



Esri International Developer Summit
Palm Springs, CA

Developing with the CityEngine SDK

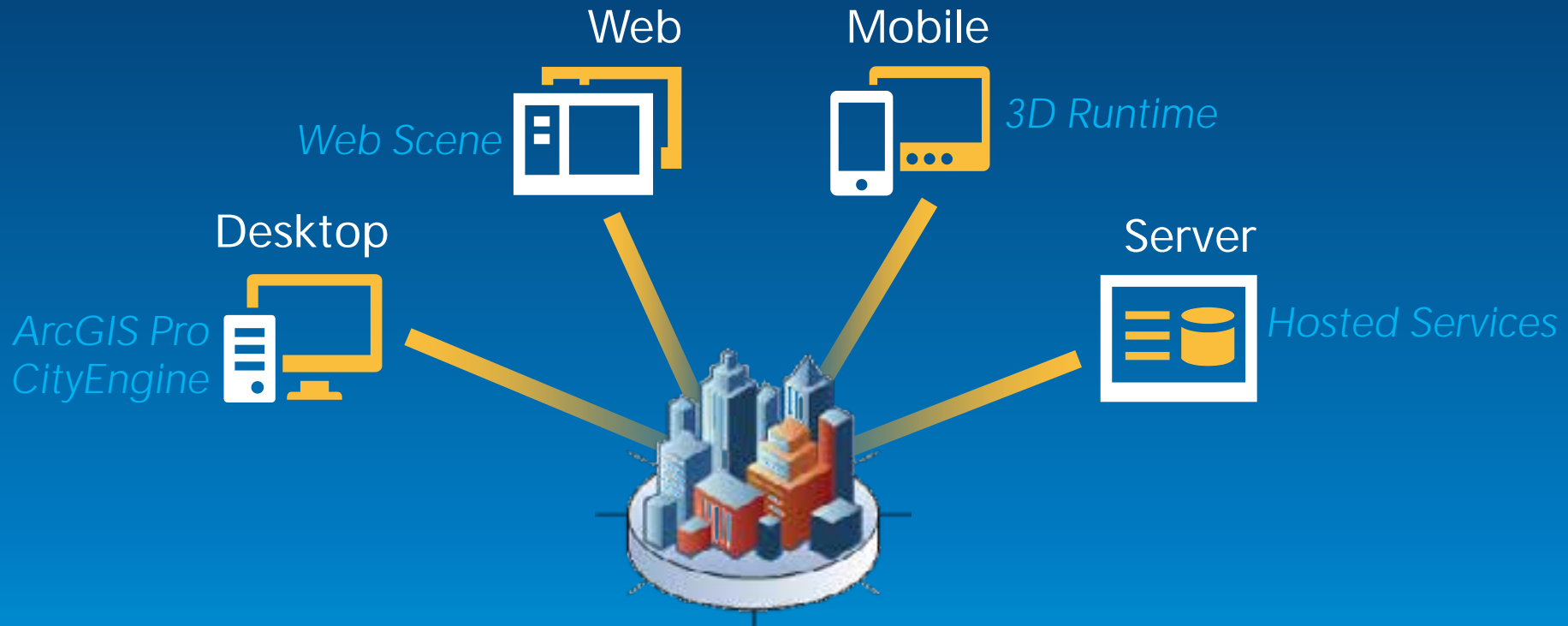
Matthias Specht, Gert van Maren

Esri R&D Center Zurich

Agenda

- **Examples (5 min)**
- **Introduction SDK (5 min)**
- **SDK architecture, code samples (10 min)**
- **Creating Apps!!! (30 min)**
 - hello world
 - STL -> show in CityEngine
 - Maya

2014 - 3D Across the Platform



CityEngine SDK Use Cases

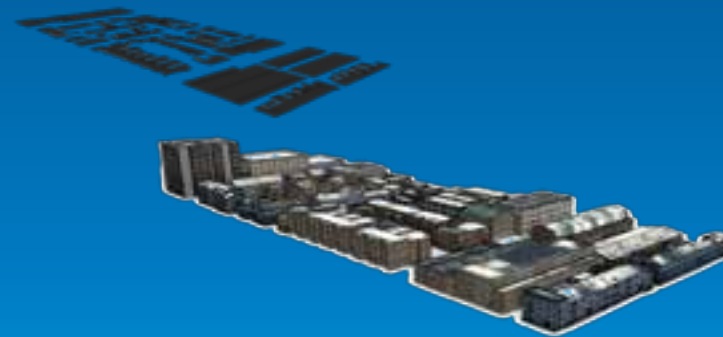
- **Entertainment Pipelines**

- DLL, no ArcGIS e.g. proprietary exporters or rendering with generation on demand



- **ArcGIS Desktop**

- Attribute-driven building geometries in ArcGIS
- Extend CityEngine with new import / export formats



- **ArcGIS Server**

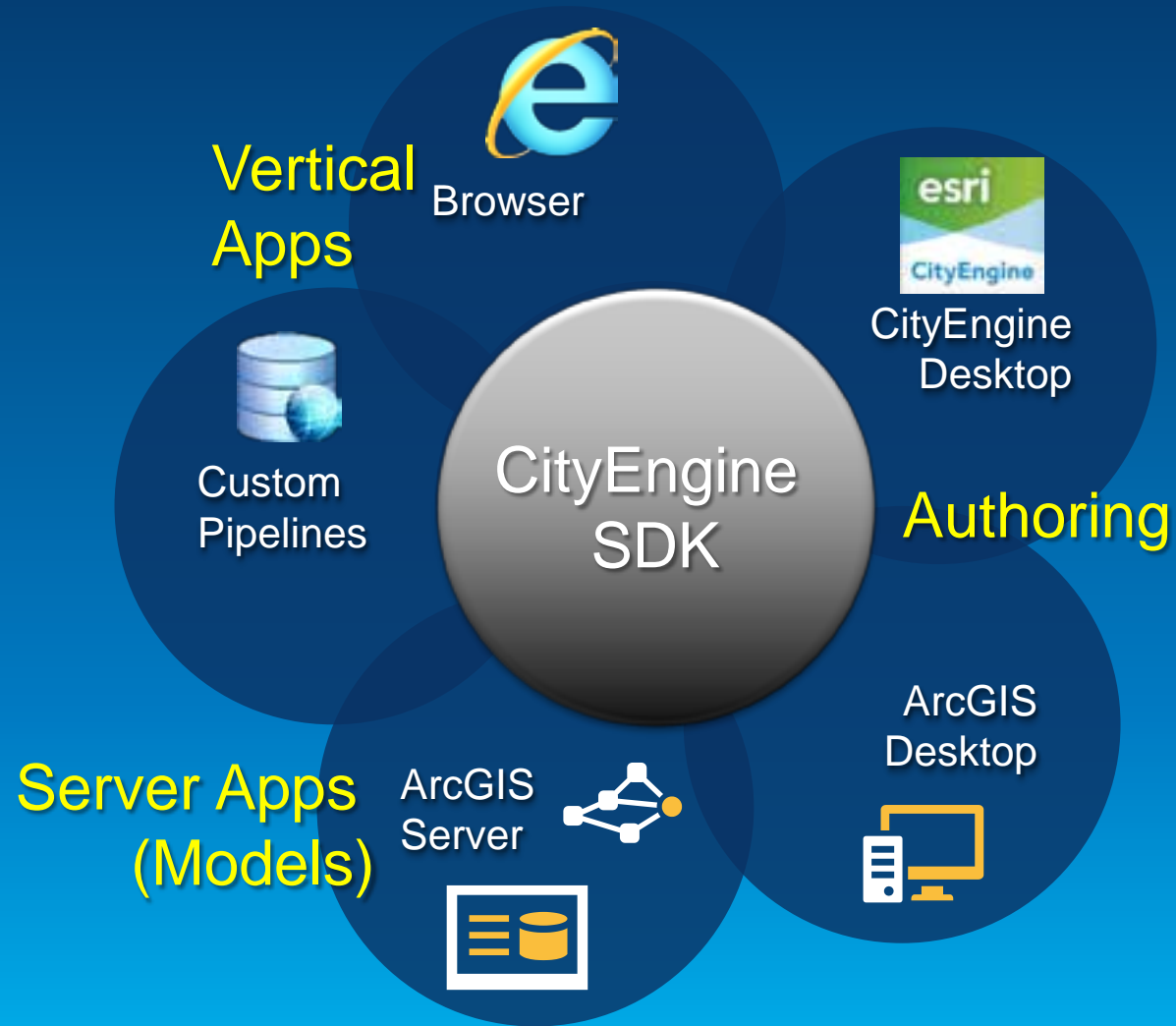
- CityEngine GP Service



What is the CityEngine SDK

- **Procedural geometry generation engine at the heart of CityEngine 2013 and two GP tools in ArcGIS 10.2, included in ArcPro**
- **Set of header files and libraries to use procedural technology without ArcGIS or CityEngine in your client application**
- **Allows extending CityEngine with additional import and export formats and storage backends beyond simple files**

