载图像。图像文件名应与当前帧名称相同，且对于 DirectX，其文件后缀应为“.dds”。

Use Specified File

若选中此选项，当 Device1 不为 Image 并且 Device2 为 Image 时，指定 Device2 的图像文件名，当 Device1 为 Image，指定 Device1 的图像文件名。

Two image file compare

当 Device1 和 Device2 均为 Image 模式时启用，指定 Device2 的图像文件名。

- **Stencil Display Mask**

保留功能，暂不支持。

- **Signature Path**

保留功能，暂不支持。

【Run】→【Quick Switch(D3D)】

在 RT0 至 RT7 与后备缓冲区之间切换。

【Run】→【Go To Frame (F3)】

Player 运行至指定帧数。

【Run】→【Go (F5)】

开始或继续运行，直至遇到错误、断点或完成所有帧。

【Run】→【Break (F1)】

脚本运行期间，按 F1 键暂停。（按菜单项无效）

【Run】→【RT Step (F6)】

运行到下一个 OMSetRenderTargets。

【Run】→【Draw Step (F7)】

运行到下一个 Draw 命令。

【Run】→【Frame Step (F8)】

开始或继续运行，直至遇到错误、断点或完成一帧。

【Run】→【 CmdList Step (F9)】

运行到下一个 D3D12 的 ExecuteCommandLists。

【Run】→【Command Step (F10)】

开始或继续执行一条指令。

【Run】→【Command Skip (F11)】

【Run】→【Stop When Fail (F4)】

保留功能，暂不支持。

【Run】→【Restart】

终止当前脚本。

3.1.1 【菜单栏 Debug】

保留功能入口。

3.1.2 【菜单栏 Window】

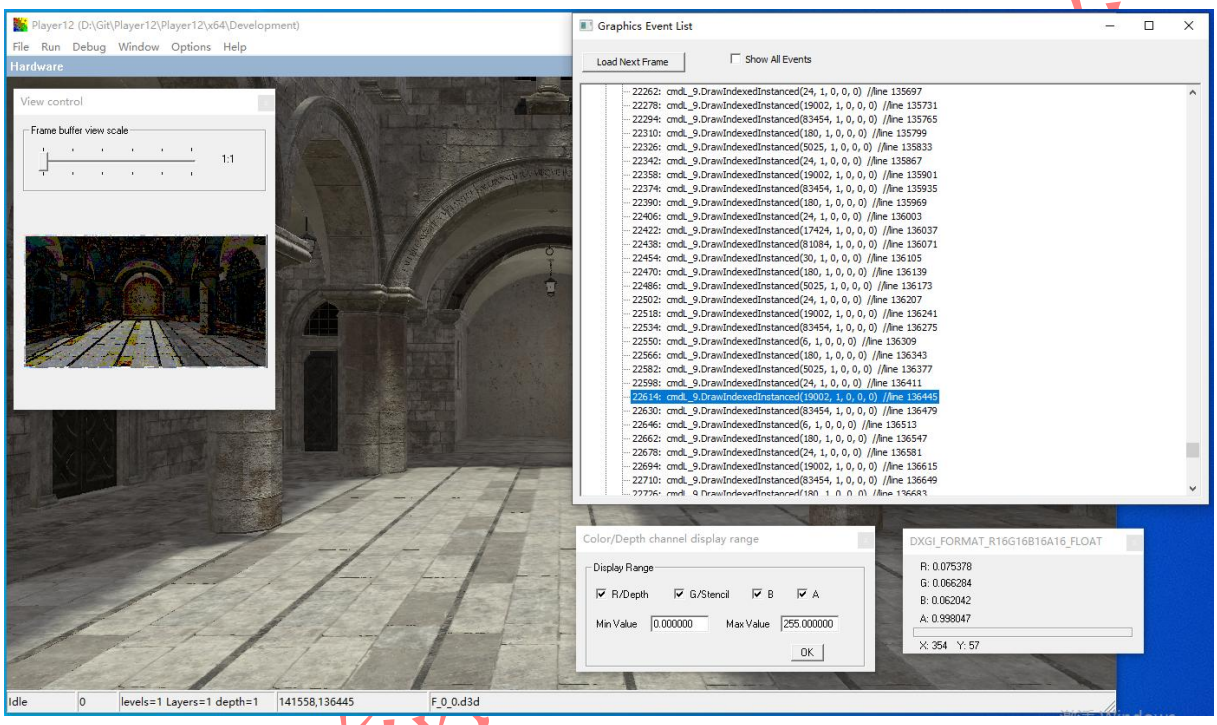

图 17

【Window】→【Show script】

保留功能入口。

