



Hacking Cities with Esri CityEngine

Markus Lipp

Esri Developer
Summit Europe

11-13 November 2013
Park Plaza Riverbank London





CityEngine

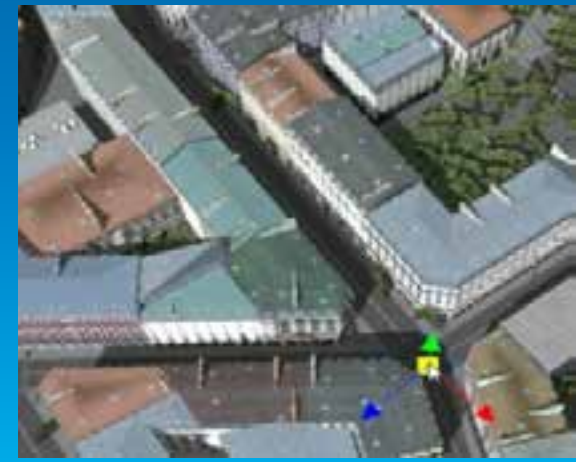
<http://www.esri.com/software/cityengine>

3D procedural modeling and design solution

- **Procedurally generate 3D urban content**
 - From 2D GIS geometry and attributes
 - Using algorithms and parametric rules



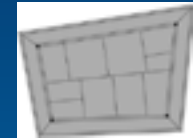
- **3D City Design**
 - Iterative design
 - Real-time feedback
 - Street sketching



Procedural modeling

3D model creation using rules / algorithms

- Base geometry



- Procedural rules



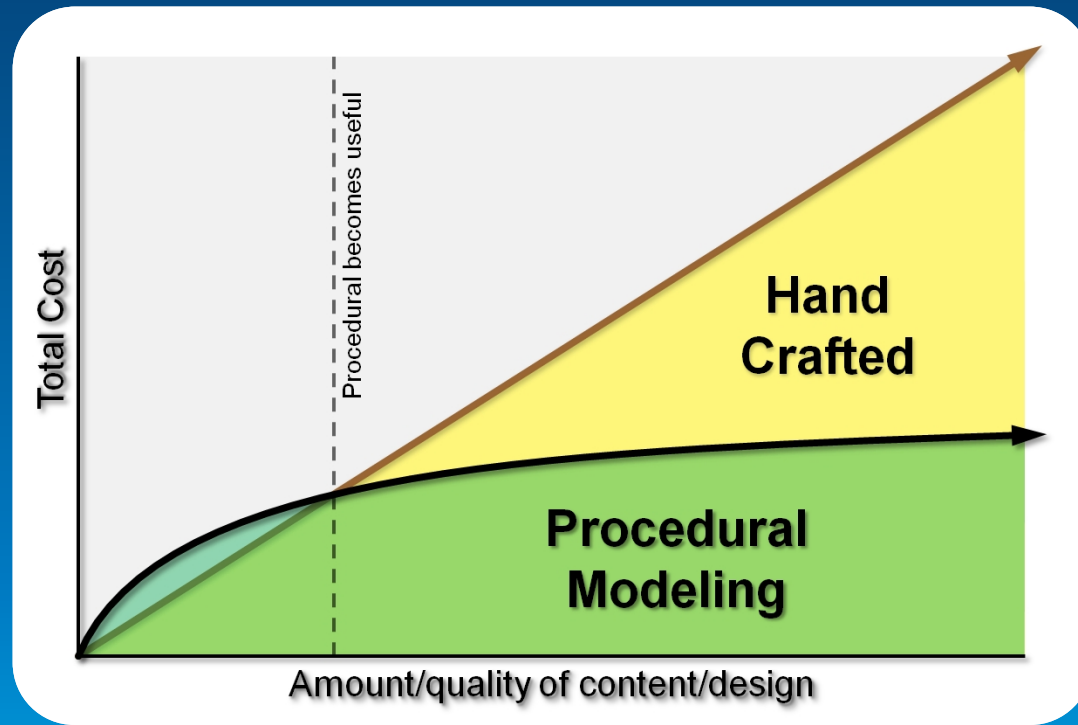
Base geometry



Final 3D model

Iteratively refine a design by creating more and more detail

Procedural Modeling vs. Manual Modeling



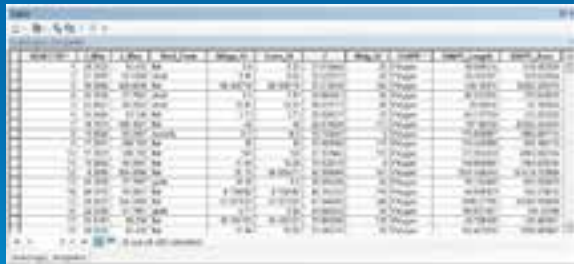
Time reduction / cost saving

GIS Data as Input

ArcGIS example



Geometry (parcels, footprints, streets)