CityEngine SDK Eco-System



Motivation: Procedural Model Generation

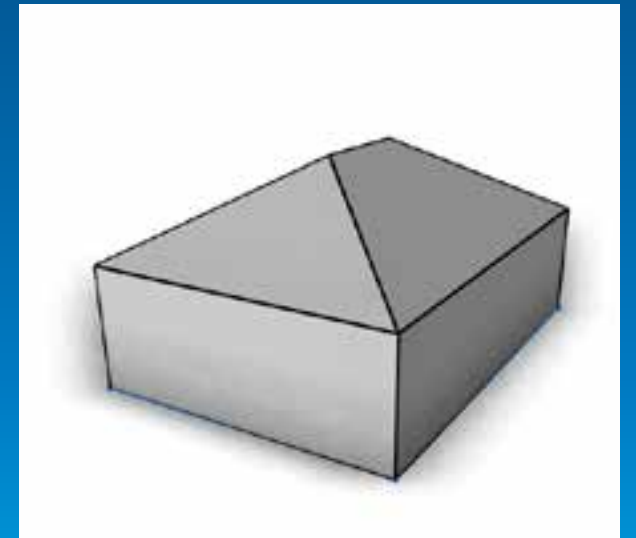
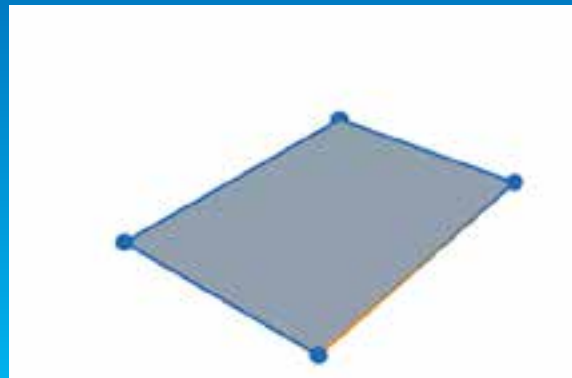
- CGA Rule + initial shape + attributes => 3D model

```
attr height = 15
attr angle = 35

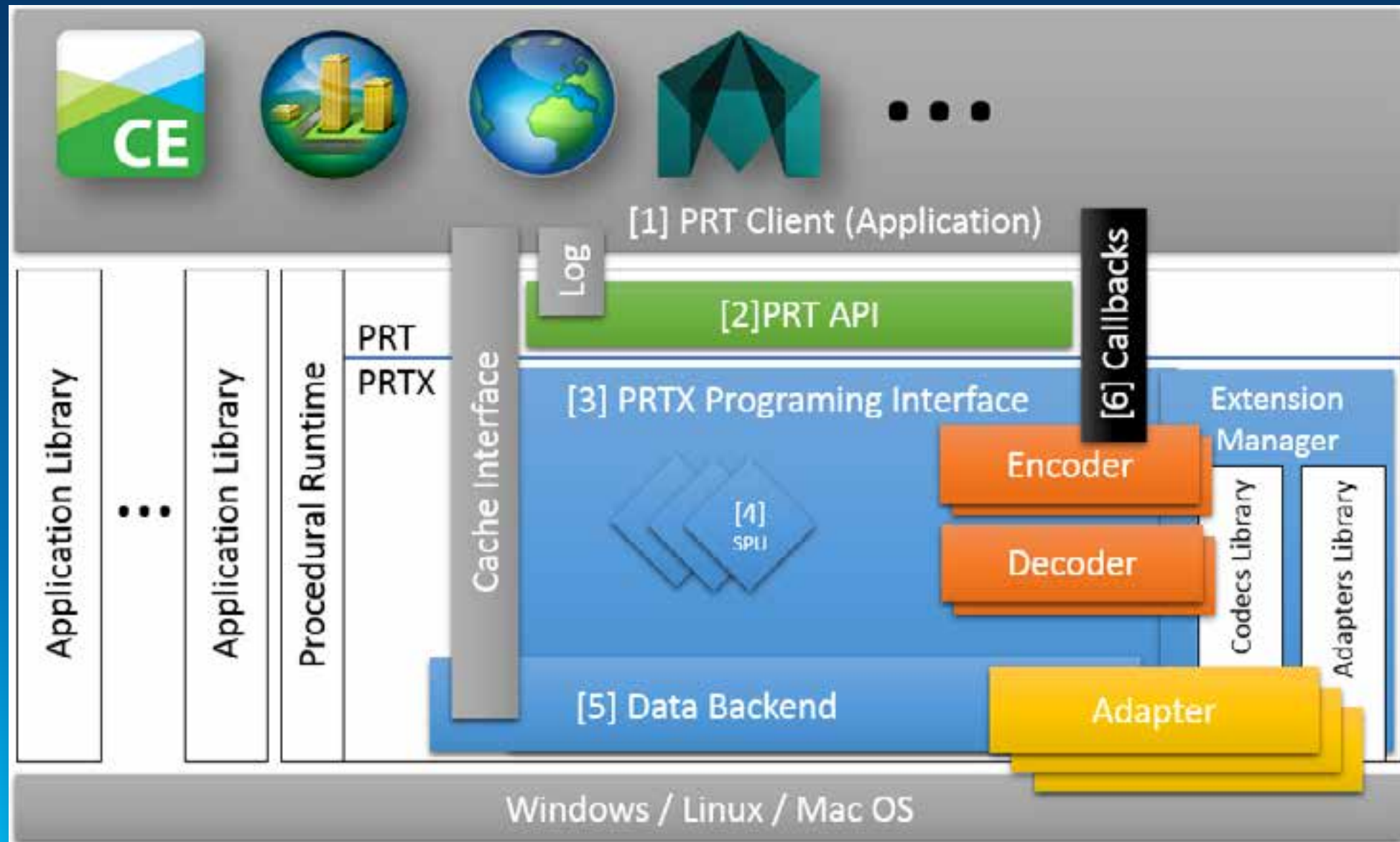
Init-->
  extrude(height)
  comp(f) {
    side : Facade. |
    top  : Roof
  }

Roof-->
  roofHip(angle)
```

angle 25
height 15



Procedural Runtime Overview



PRT and PRTX

- **Two worlds from a developer point of view**
- **PRT: API used for**
 - Procedural generation of 3D content
 - Runtime queries
- **PRTX: Extension interface**
 - Encoders
 - Decoders
 - Adapters

Use cases

Use case	PRT	PRTX
Procedural 3D model generation in custom client application with the supplied codecs and callbacks	<ul style="list-style-type: none"> - Invoke PRT API from client application - Use one of the supplied callback implementations for results 	
Extend CityEngine with custom export functionality		Write encoder for custom format
Extend CityEngine with custom asset reader functionality		Write decoder for custom format
Integrate procedural 3D model generation with existing Digital Content Creation (DCC) application	<ul style="list-style-type: none"> - Invoke PRT API from DCC application - Implement custom callback for transferring 3D geometry data back to DCC application 	Write encoder for in-memory 3D geometry data suitable for custom callback implementation
Create a language binding for PRT	<ul style="list-style-type: none"> - Wrap PRT API calls for target language - Implement custom callback for transferring 3D geometry data back to the caller 	Write encoder for in-memory 3D geometry data suitable for custom callback implementation
Add custom asset repository access	Use PRT API with URI scheme for custom asset repository	Write and register adaptor for custom asset repository URI scheme
Custom (persistent) cache implementation / caching policy	Implement the PRT cache protocol	

Procedural Runtime API

- `prt` namespace
- Coarse grained
- Minimized use of C++ (C-style API)
 - No STL
 - No exceptions, all API calls return a status value
- Memory allocation / de-allocation hidden (compiler firewall)
 - `create()` and `destroy()`
 - Client's responsibility
 - Client and PRT can use different C-runtimes
- ABI-level compatibility for compilers and linkers
- Thread-safe

PRT: Data Types

- Most PRT objects are immutable, use builders for construction
- PRT is stateless, client can manage state
- URIs for identifying resources
- Double precision floating point numbers
- Coordinates: Cartesian, Y-up, right handed, metric
- Wide strings => UTF16 support (basic multilingual plane)
- `toXML()` for debugging, logging, and serialization

PRTX – The Procedural Runtime Extension Interface

- `prtx` namespace
- Large collection of classes and functions
- Fine grained access to PRT data structures
- C++03 including exceptions
- Memory management:
 - Same CRT for core & extensions
 - Shared pointers - generally no need to worry about memory management
- Some STL and boost classes
- Must be compiled & linked with same settings as PRT
- Plugin-in mechanism, usually a shared library per extension

