khw7096 Update usd.md 63ea4af 8 days ago

1 contributor

271 lines (216 sloc) | 6.44 KB

USD

Universal Scene Description의 약자입니다. 여러분이 아마도 TD 업무를 하면서 미래에는 이 포멧을 자주 다루게 될것 같습니다. 픽사의 3D 파이프라인 코어입니다. 아래 장점들을 가지고 있습니다.

- 일반 언어형태로 사용할 수 있습니다.
- 3D 데이터를 패키징 할 수 있습니다.
- 3D 데이터를 Assembling(조합) 할 수 있습니다.
- 3D 데이터를 편집할 수 있습니다.
- 여러 툴을 이용해서 콘텐츠를 만들 수 있습니다.(아직 USD가 활발하게 모든 툴에 탑제 되려면 조금더 시간이 필요하긴 합니다.)
- 여러 아티스트가 하나의 에셋을 작업할 수 있습니다.
- 데이터 로딩시 소요되는 시간을 최소화할 수 있습니다.
- 사용자가 커스텀하게 원하는 데이터를 추가할 수 있습니다.(굉장히 강력한 기능입니다.)

홈페이지 : https://graphics.pixar.com/usd/docs/index.html

지원하는 프로그램

- 마야
- 후디니
- 카타나
- Unreal : UE4\Plugins\Editor\USDImporter 파일은 존재함. 준비중.
- Renderman

파일의 종류

- .usd
- .usda : 아스키 파일
- .usdc : USD Crate 파일(바이너리 파일)
- .usdz : .usd Zip 압축파일

PyOpenGL 설치

usdview는 PyOpenGL을 사용합니다. 설치해주세요.

```
# pip install PyOpenGL
```

컴파일

굉장히 오래걸립니다. 기존에 배운것들을 실습하거나 자습하며 컴파일을 진행합니다.

```
$ cd ~/app
$ git clone https://github.com/PixarAnimationStudios/USD USD_src
```

```
$ cd USD_src
$ python build_scripts/build_usd.py --alembic --openimageio ~/app/USD
```

컴파일이 되면 아래 리스트가 설치됩니다.

```
STATUS: Installing boost...
STATUS: Installing TBB...
STATUS: Installing OpenEXR...
STATUS: Installing Alembic...
STATUS: Installing GLEW...
STATUS: Installing OpenSubdiv...
STATUS: Installing JPEG...
STATUS: Installing TIFF...
STATUS: Installing PNG...
STATUS: Installing OpenImageIO...
STATUS: Installing USD...
```

터미널에서 2개의 환경변수값을 추가합니다.

```
$ export PYTHONPATH="${PYTHONPATH}:$HOME/app/USD/lib/python"
$ export PATH="${PATH}:$HOME/app/USD/bin"
```

```
$ usdview extras/usd/tutorials/convertingLayerFormats/Sphere.usda
```

화면처럼 Usdview가 잘 뜨면 위 환경변수를 자신의 .bashrc에 잘 추가해두세요. 우리는 많은 기능을 활성화해서 컴파일하지 않았습니다. 더 많은 기능을 이용하고 싶다면 사용하는 라이브러리를 추가 설정하여 다시 컴파일하면 됩니다.

명령어

sdfdump

usd파일에 대한 Report를 출력합니다.

```
$ sdfdump input.usd
```

output

```
@test.usd@
</> : SdfSpecTypePseudoRoot
  primChildren: vector<TfToken, allocator<TfToken> > = [ Box ]
</Box> : SdfSpecTypePrim
  specifier: SdfSpecifier = SdfSpecifierDef
  typeName: TfToken = Cube
  properties: vector<TfToken, allocator<TfToken> > = [ xformOp:scale xformOpOrder ]
</Box.xformOp:scale> : SdfSpecTypeAttribute
  custom: bool = 0
  typeName: TfToken = float3
  variability: SdfVariability = SdfVariabilityVarying
  default: GfVec3f = (5, 5, 5)
</Box.xformOpOrder> : SdfSpecTypeAttribute
  custom: bool = 0
  typeName: TfToken = token[]
  variability: SdfVariability = SdfVariabilityUniform
  default: VtArray<TfToken> size 1
```