ID	Parcel ID	Area	Perimeter	Height	Roof Type	Street Width	...
1	101	1500	1200	10	Flat	10	...
2	102	2000	1500	15	Gabled	15	...
3	103	3000	2000	20	Flat	20	...
4	104	4000	2500	25	Flat	25	...
5	105	5000	3000	30	Flat	30	...

Attributes (height, roof type, street width)

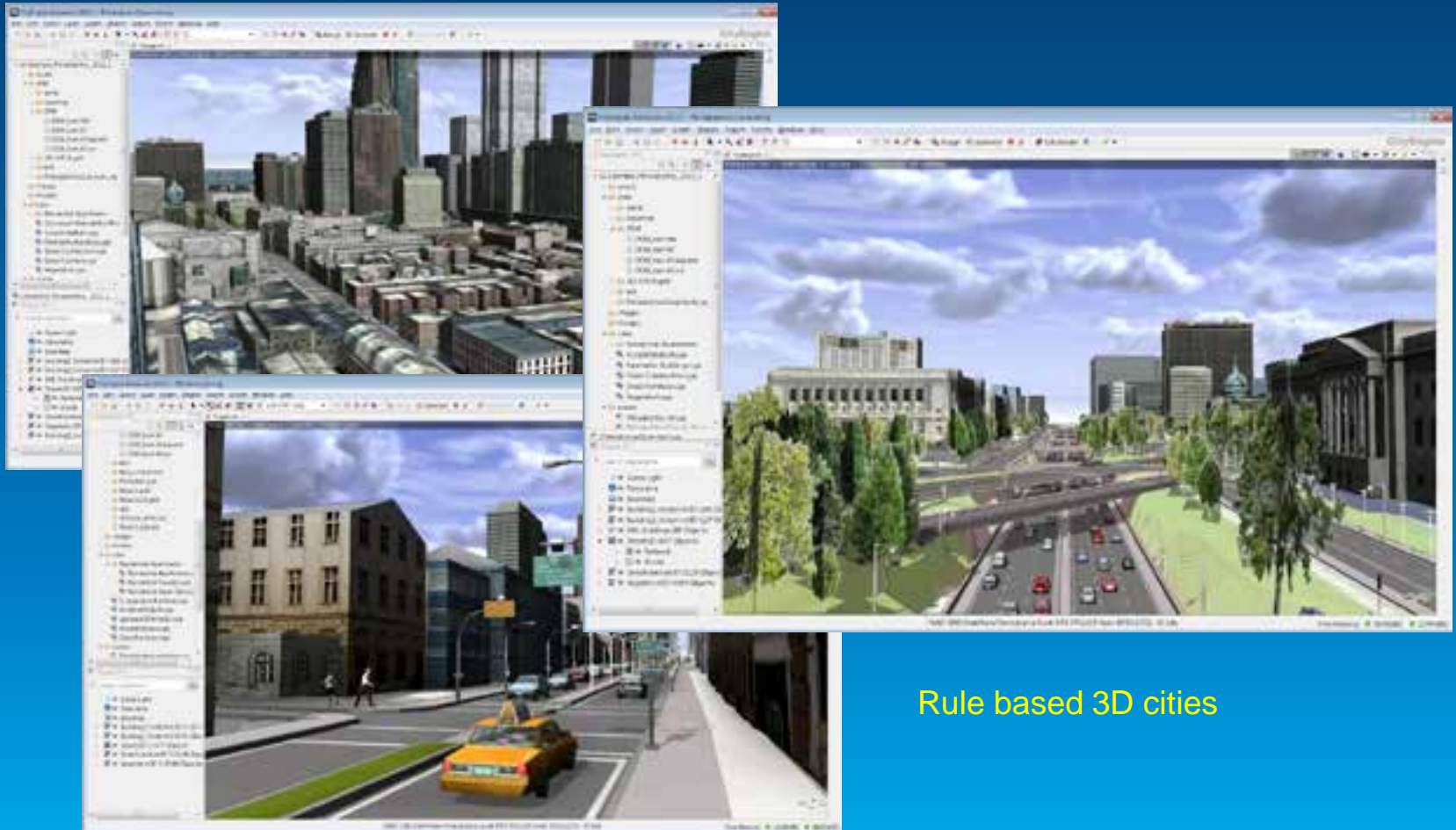