Inside Procedural Model Generation: Shape Processing

- Rule + initial shape + attributes => shape tree => 3D model

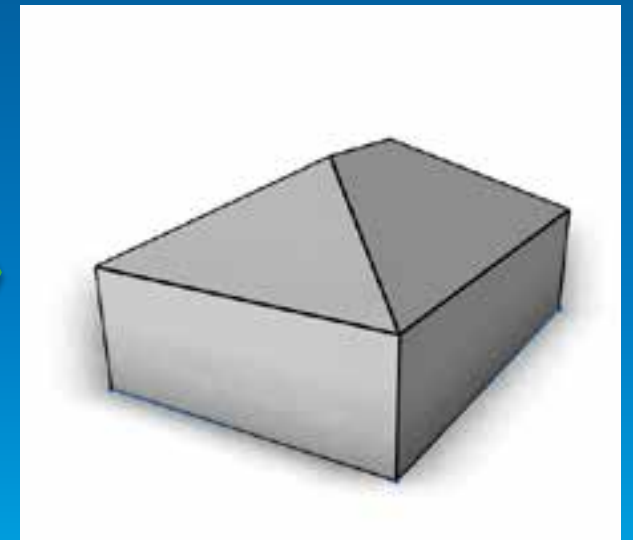
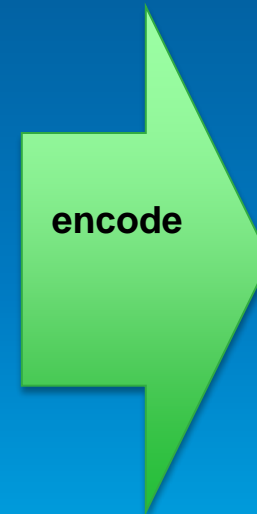
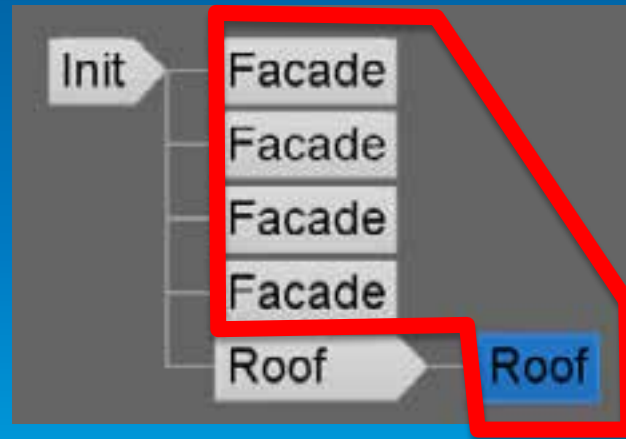
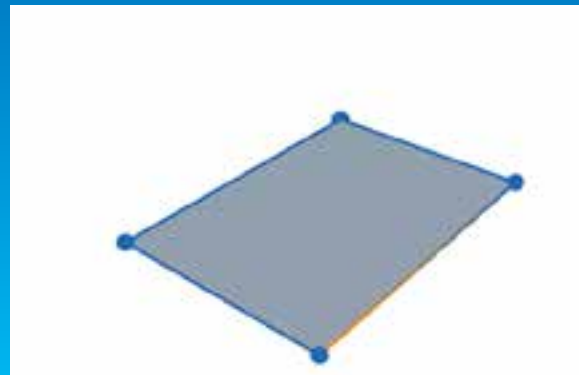
```
attr height = 15
attr angle = 35

Init-->
  extrude(height)
  comp(f) {
    side : Facade. |
    top : Roof
  }

Roof-->
  roofHip(angle)
```

angle

height



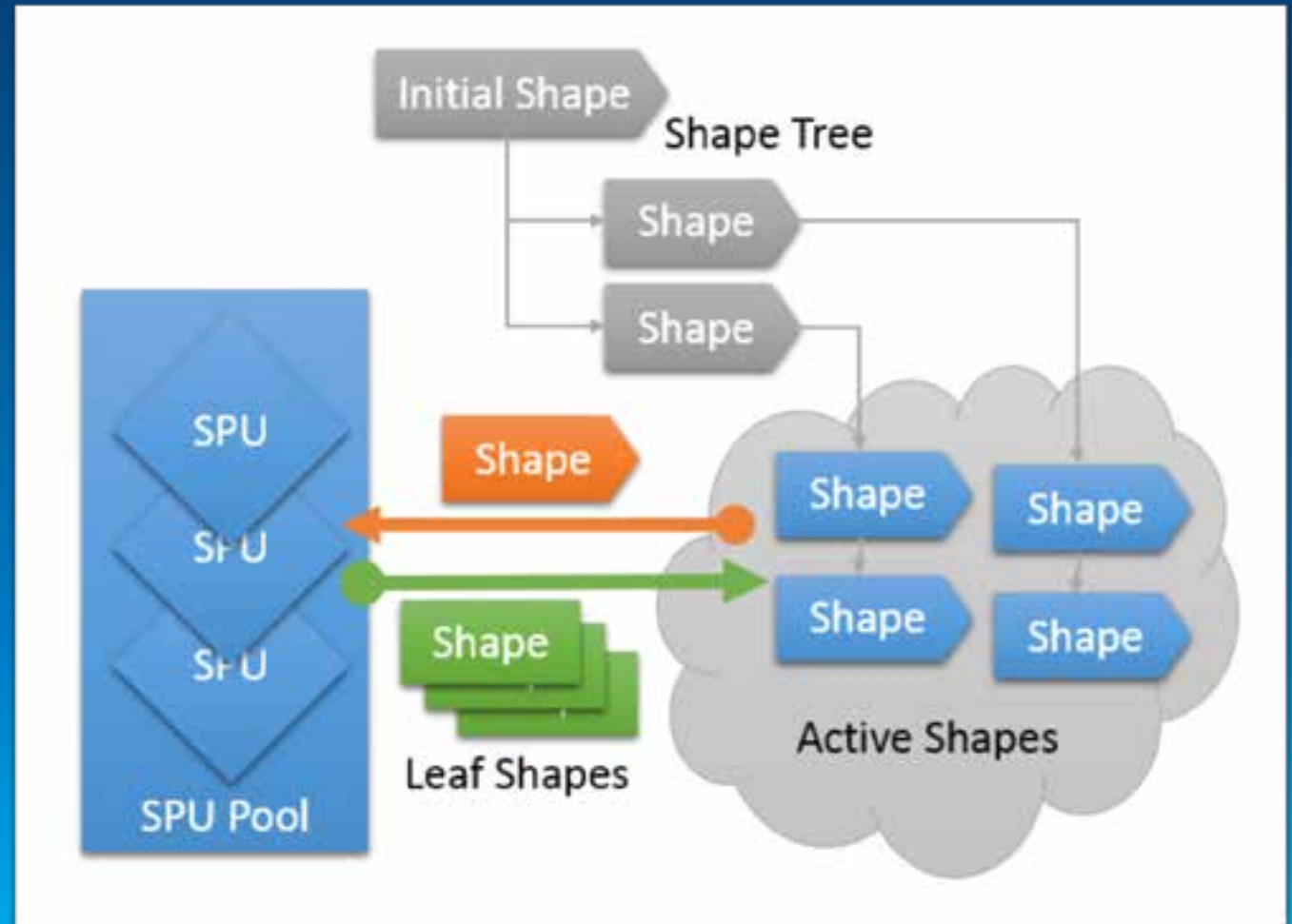
CGB Rule Files

- **Compiled form of CGA files**
- **Only rule format that PRT can process**
- **Based on Java class file format (bytecode)**
- **PRT API allows introspection of CGB**