sdffilter

필터링하여 데이터를 검색 다른 데이터로 아웃풋 할 수 있다.

```
$ sdffilter
```

stringify

파일에 어떤 문자열이 있는지 전부 출력한다. 엔터역시 `\n` 으로 출력된다.

```
$ stringify test.usd
```

testusdview

```
$ testusdview --testScript scriptpath input.usd
```

usdcat

리눅스의 cat 과 비슷한 명령어 입니다. usdz 파일도 아스키로 보여줍니다.

```
$ usdcat input.usd
$ usdcat input.usdz
```

abc를 usd로, usd를 abc로 바꿀 수 있습니다.

```
$ usdcat -o ouput.usdc input.usda
$ usdcat input.usda -o output.abc
$ usdcat input.abc -o output.usda
```

usdchecker

usd 데이터에 이상이 있는지 체크합니다.

```
$ usdchecker input.usd
```

아래 문장이 나오면 파일에 이상이 없다는 것 입니다.

```
Success!
```

usddiff

```
usddiff input1.usd input2.usd
```

만약 다르다면 다른 부분만 아래처럼 출력됩니다.

```
@ -5 +5 @@
-    float3 xformOp:scale = (5, 5, 5)
+    float3 xformOp:scale = (10, 5, 5)
```

usdedit

usd파일을 읽을 수 만 있는 에디터를 실행합니다. USD_EDITOR 환경변수로 잡혀있는 에디터를 이용해서 usd파일을 엽니다. 만약 편집하고 싶다면 `-f` 옵션을 넣고 실행합니다.

```
$ usdedit input.usd
```

```
$ usdedit -f input.usd
```

usdstitch

각각의 usd 파일을 묶는 명령어입니다. 프레임 데이터가 충돌하면 strong layer를 우선시 하여 연산합니다.

```
$ usdstitch -o output.usd in.0001.usd in.0002.usd ....
```