【Window】→【Show view control】

启动视图缩放窗口。可以放大渲染场景，也可以缩小还原，放大功能对两个 Rendering Device 均生效。放大后，渲染窗口会出现横向和纵向滚动条。你可以滚动窗口查看渲染结果的全部内容。在视图控制窗口中，有一幅 Device1 渲染结果的全景缩略图，上面有一个白色方框，标示当前渲染窗口所显示的区域范围。你可以在白色方框中间点击左键并拖动，将视图移动到任意位置。渲染窗口会随之更新。

【Window】→【Show pixel info】

显示像素信息窗口，该窗口显示渲染区域内当前鼠标指针所指向像素的格式、RGBA 值以及位置坐标。若像素格式为深度模板（Depth-Stencil）格式，则 R 通道显示深度（Depth）值，G 通道显示模板（Stencil）值。若启动两个 Rendering Device，则左侧 RGBA 值来自 Device1，右侧 RGBA 值来自 Device2， 如图 18。

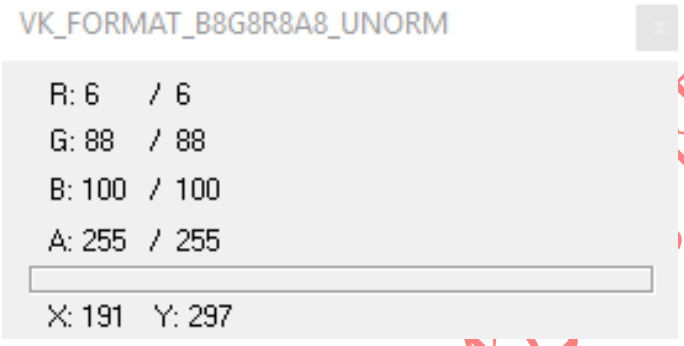

图 18

【Window】→【Show Graphics Event List】

选择脚本后点击 Graphics Event List，并点击 Load Next Frame 按钮，此时该脚本将被解析，并以树形视图展示。可以点击“+”号展开树节点，查看详细信息。若未勾选 Show All Events，树形视图仅显示渲染相关的 API；若已勾选，则显示所有 D3D API。可以点击树形视图中的任意指令，Player12.exe 将运行至选定的事件。例如：点击树中的某条 Draw 指令，即可查看该 Draw 的渲染结果。

【Window】→【Show Display Range】

从 RGBA 或 Depth/Stencil 中选择要显示的 Color Channel；同时可以通过 Min/Max Value 调整显示的 Color Channel 的值。

3.1.3 【菜单栏 Option】

3.1.3.1 【Options】→【General 页】

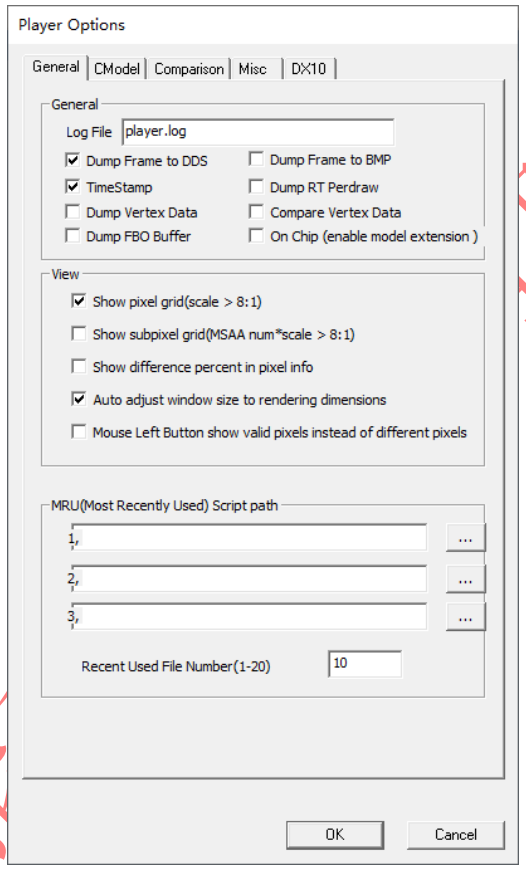

图 19

【General 区域】

Log File:

指定 Player 的输出日志文件。

Dump Frame to DDS:

在每帧结束时，将当前 RT 的渲染结果 dump 为外部 DDS 文件。文件路径为：Hardware Device 使用 script_path/../dump.hal，Reference Device 使用 script_path/../dump.ref。

Dump Frame to BMP:

在每帧结束时，将当前 RT 的渲染结果 dump 为外部 BMP 文件。

TimeStamp:

在 Draw 命令前后插入时间戳。

Dump RT Perdraw:

每次渲染操作执行完成后，将 RT 的渲染结果 dump 为外部 DDS 文件。

Dump Vertex Data/ Compare Vertex Data:

作用于内部 CModel 开发环境。

Dump FBO Buffer:

保留功能，暂不支持。

On Chip:

指定当前是硬件 Chip 环境，以启用 Model Extension 功能。

【View 区域】

Show pixel grid(scale > 8:1):

当纹理图像放大超过 8 倍时显示像素网格。

Show subpixel grid(MSAA num*scale > 8:1):

保留功能，暂不支持。

Show difference percent in pixel info:

若启用该设置，在像素信息窗口中，将显示不同渲染窗口中同一像素的差异百分比。

Auto adjust window size to rendering dimensions:

若启用此设置，主窗口将自动调整其大小以适配渲染窗口。但每个渲染窗口的最小尺寸限制为 320x320。若未启用此设置，且主窗口尺寸小于渲染窗口时，将显示滚动条。

Mouse Left Button show valid pixels instead of different pixels:

保留功能，暂不支持。

【MRU Script path 区域】

设置 3 个固定常用的脚本路径。用于快速选择脚本路径。

Recent Used File Number:

最近使用文件的显示个数。

3.1.3.2 【Options】→【CModel】

本 Tab 页用于内部 CModel 开发平台的设置， 此处不过多赘述。

3.1.3.3 【Options】→【Comparison】

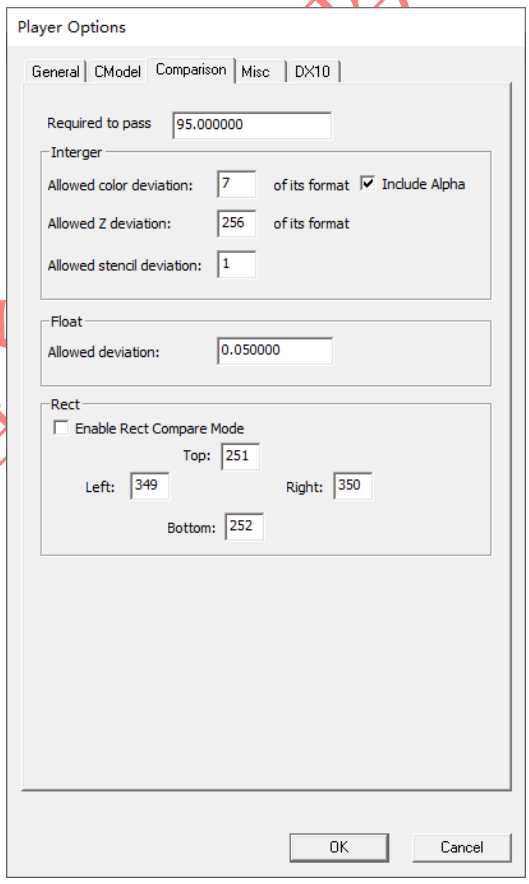

图 20

Required to pass XXX:

Player12.exe 将在状态栏和 player.log 日志文件中显示两个 Rendering Device 渲染结果的相似度比率。若该比率高于当前设定阈值，Player12.exe 将显示“Passed (xx%)”；否则，将显示“Failed (xx%)”。

【Interger 区域】

Allowed Color Deviation:

每个 Color Channel 允许的颜色偏差 x（以其格式为单位）：比较两个 Device 渲染结果的 Color Buffer 时，若同一像素的 R（或 G、B）值差值低于此设置，则该通道的相似度百分比视为 100%。

Include Alpha:

比较两个 Device 渲染结果时，若勾选此项，则比较将包含 Alpha 通道。

Allowed Z deviation:

保留功能，暂不支持。

Allowed stencil deviation:

保留功能，暂不支持。

【Float 区域】

Float Deviation:

比较两个 float 格式缓冲区时，若同一像素的浮点值差值低于此设置，则该像素的相似度百分比视为 100%。

【Rect 区域】

保留功能，暂不支持。

3.1.3.4 【Options】→【Misc】

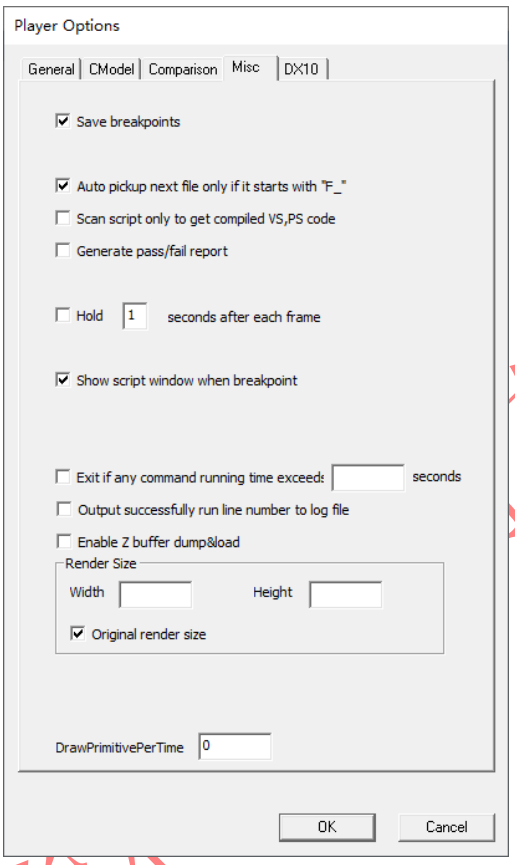

图 21

Save breakpoints:

保留功能，暂不支持。

Auto pickup next file only if it starts with “F_”:

保留功能，暂不支持。

Scan script only to get compiled VS,PS code:

保留功能，暂不支持。

Generate pass/fail report:

将比较结果写入 report.csv 文件。

Hold XXX seconds after each frame:

若启用此设置，Player12.exe 在【Run】→【Go（F5）】模式下，每帧结束后将暂停 n 秒。

Show script window when breakpoint:

Compare backbuffer MSAA border pixels in CIL-MODEL:

Exit if any command running time exceed n seconds:

Output successfully run line number to log file:

Enable Z buffer dump&load:

Render Size->Width/Height/Original render size:

DrawPrimitivePerTime XXX:

上述开关为保留功能，暂不支持。

3.1.3.5 【Options】→【DX10】

保留功能页，暂不支持。

3.1.4 命令行参数

运行“Player12.exe -help”输出 Player12.exe 支持的命令行参数。

图 22

3.2 跨设备回放任务

3.2.1 Player12.ini 配置文件

Player12.ini 是一个可选的文本文件，用于配置 Player12.exe 的一些特有功能，例如指定 Dump 范围、跨设备回放等。Ini 中描述了特有功能的详情及开关。需注意，该配置文件必须与 Player12.exe 放在同一路径中。本章将介绍一下使用 ini 文件开启跨设备回放功能。

3.2.2 跨设备回放的情况

- 涉及的资源： ID3D12Heap 和 ID3D12Resource;
- 涉及的 API:

ID3D12Device::CreateHeap()

ID3D12Device::CreatePlacedResource()

ID3D12Device::CreateReservedResource()

ID3D12CommandQueue::UpdateTileMappings()

- 当脚本中涉及到上述相关内容时，需要考虑跨设备回放模式。

3.2.3 如何运行跨设备回放

（1） 将 Player12.ini 放置到 Player12.exe 同路径下，并设置：

PL_SUPPORT_CROSS_DEVICE=1

PL_PREPASS_FOR_CROSS_DEVICE=1

（2） 启动 Player12.exe 正常回放脚本。回放完成后，在脚本路径中生成 prepass.txt 文件，该文件记录了当前回放设备对于相关 Resource 需要调整的参数。

（3） 将 prepass.txt 重命名为 insert.txt（可能还需重命名脚本），并在 Player12.ini 设置：

PL_PREPASS_FOR_CROSS_DEVICE = 0.