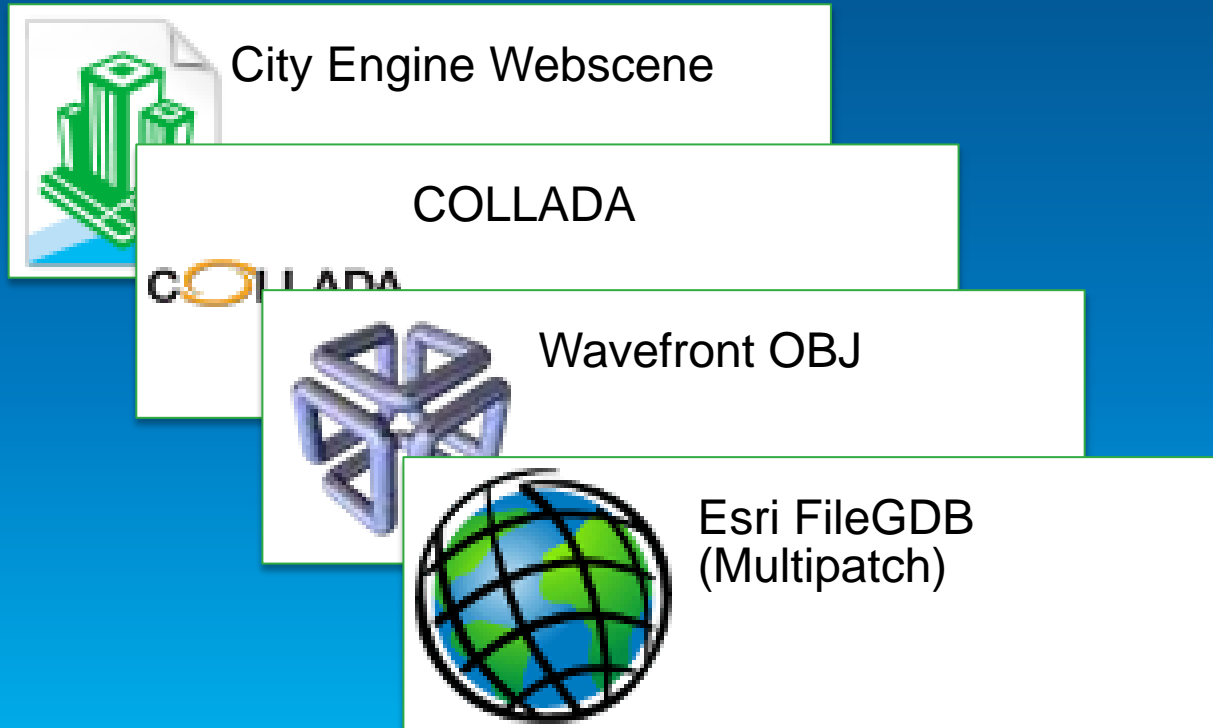
Shape Processing Units

- Virtual machine to execute CGB bytecode
- 1 Shape in, n Shapes out



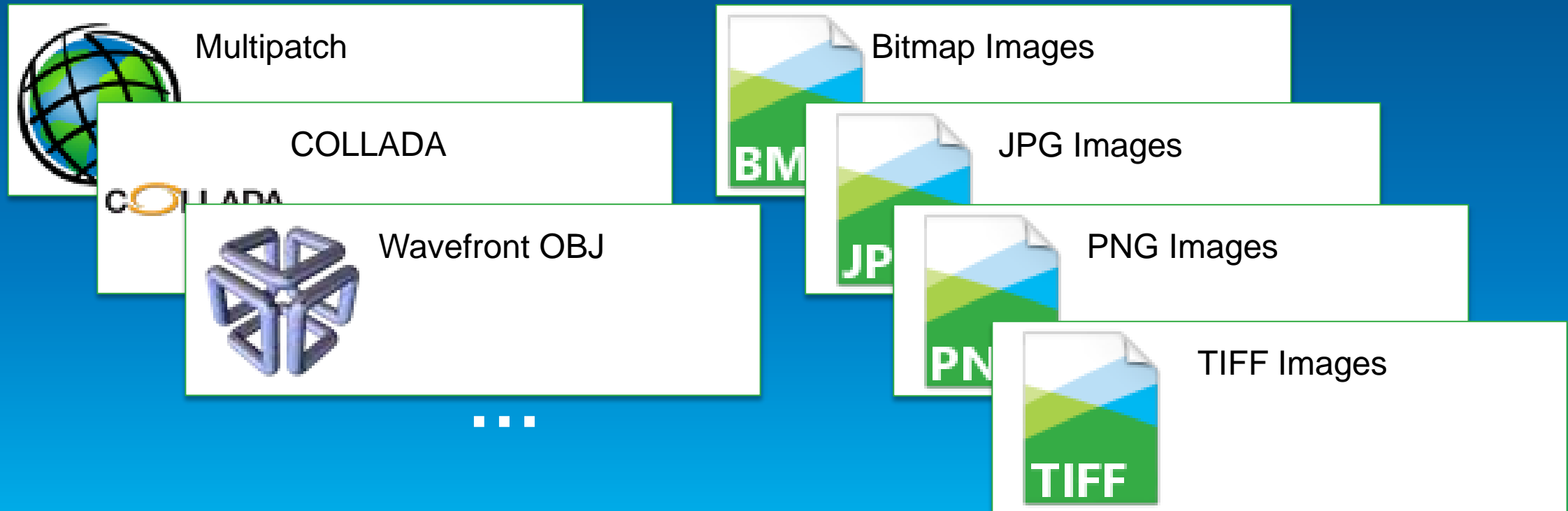
Encoders

- Extract geometry (or reports etc.) from the Shapes generated by the SPUs

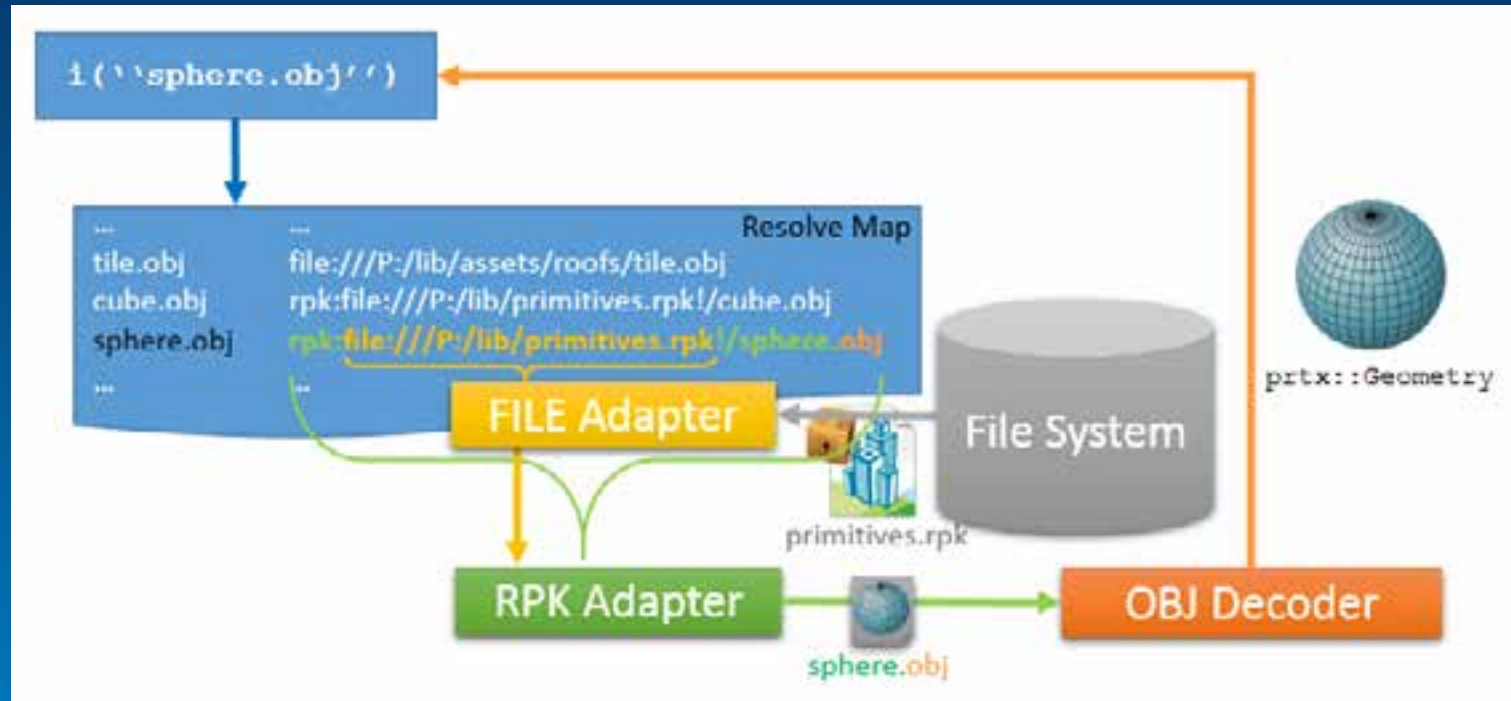


Decoders

- Used to decode external resources such as images (textures) and geometry

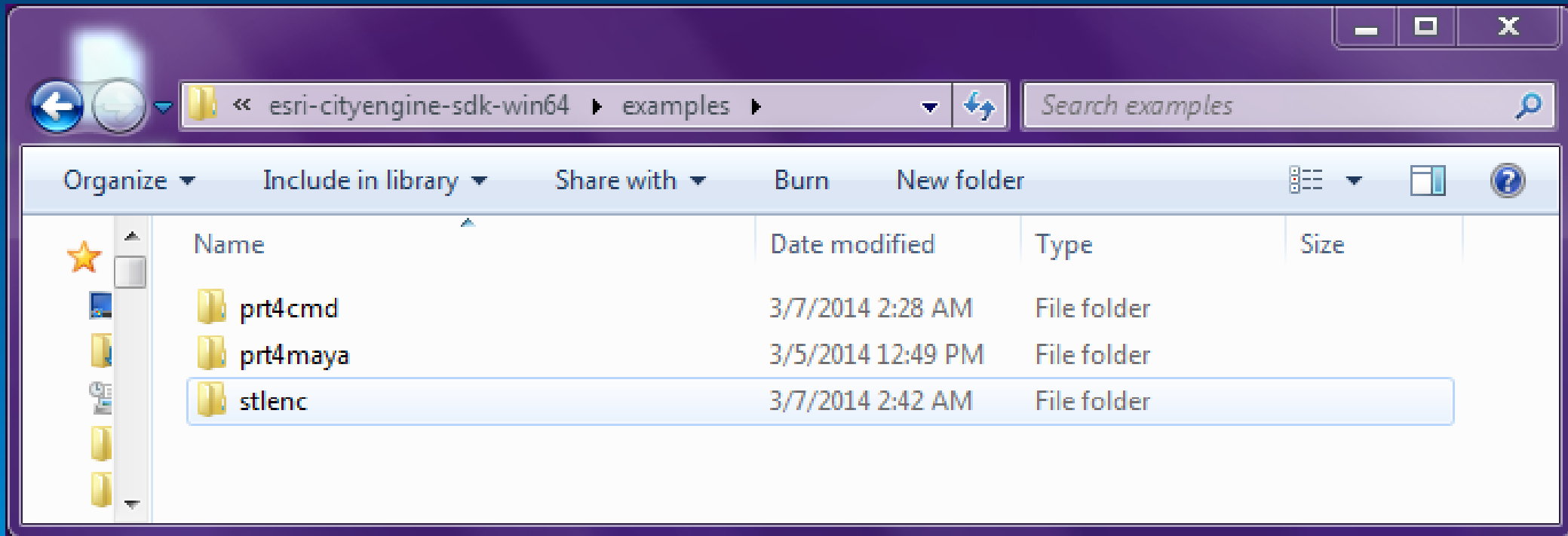


Adaptors & URIs



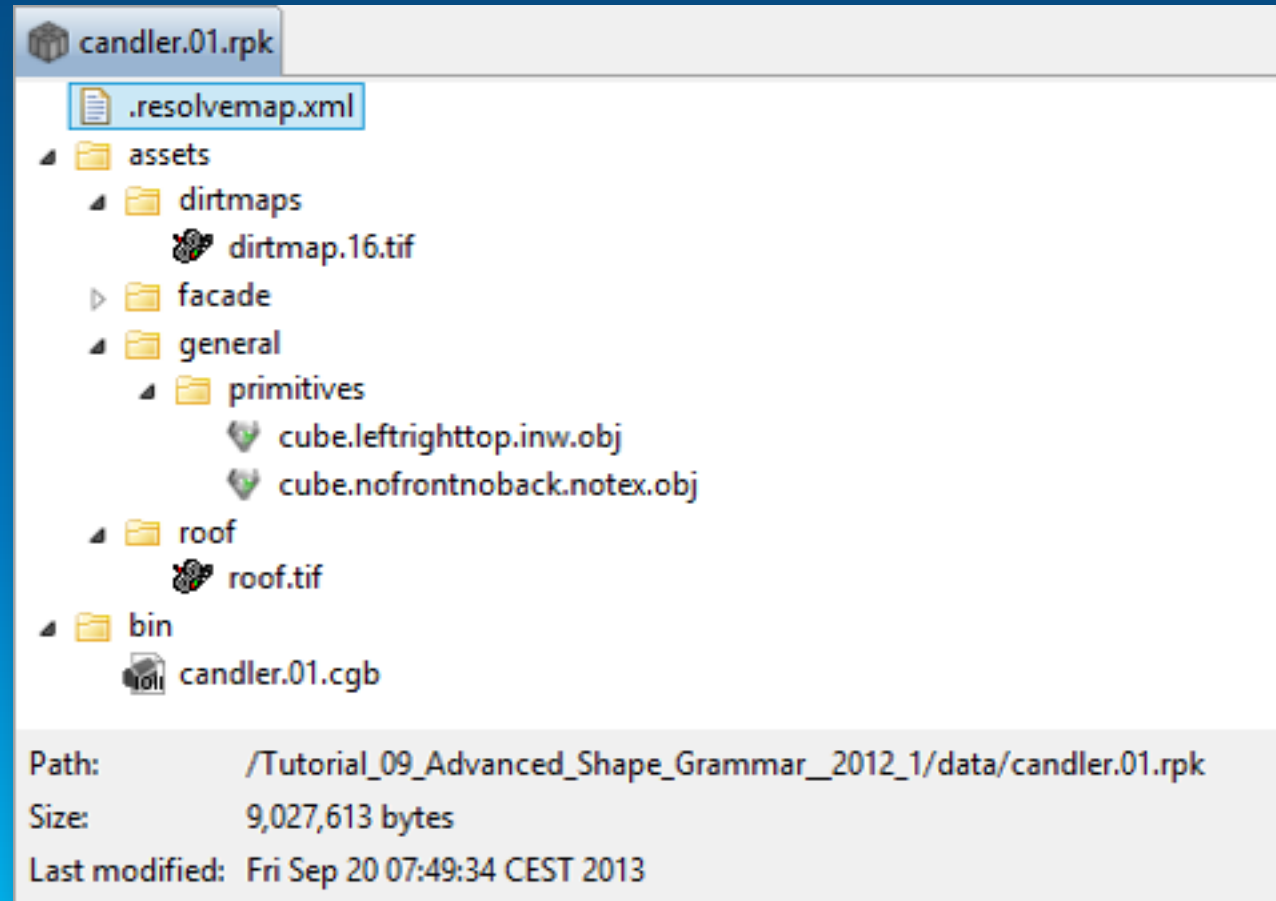
- Supported schemes (extensible):
 - file, zip, rpk, memory

CityEngine SDK Examples



Example 1: Hello World

- Simple command-line tool to take an initial shape and apply a rule package (RPK)
 - RPK = CGB file + assets (textures, geometry)



Example 1: Init

```
...
prt::ConsoleLogHandler* logHandler =
prt::ConsoleLogHandler::create(prt::LogLevel::ALL, (size_t)6);
prt::addLogHandler(logHandler);
...
// -- setup the licensing information
prt::FlexLicParams flp;
flp.mActLibPath = flexLib.c_str();
flp.mFeature = inputArgs.mLicFeature.c_str();
flp.mHostName = inputArgs.mLicHost.c_str();

// -- initialize PRT with the path to its extension libraries, the desired
// log level and the licensing data
const prt::Object* licHandle = prt::init(&cExtPath, 1,
                                         (prt::LogLevel)inputArgs.mLogLevel, &flp);
...
```

Example 1: Callbacks, Cache, ResolveMap

```
...
prt::FileOutputCallbacks* foc =
    prt::FileOutputCallbacks::create(inputArgs.mOutputPath.c_str());
prt::CacheObject* cache =
    prt::CacheObject::create(prt::CacheObject::CACHE_TYPE_DEFAULT);
...

const prt::ResolveMap* resolveMap =