Rules



3D city content from GIS data

procedural city modeling



Rule based 3D cities

3D City Design – Procedural Approach

Urban planning example



Add a floor

Add a roof

New development – draw streets

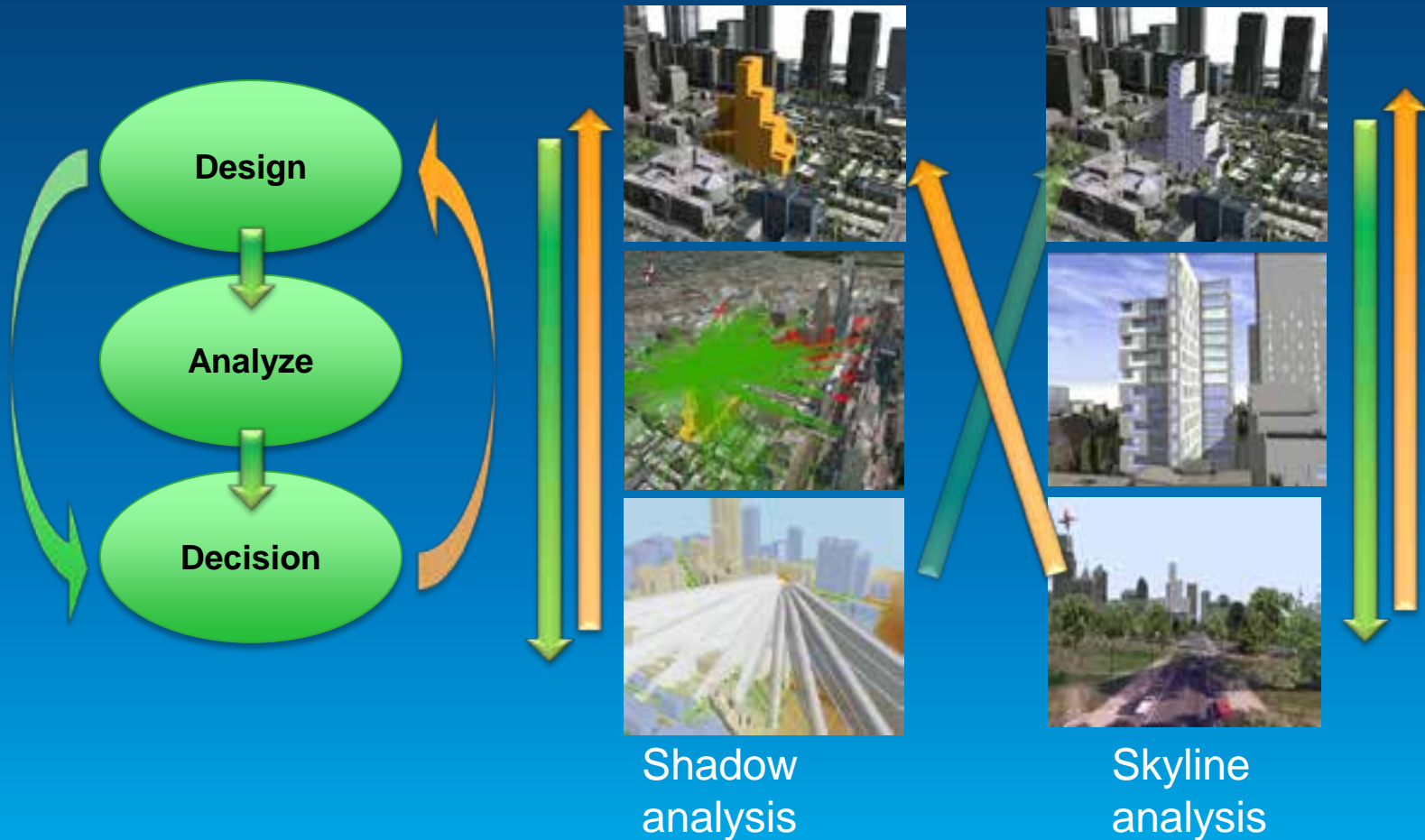


Reporting (area ratios...)

