

Agenda

01 **Background**
02 **Technical Details**
03 **Plan and Progress**
04 **Questions**

Background

Motivation

- Some computational behaviors in schema methods already
- Desire to grow computational capabilities (e.g., constraints)
- Need for richer object modeling of computational behaviors
- Potential future rig interchange

A general purpose framework for computational behaviors in USD.

Scope of OpenExec

What it IS

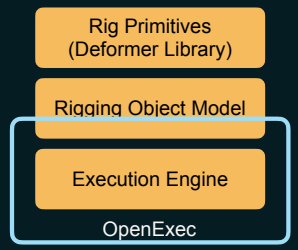
+ Outfit USD schemas with your own computations
+ Scenes provide computed values in addition to authored values
+ Arbitrary data types and custom C++ callback code
+ High performance evaluation, automatic caching and invalidation

What it IS NOT

- Computations are value-driven not event-driven
- Does not procedurally generate or modify the scene graph (outputs values, not topology)
- Not a complete rigging system

Notes on Rigging

- “Potential future rig interchange”
- Three major pieces to a rigging system
- Immediate goal of OpenExec is to lay a foundation
- Later work with community to figure out if Pixar’s rigging OM even makes sense outside the studio
- No immediate plans to open-source Pixar’s library of deformers



Technical Details

© 2024 Disney/Pixar. All rights reserved.

Distribution

- OpenExec will ship with OpenUSD distribution
- New set of libraries in pxr/usd/exec



vdf
vectorized data flow network, scheduling and evaluation



ef
execution foundation introduces higher-level concepts, such as specialized data flow node types

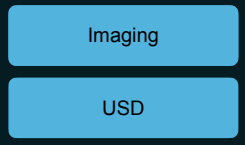


exec
client-facing, high-level execution library for registration of behaviors and requesting of values



© 2024 Disney/Pixar. All rights reserved.

Library Stack





© 2024 Disney/Pixar. All rights reserved.

Summary of New Concepts

- Named computations
- Static, declarative input parameters to computations
- Custom computation callbacks
- Execution behavior registration as an extension to USD schema registration
- Actions for “driving” of computed values, and execution markers to locate the source of a computed value in a stack of actions
- Client API for requesting computed values, as well as invalidation notification



© 2024 Disney/Pixar. All rights reserved.

Named Computations

- Any USD scene object can provide an arbitrary number of named computations

```
ExecValueKey(SdfPath providerObject, TfToken computationName)
```

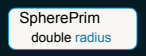
- Computations declare input dependencies as input parameters
- Callbacks do the computational work
- Published from two sources:
  1. Built-in computations: Hard-coded set of computations available on every USD object
  2. Registered computations: Registered through extension of schema registration



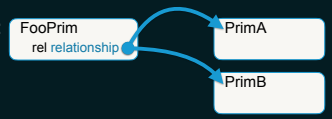
© 2024 Disney/Pixar. All rights reserved.

Input Parameters

- Computations can register one or more input parameters
- Declarative, a-priori, static input dependencies
- Every input parameter flows one or more input values to the computation callback
- Many different types of inputs: For example, computation input
- For example, attribute input:



- For example, uses relationship:





© 2024 Disney/Pixar. All rights reserved.

Computation Callbacks

- C++ function object performing the computational work

```
static void MyCallback(const VdfContext &ctx) {
    // Read input value from 'myInput' input parameter
    const int inputValue = ctx.GetInputValue<int>(tokenSet->myInput);

    // Set an output value
    ctx.SetOutput(inputValue + 1);
}
```

- Stateless, side-effect free, and concurrently invoked blackboxes
- Constant input values accessible via `VdfContext`
- Output value(s) support arbitrary data types, ownership transfers to evaluation engine
- Automatic caching and invalidation of output values



© 2024 Disney/Pixar. All rights reserved.

Execution Behavior Registration

- Extends USD schema registration with ability to register computations in C++ schema class files

```
// --(BEGIN CUSTOM CODE)--
TF_REGISTRY_FUNCTION(ExecRegistration)
{
    // Register computational behaviors on UsdPrimSchema.
    ExecRegistration().Register<UsdPrimSchema>()

      .Computation(tokens->myComputation)
        .Callback(&MyCallback)
        .AttributeInput<int>(UsdPrimSchema->someAttr)
        .ComputationInput<MyCustomStruct>(tokens->myOtherComputation)
```

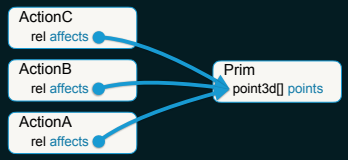
- C++ meta-programming based domain specific registration language
- Limited to codeful schemas



© 2024 Disney/Pixar. All rights reserved.

Actions and Execution Markers

- Actions enable “driving” (“affecting”) of attribute values
- Can be stacked to form a chain of actions executing in reverse namespace order



- Execution markers locate computed values in a chain of actions (@name-of-action)
- Canonical execution markers, e.g. @final



© 2024 Disney/Pixar. All rights reserved.

Client API

- `ExecSystem` as state-holder and primary gateway into OpenExec

```
ExecSystem exec(stage);
```

- Optimized for making batch requests for computed values

```
ExecRequest request = exec.BuildRequest({
    ExecValueKey(myPrimA, MyPrimASchemaComputationTokens->foo),
    ExecValueKey(myPrimB, MyPrimBSchemaComputationTokens->bar),
    . . . });
exec.Compute(request);
```

- Value extraction API through index (fast-path), or `ExecValueKey` inversion

```
VtValue fooValue = exec.ExtractValue(request, 0);
VtValue barValue = exec.ExtractValue(request, 1);
```

- Pay for what you use



© 2024 Disney/Pixar. All rights reserved.