    prt::createResolveMap(rpURI.c_str(), false, &status);
```

Example 1: InitialShapeBuilder

```
...
// -- setup initial shape
prt::InitialShapeBuilder* isb = prt::InitialShapeBuilder::create();
...
isb->resolveGeometry(inputArgs.mInitialShapeGeo.c_str(), resolveMap, cache);
...
isb->setAttributes(ruleFile.c_str(), startRule.c_str(), seed,
    shapeName.c_str(), inputArgs.mInitialShapeAttrs, resolveMap);
...
// -- create initial shape
const prt::InitialShape* initialShape = isb->createInitialShapeAndReset();
isb->destroy();
...
```


Example 1: Encoders & generate() call

```
const prt::AttributeMap* encoderOpts[] = {
    validatedEncOpts, validatedErrOpts, validatedPrintOpts };
const wchar_t* encoders[] = {
    inputArgs.mEncoderID.c_str(), // our desired encoder
    ENCODER_ID_CGA_ERROR, // an encoder to redirect rule errors into CGAErrors.txt
    ENCODER_ID_CGA_PRINT // an encoder to redirect CGA print statements to CGAPrint.txt
};

prt::Status stat = prt::generate(&initialShape, 1, 0,
                                encoders, 3, encoderOpts, foc, cache, 0);
if(stat != prt::STATUS_OK)
    std::cerr << "prt::generate() failed with status: '"
        << prt::getStatusDescription(stat) << "' (" << stat << ")'" << std::endl;
```

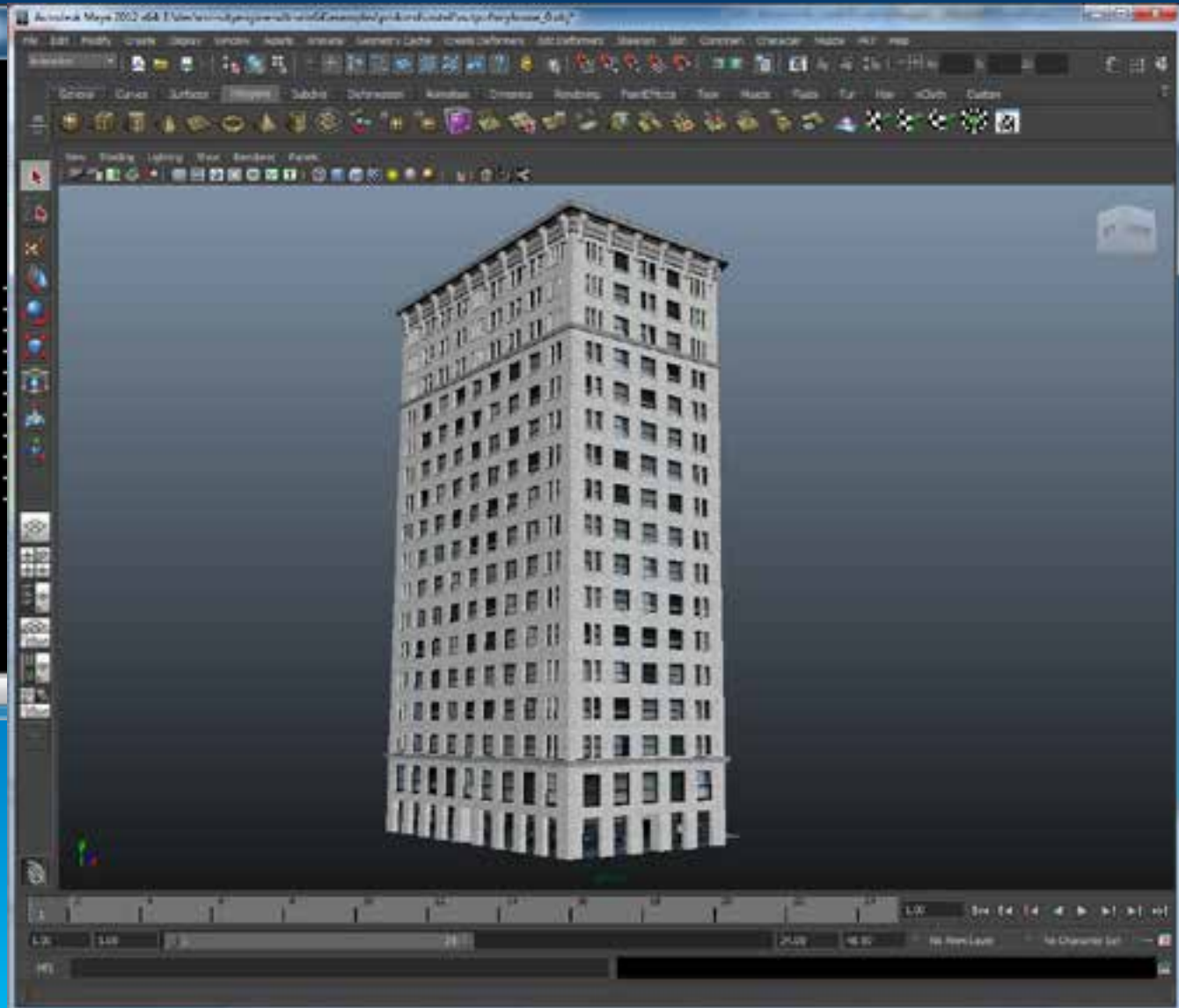
Example 1: Cleanup

```
...  
val i datedEncOpts->destroy();  
val i datedErrOpts->destroy();  
i ni ti al Shape->destroy();  
resol veMap->destroy();  
  
foc->destroy();  
cache->destroy();  
  
// release prt license and shutdown  
li cHandl e->destroy();  
  
// -- remove loggers  
prt::removeLogHandl er(logHandl er);  
logHandl er->destroy();
```