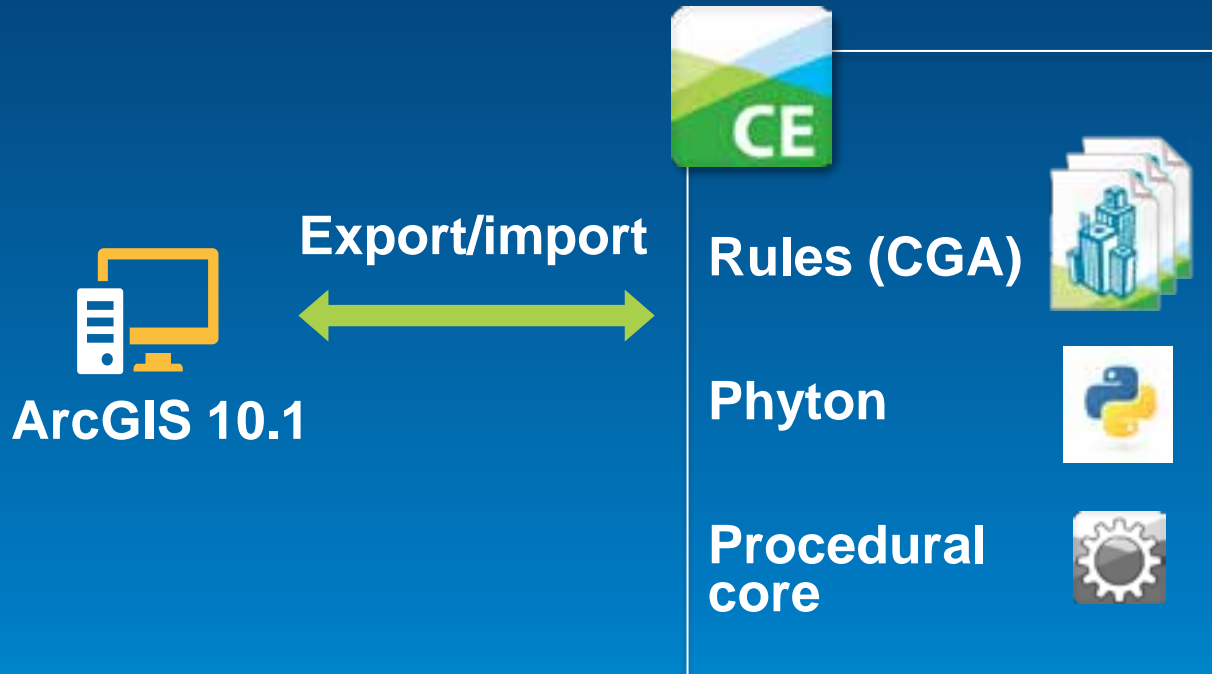


3D City (Geo)design

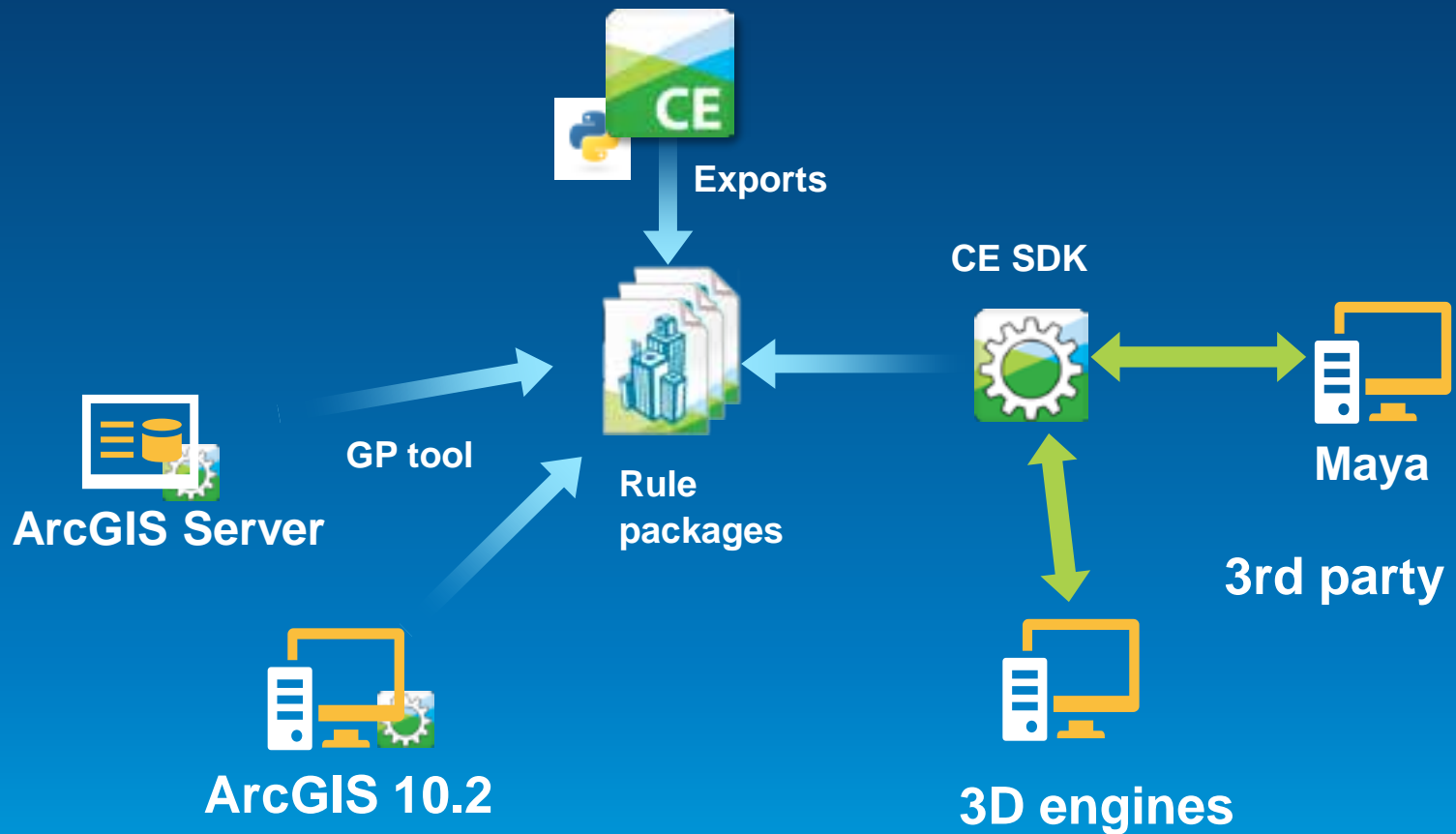
Iterative analysis while designing



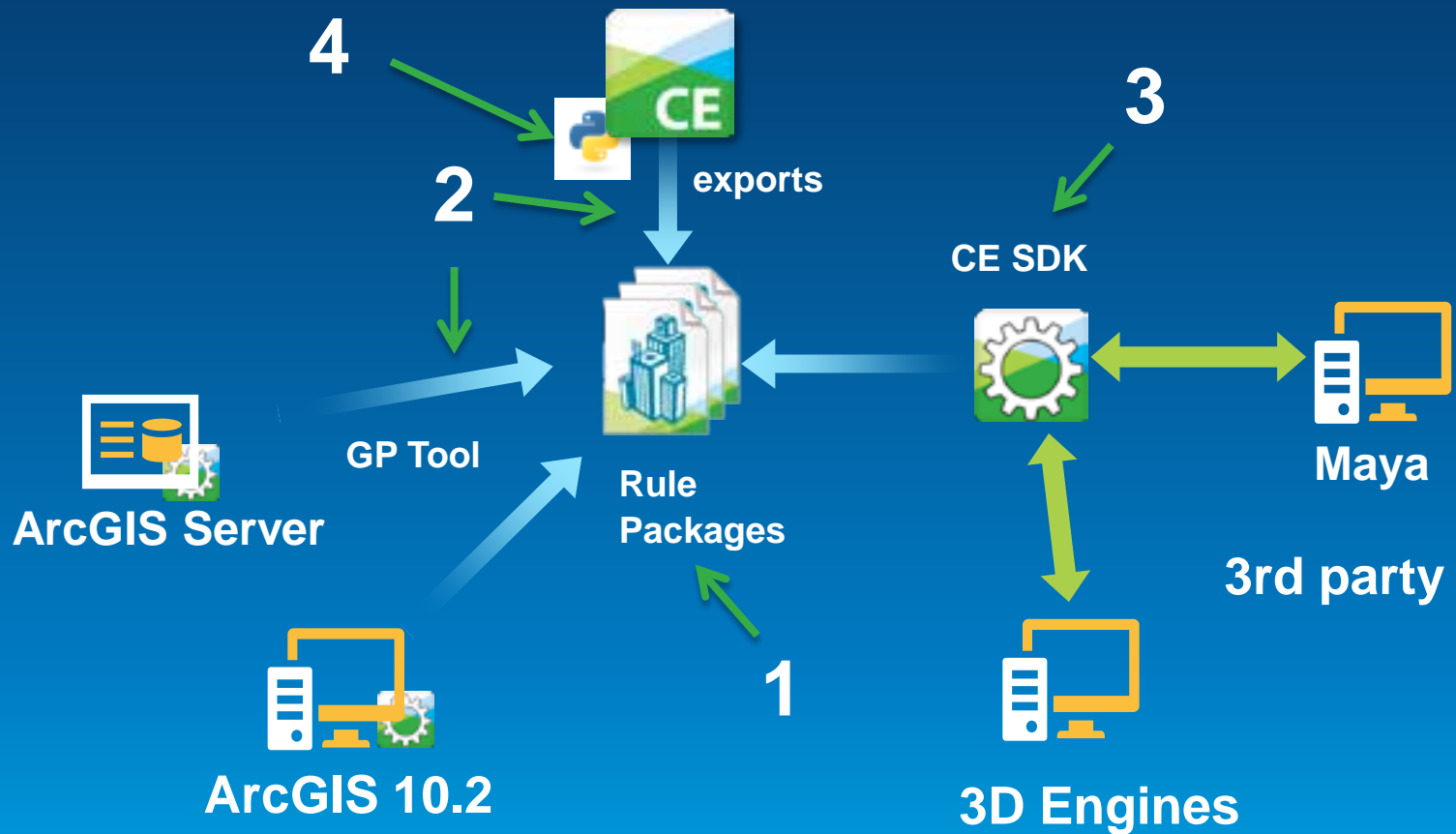