（4） 重新启动 Player12.exe 以正常模式回放脚本。完成跨设备回放。

3.3 脚本命令实现具体任务

3.3.1 Insert.txt

和 2.2 章节介绍的 Player 系列工具支持自定义 API 命令一样，Player12.exe 也支持一些自定义的 API 命令。你可以将自定义的 API 命令直接写到脚本中，也可以使用 insert.txt 文件向脚本中插入命令。insert.txt 可包含多行，每行的格式如下：

frameId, LineId, scriptcommand

如果脚本所在文件夹中存在 insert.txt，Player12.exe 将解析该文件，并在指定 frameId 的脚本中、在指定 LineId 之前插入该行脚本命令。

注意：

（1） 如果脚本文件名**不是**以数字结尾，frameId 需设为 -1。

（2） LineId 从 0 开始计数；

（3） 若要在第 0 行之前插入命令，LineId 需设为 0；

（4） 若要在帧的最后一行之后插入命令，请设置 LineId=lastLineId+1；

insert.txt 的示例如下：

0, 135, plDumpResource(cmdQ_1, resC_6, DXGI_FORMAT_UNKNOWN, 0, "resC_6_L135.dds")

1, 64, plBreak()

//示例: Player12.exe 将在 F_0.d3d 的第 135 行之前插入 plDumpResource 命令，并在 F_1.d3d 的第 64 行之前插入 plBreak 命令。

3.3.2 自定义脚本命令

在这里列出 Player12.exe 常用的实现具体任务的一些自定义命令名称及功能，不再详细展开每个命令的参数及取值，用户可在《D3D 脚本语言指南》中查阅详细的参数信息。

自定义脚本命令	功能
通用自定义脚本命令	
plBreak	Player12.exe 停下并等待继续运行命令。
plRelease	Release D3D resource, Resource 的 ref count 变回 0。
plCreateShaderFromFile	从外部文件创建指定的 shader 类型，例如 VS, PS, HS, Gs 等。
plSuspendRendering	挂起渲染命令。
plResumeRendering	恢复渲染命令。
plSetString	用于设置搜索资源文件的相对路径。
plSetUINT	plSetUINT(PL_ FREE_PTR_AUTO, 0/1) 是否释放 define pointer 的 memory
plDefineUINT	为变量赋值， 示例： plDefineUINT(“name_0”, 0)
plFreePtr	释放某个 define pointer 的 memory
D3D10 现代语法脚本自定义命令	
plPresent10	用指定的 format 显示 resource 的某个 subresouce。
D3D11 现代语法脚本自定义命令	
plPresent11	用指定的 format 显示 resource 的某个 subresouce。
plDumpResource11	Dump Resource 数据。

plCompareResource11	将 Resources 数据和外部文件做对比。
D3D12 脚本自定义命令	
plDumpEveryRTAfterEveryDraw	每一个 Draw 命令后 dump RTV、DSV，在 Player.ini 中设置 dump 范围。
plDumpEveryRT	CmdList 更换 RT 或 Cmlist Close 时，dump RT 最后的渲染结果。
plUpdateSubresources	在 cmdlist 内部执行，更新 resource 的目标 subresource 数据
plUpdateSubresourcesFromFile	在 cmdlist 内部执行，更新 resource 的目标 subresource 数据
plPresent	用指定的 format 显示 resource 的某个 subresouce。
plExecuteCommandListToDraw	执行 CmdList 中的渲染命令的截止点。
plDumpResource	立即 Dump Resource 数据。
plDumpResourceInCmdList	在 cmdlist 内部执行，Dump Resource 数据。
plCompareResource	将 Resources 数据和外部文件做对比。
plRealUpdateSubresources	立即更新 resource 的目标 subresource 数据
plRealUpdateSubresourcesFromFile	立即更新 resource 的目标 subresource 数据

表 5