Example 1: Compile & Run

```
Visual Studio x64 Win64 Command Prompt (2010)
E:\dec\esri-cityengine-sdk-win64\examples\prt4cmd>cd build
E:\dec\esri-cityengine-sdk-win64\examples\prt4cmd\build>nnake install
Microsoft (R) Program Maintenance Utility Version 10.00.40219.01
Copyright (C) Microsoft Corporation. All rights reserved.

[100%] Built target prt4cmd
Install the project...
-- Install configuration: "Release"
-- Installing: E:/dec/esri-cityengine-sdk-win64/examples/prt4cmd/src/
-- Installing: E:/dec/esri-cityengine-sdk-win64/examples/prt4cmd/src/
-- Installing: E:/dec/esri-cityengine-sdk-win64/examples/prt4cmd/src/
-- Installing: E:/dec/esri-cityengine-sdk-win64/examples/prt4cmd/src/
-- Installing: E:/dec/esri-cityengine-sdk-win64/examples/prt4cmd/src/
-- Installing: E:/dec/esri-cityengine-sdk-win64/examples/prt4cmd/src/
-- Installing: E:/dec/esri-cityengine-sdk-win64/examples/prt4cmd/src/
-- Installing: E:/dec/esri-cityengine-sdk-win64/examples/prt4cmd/src/
-- Installing: E:/dec/esri-cityengine-sdk-win64/examples/prt4cmd/src/
-- Installing: E:/dec/esri-cityengine-sdk-win64/examples/prt4cmd/src/
-- Installing: E:/dec/esri-cityengine-sdk-win64/examples/prt4cmd/src/
E:\dec\esri-cityengine-sdk-win64\examples\prt4cmd\build>
```



Example 2: Adding a new Export Format to CityEngine

Implement abstract class:

```
class STLEncoder : public prtx::GeometryEncoder {
public:

STLEncoder(const std::wstring& id, const prtx::AttributeMap* defaultOptions,
           prtx::Callbacks* callbacks);
virtual ~STLEncoder();

virtual void init(prtx::GenerateContext& context);
virtual void encode(prtx::GenerateContext& context, size_t initialShapeIndex);
virtual void finish(prtx::GenerateContext& context);

private:
    prtx::DefaultNamePreparator mNamePreparator;
    prtx::EncodePreparatorPtr mEncodePreparator;
};
```

Example 2: Adding a new Export Format to CityEngine

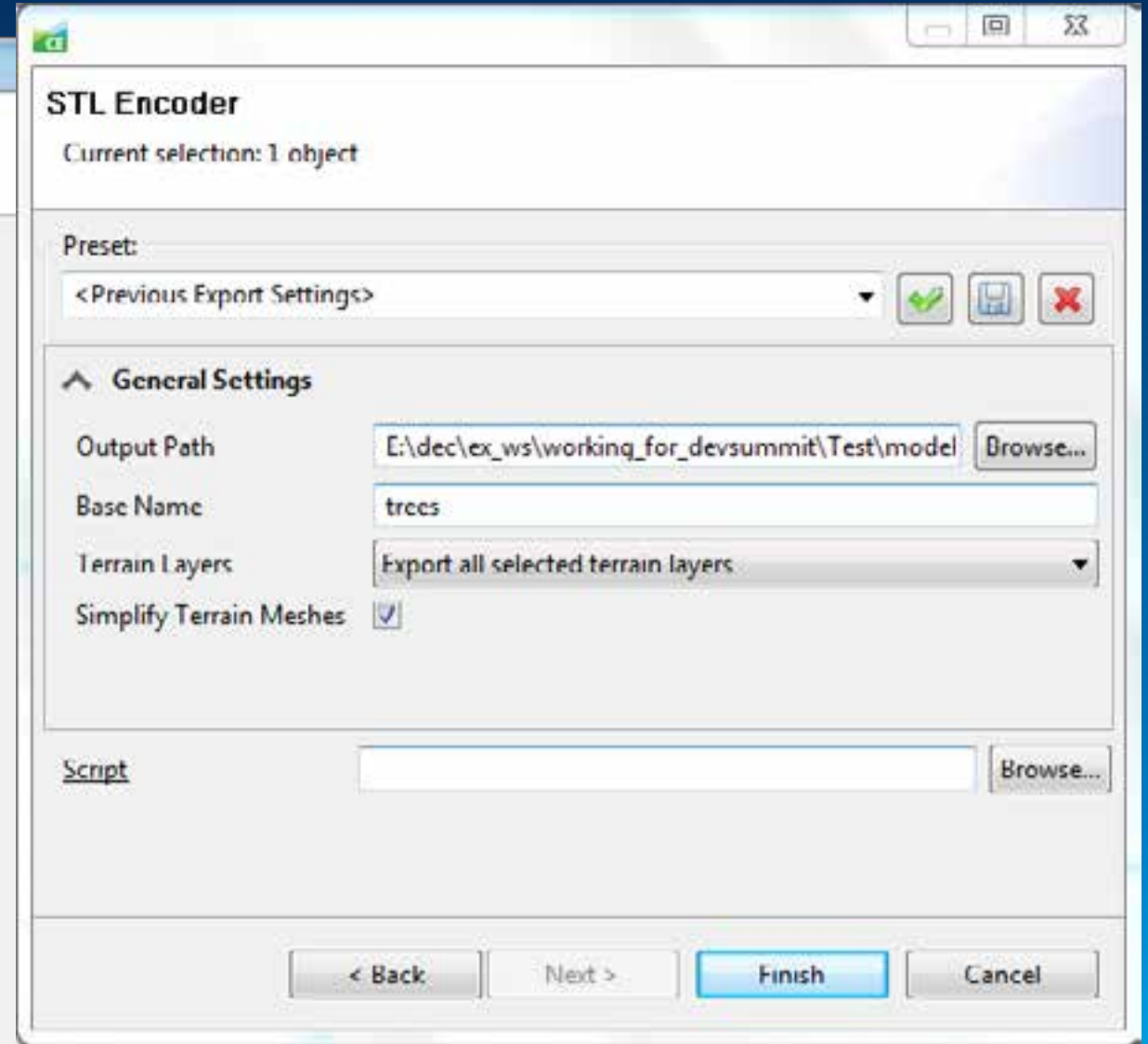
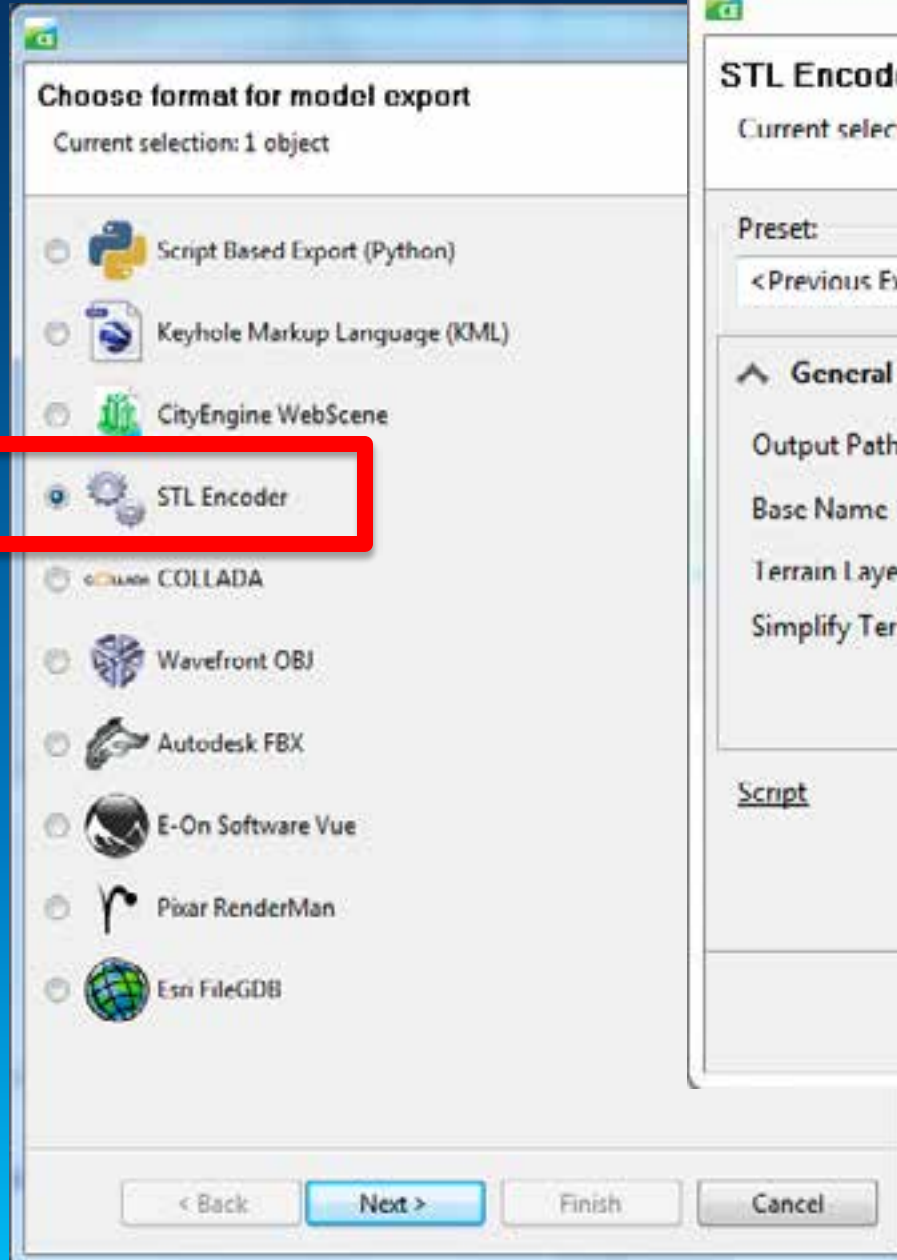
encode() iterates over shape tree and extracts geometry:

```
void STLEncoder::encode(prtx::GenerateContext& context, size_t initialShapeIndex) {
    const prtx::InitialShape* is = context.getInitialShape(initialShapeIndex);
    try {
        prtx::LeafIteratorPtr li =
            prtx::LeafIterator::create(context, initialShapeIndex);
        for(prtx::ShapePtr shape = li->getNext(); shape.get() != 0;
            shape = li->getNext())
        {
            mEncodePreparator->add(context.getCache(), shape, is->getAttributeMap());
        }
    } catch(...) {
        mEncodePreparator->add(context.getCache(), *is, initialShapeIndex);
    }
}
```