CityEngine 2012 – Opportunities for Developers



CityEngine 2013 for Developers



CityEngine 2013 for Developers



1. Rules, Rule Packages, CGA

- Rule: description of shape refinement



- Rule Package: multiple rules & assets

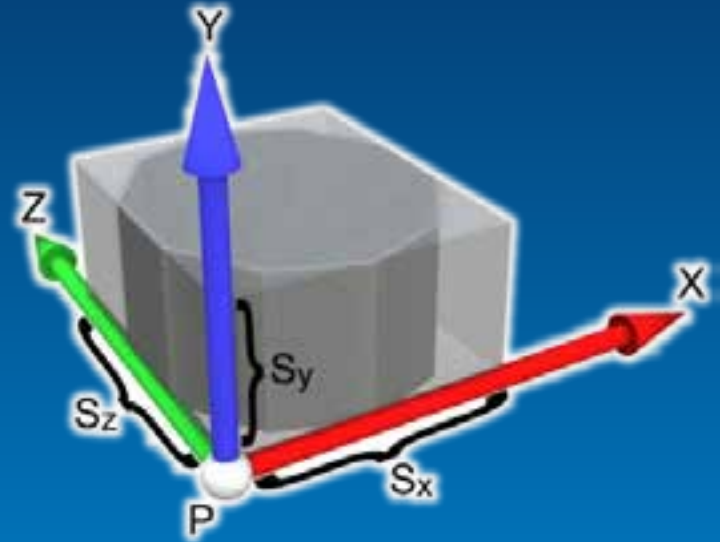


- CGA: «scripting language for shapes»

```
Mass (h, roofType) -->  
  extrude (h*HeightFactor) Stories comp (I) {top: Roof (roofType) }
```