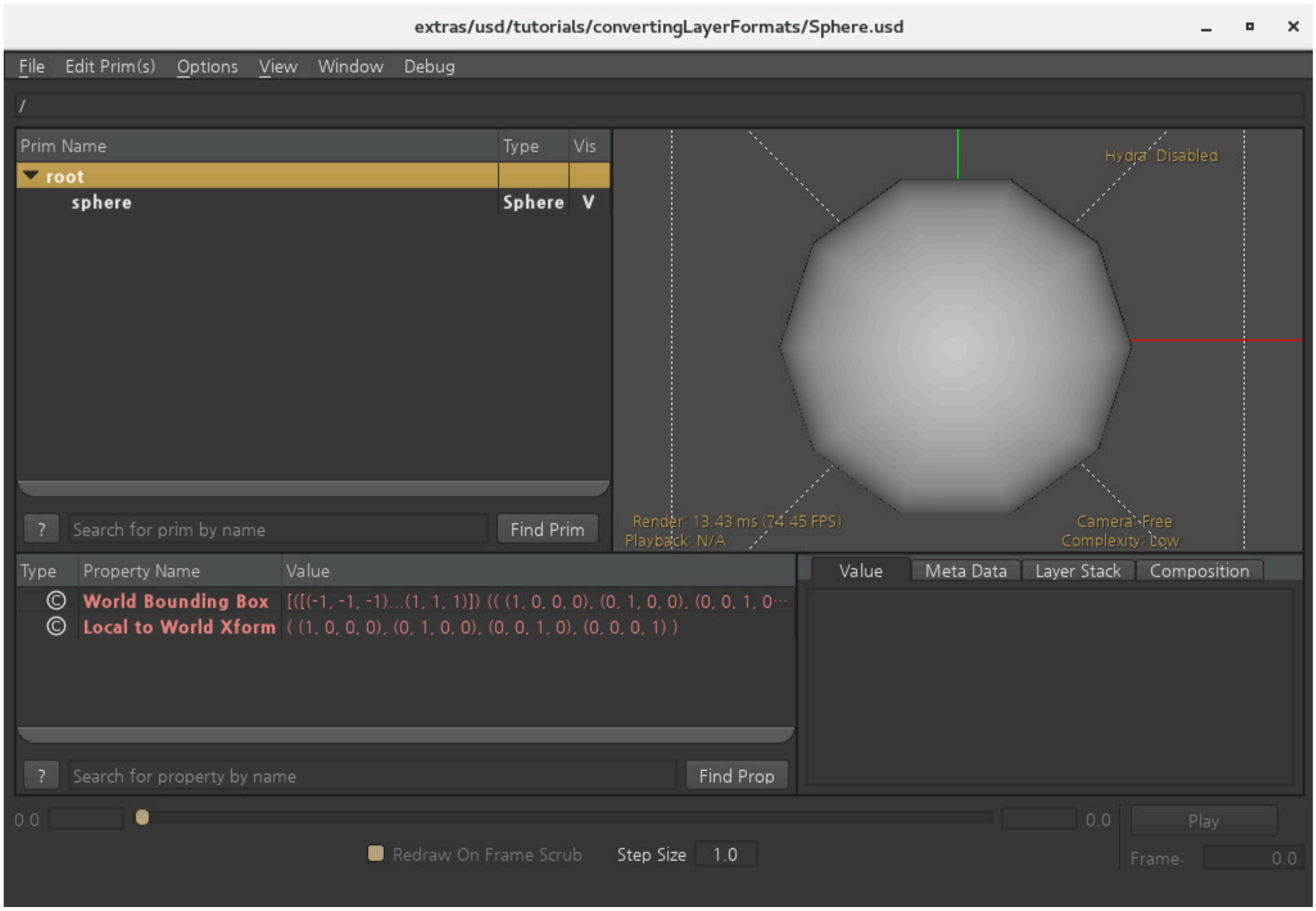
usdstitchclips

여러 usd파일을 합쳐서 하나의 클립으로 제작합니다.

```
$ usdstitchclips -o output.usd --clipPath /path/clip1.usd clip2.usd
```

usdview

usd 파일을 볼 때 사용합니다.




```
$ usdview input.usd
$ usdview input.abc
```

usdzip

에셋들을 하나의 .usdz 파일로 만들 때 사용합니다.

```
$ usdzip output.usdz asset1.usd asset2.usd
```

파일구조

usd파일은 아래와 같은 형태를 가지고 있습니다.

```
#usda 1.0

class "_class_Planet"
{
    bool has_life=False
}

def Xform "SolarSystem"
{
    def "Earth" (
        references = @./planet.usda@</Planet>
    )
    {
        bool has_life = True
        string color = "blue"
    }

    def "Mars" (
        references = @./plant.usda@</Plant>
    )
    {
        string color = "red"
    }

    def "Saturn" (
        references = @./plant.usda@</Plant>
        variants = {
            string rings = "with_rings"
        }
    )
    {
        string color = "beige"
    }
}
```

```
#usda 1.0

def Cube "Box"
{
    float3 xformOp:scale = (5, 5, 5)
    uniform token[] xformOpOrder = ["xformOp:scale"]
}
```

Sample 파일 다운로드

- http://graphics.pixar.com/usd/downloads.html

Reference

- https://github.com/PixarAnimationStudios/USD
- https://github.com/vfxpro99/usd-build-club
- https://github.com/vfxpro99/usd-build-club/tree/master/prerequisites-linux
- https://github.com/meshula/mkvfx
- https://graphics.pixar.com/usd/overview.html