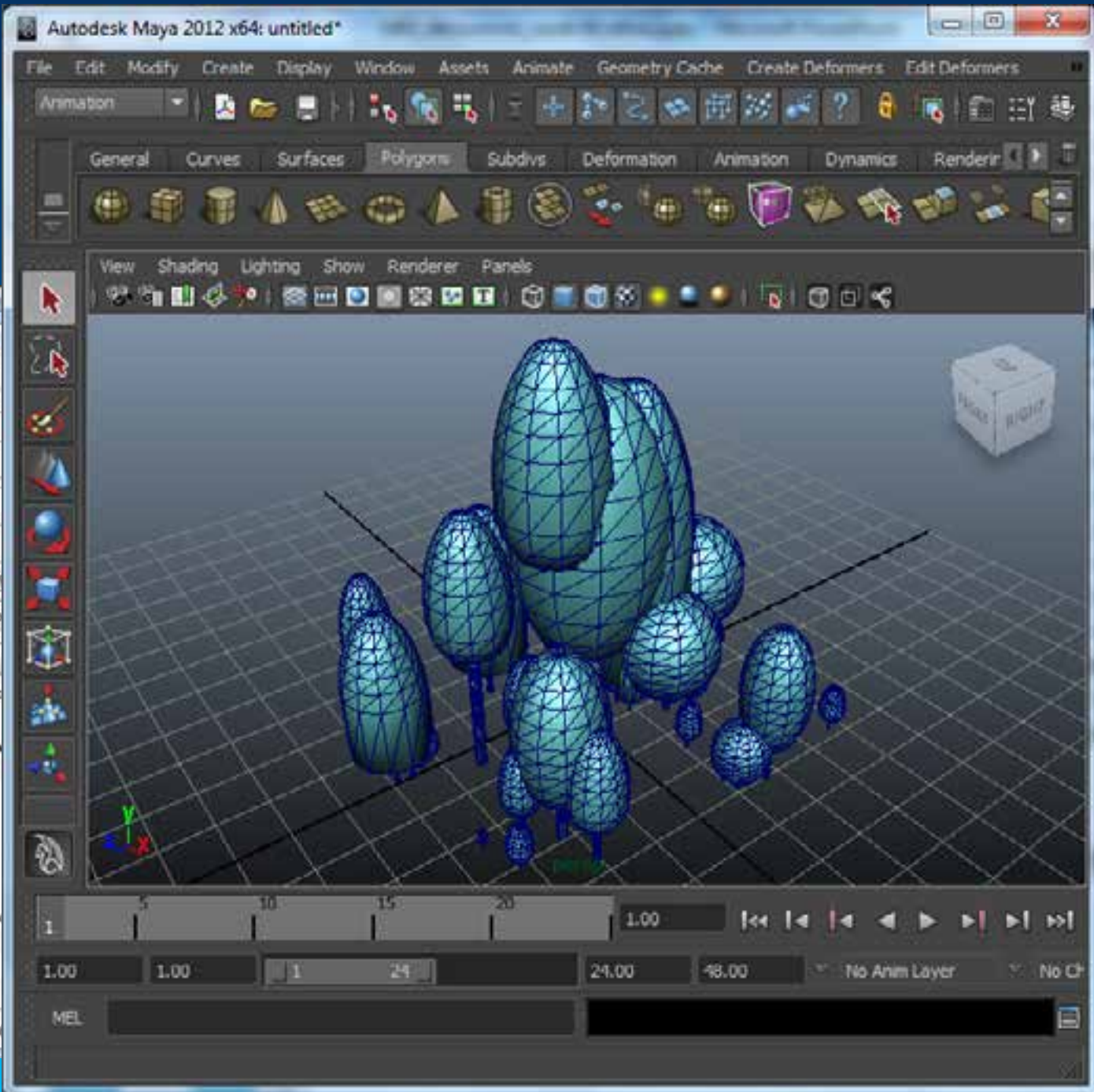
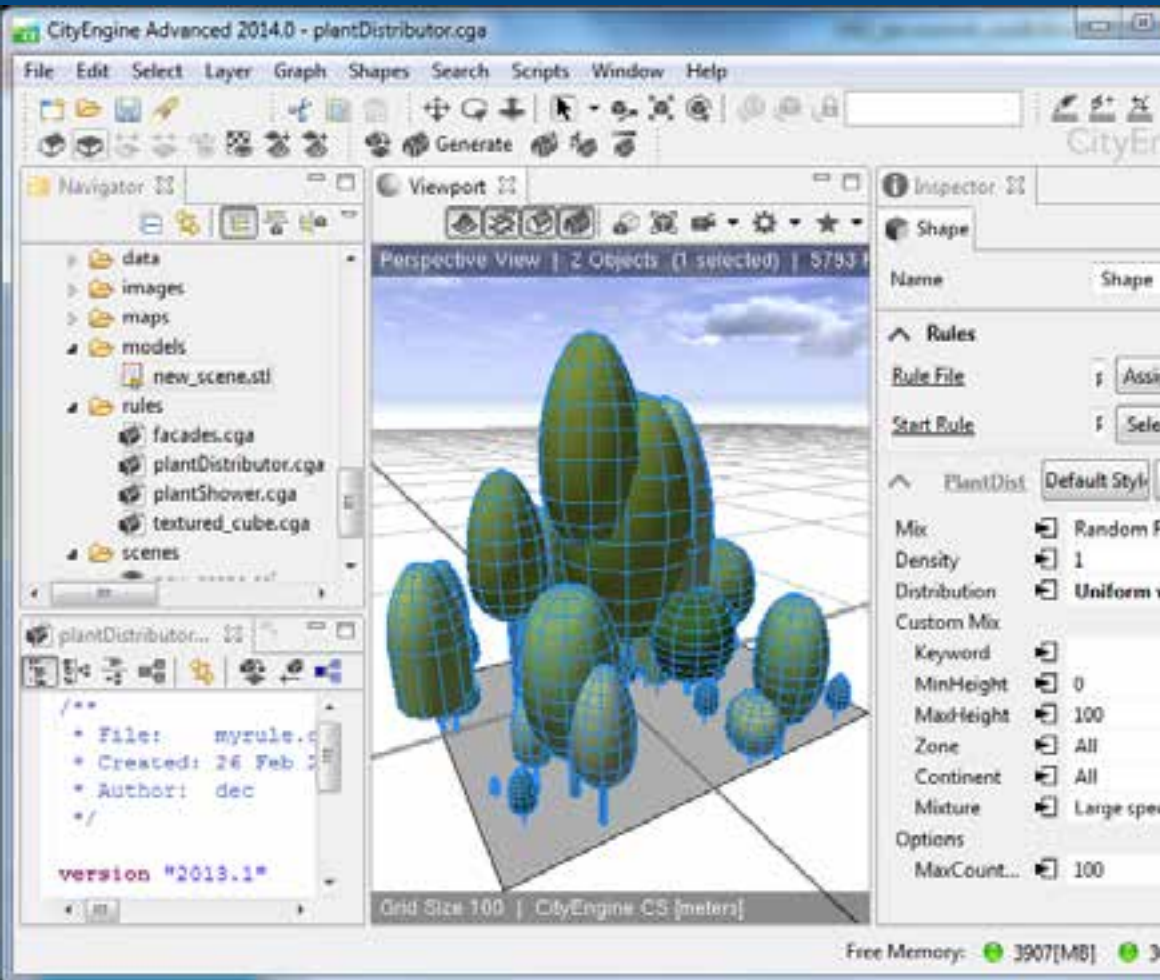
Example 2: finish() writes to Callbacks

```
void STLEncoder::finish(prtx::GenerateContext& /*context*/) {
    prt::SimpleOutputCallbacks* soc =
        dynamic_cast<prt::SimpleOutputCallbacks*>(getCallbacks());
    ...
    std::wostream out;
    ...
    for(uint32_t fi = 0, n = m->getFaceCount(); fi < n; fi++) {
        ...
        out << L"facet normal " << fn[0] << L" " << fn[1] << L" " << fn[2] << WNL;
        ...
        // let the client application write the file via callback
        std::wstring fileName = baseName + STL_EXT;
        uint64_t h = soc->open(ID.c_str(), prt::CT_GEOMETRY, fileName.c_str(),
            prt::SimpleOutputCallbacks::SE_UTF8);
        soc->write(h, out.str().c_str());
        soc->close(h, 0, 0);
    }
}
```