Three Stages of Execution



Compilation
turns scene objects and execution registration into data-flow network and maintains said network



Scheduling
given a request for values, builds an acceleration structure for optimal network traversal during evaluation

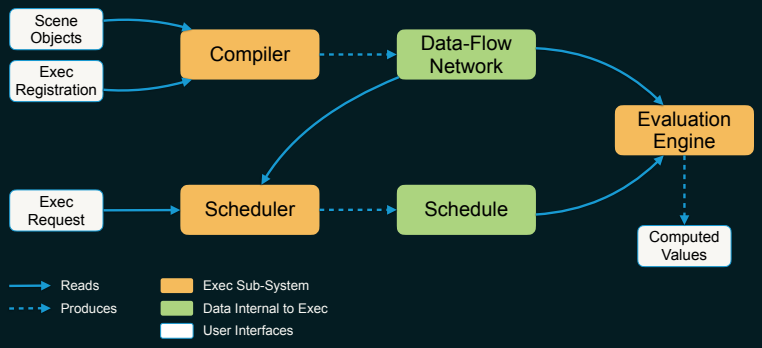


Evaluation
runs callbacks included in the schedule to compute uncached values, and populates caches



© 2024 Disney/Pixar. All rights reserved.

System Architecture





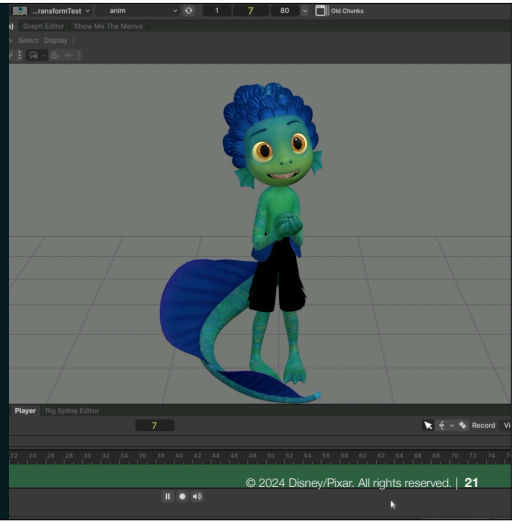
© 2024 Disney/Pixar. All rights reserved.

Plan and Progress

© 2024 Disney/Pixar. All rights reserved.

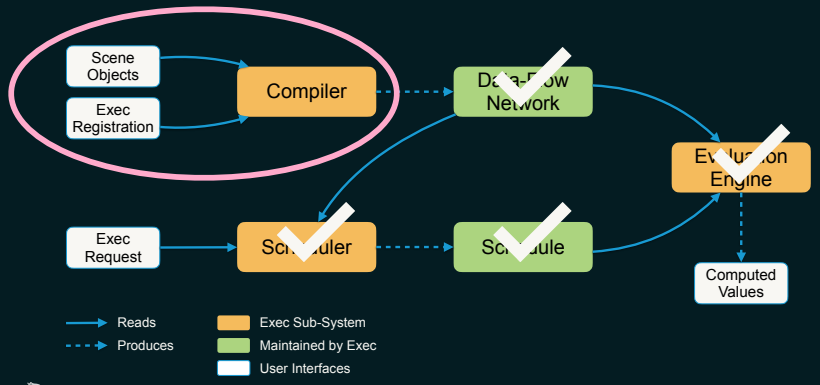
Presto Execution System

- OpenExec, aka the Presto Execution System
- Open-source parts of this existing system
- Adjust to USD-isms where appropriate
- Leverage community to improve and grow



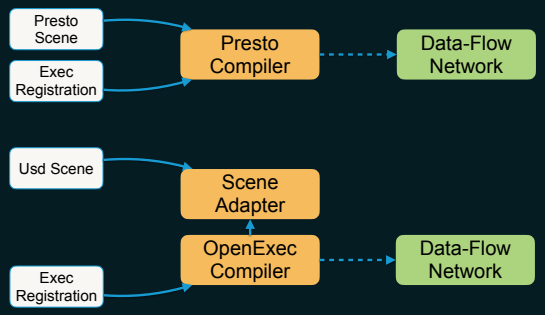
© 2024 Disney/Pixar. All rights reserved.

Progress



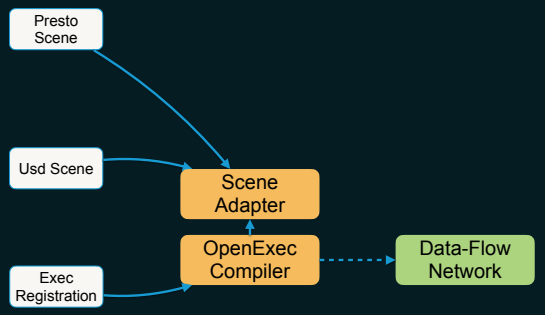
© 2024 Disney/Pixar. All rights reserved.

OpenExec Compiler



© 2024 Disney/Pixar. All rights reserved.

OpenExec Compiler



© 2024 Disney/Pixar. All rights reserved.

“Minimally Featured Product”



Minimally Featured Product

- Compute local- and world-space transforms on UsdGeomXformable

```
bool UsdGeomXformable::GetLocalTransformation(GfMatrix4d *transform, . . .);
```

- Introduce concepts and provide client API to achieve at least this
- Aiming for 25.08 / SIGGRAPH 2025 OpenUSD release



© 2024 Disney/Pixar. All rights reserved.

Questions?