Example 2

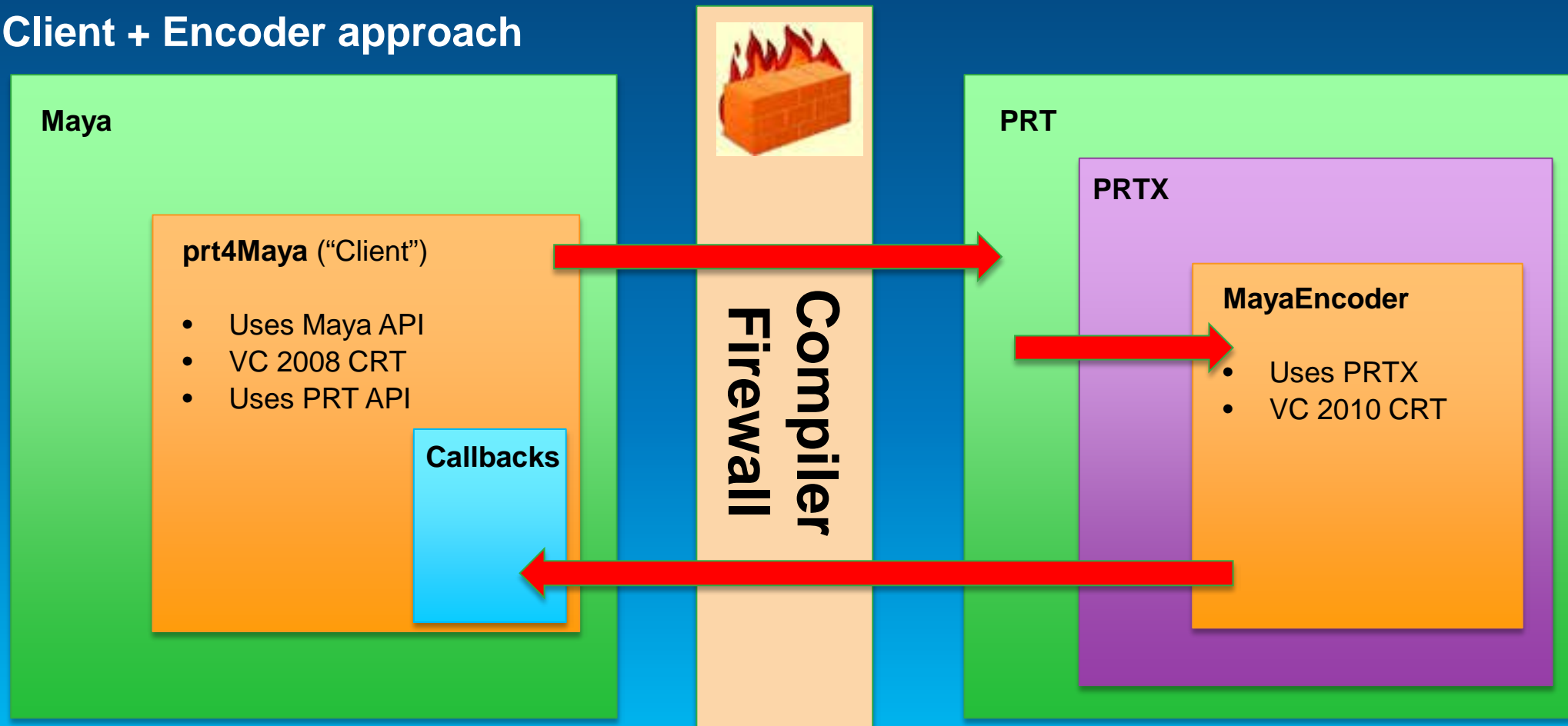


Example 2



Example 3: DCC Plugin

- “Digital Content Creation Tool”, e.g. Autodesk Maya
- Client + Encoder approach



Example 3: Client-specific Callbacks

```
#include "prt/Callbacks.h"

class I MayaCallbacks : public prt::Callbacks {
public:
    virtual ~IMayaCallbacks() { }

    virtual void setVertices(double* vtx, size_t size) = 0;
    virtual void setNormals(double* nrm, size_t size) = 0;
    virtual void setUVs(float* u, float* v, size_t size) = 0;

    virtual void setFaces(int* counts, size_t countsSize, int* connects, size_t connectsSize,
        int* uvCounts, size_t uvCountsSize, int* uvConnects, size_t uvConnectsSize) = 0;
    virtual void createMesh() = 0;
    virtual void finishMesh() = 0;

    virtual void matSetColor(int start, int count, float r, float g, float b) = 0;
    virtual void matSetDiffuseTexture(int start, int count, const wchar_t* tex) = 0;
};
```

Example 3: Client Gets Called by Maya

```
MStatus PRTNode::compute(const MPlug& plug, MDataBlock& data ) {  
    MStatus stat;  
    ...  
    prt::Status generateStatus = prt::generate(&shape, 1, 0, &ENC_MAYA, 1,  
                                               &mMayaEncOpts, outputHandler, PRTNode::theCache, 0);  
    ...  
    return MS::kSuccess;  
}
```

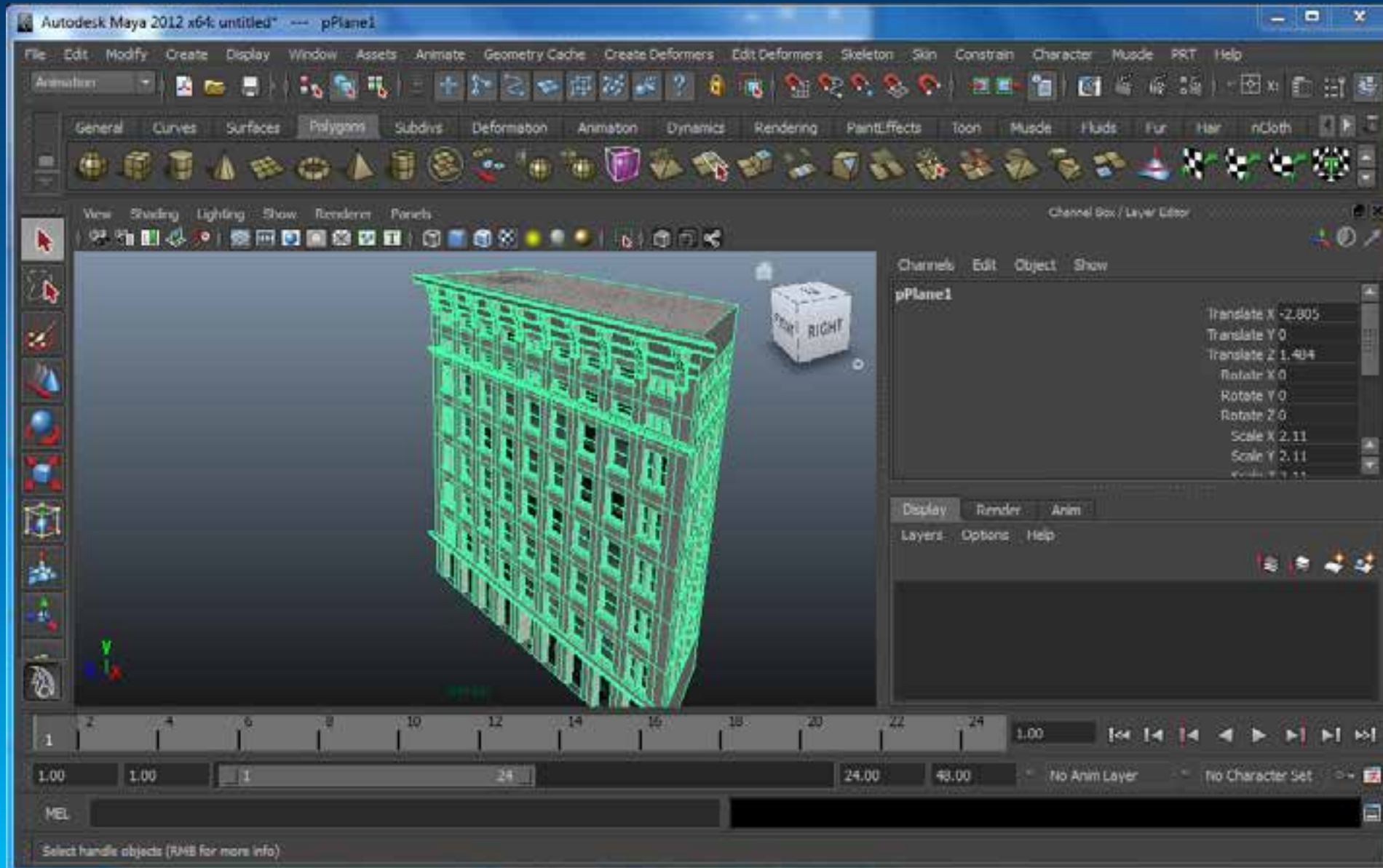
Example 3: Encoder extracts Geometry

```
void MayaEncoder::encode(prtx::GenerateContext& context, size_t initialShapeIndex) {  
    ...  
    prtx::EncodePreparatorPtr encPrep =  
        prtx::EncodePreparator::create(true, namePrep, nsMesh, nsMaterial);  
  
    prtx::LeafIteratorPtr li =  
        prtx::LeafIterator::create(context, initialShapeIndex);  
    for (prtx::ShapePtr shape = li->getNext(); shape != 0; shape = li->getNext())  
        encPrep->add(context.getCache(), shape);  
  
    ...  
    mayaCallbacks->setVertices(&vertices[0], vertices.size());  
    ...  
    mayaOutput->createMesh();  
    ...  
}
```

Example 3: Callbacks sets up Mesh in Maya

```
void MayaCallbacks::createMesh() {  
    MStatus stat;  
  
    ...  
    mFnMesh = new MFnMesh();  
    MObject oMesh = mFnMesh->create(mVertices.length(),  
        mVerticesCounts.length(), mVertices, mVerticesCounts,  
        mVerticesConnects, newOutputData, &stat);  
    MCHECK(stat);  
  
    ...  
}
```

Example 3: Maya Plugin



CityEngine SDK availability and licensing

- SDK is part of CityEngine (2013 onwards)
- Requires a CityEngine license to run (basic or advanced)
- Middleware licensing will be on a case-by-case basis
- Documentation and examples (e.g. Maya plugin) on GitHub
<https://github.com/Esri/esri-cityengine-sdk>
- CityEngine Trial
 - [http:// www.esri.com/cityengine](http://www.esri.com/cityengine)



Understanding our world.