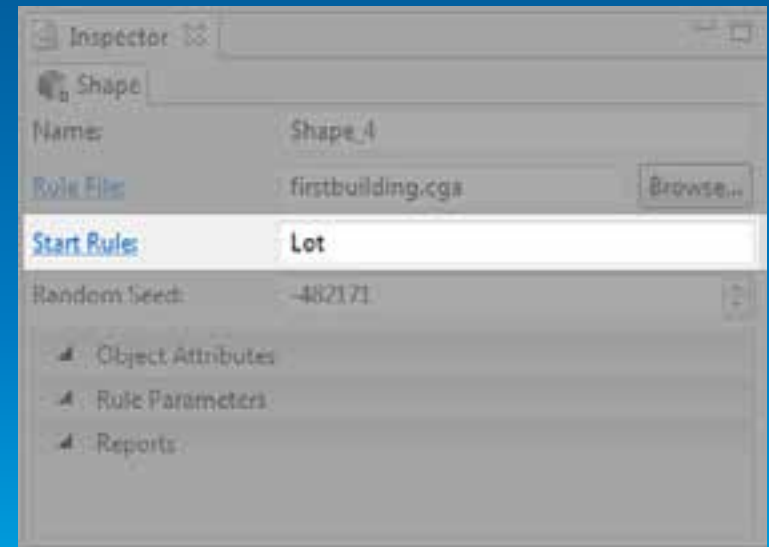
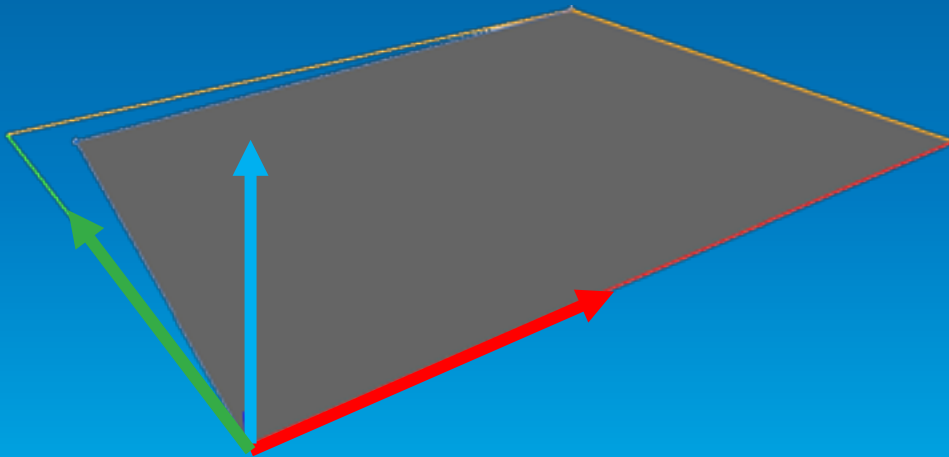

CGA Shape Grammar - Definition

- A *shape* consists of:
 - Symbol
 - Attributes
 - Geometry (polygonal mesh)
 - Oriented bounding box called *scope* (numeric attributes)
- Initial shape: *axiom*
- A *rule* describes the transformation of a shape into one or more successor shapes



GIS Lot as Initial Shape

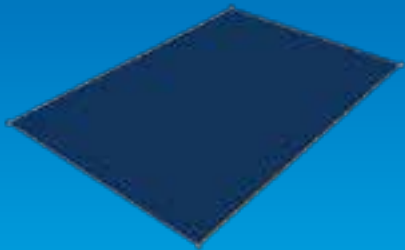
- **Symbol = start rule**
- **Attributes: height, zoning...**
- **Geometry = only one face**
- **Scope oriented on first edge**



Rule Example

```
Lot --> extrude(10) Mass
```

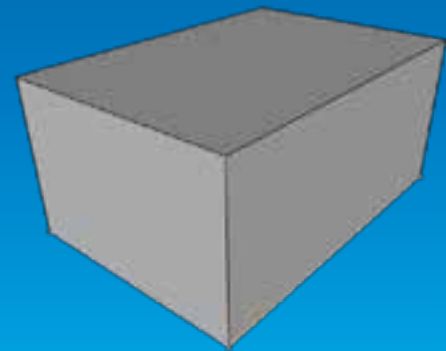
- *Lot* and *Mass* are shapes
- A modified copy of shape *Lot* becomes shape *Mass*
- *Mass* is called a leaf shape
- Output geometry = all leaf shapes



Lot with shape symbol *Lot*



Rule application (generation)



Resulting shape *Mass*
Displayed geometry

Multiple rules

<code>Lot --> extrude(10) Mass</code>	<i>Rule #1</i>
<code>Mass --> C D</code>	<i>Rule #2</i>

- Rule #2 is a matching rule for shape Mass
- Shape Mass is replaced by shapes C and D
- Mass NOT leaf shape here

CGA Syntax Example

```
attr height = 20
const heightG = 8.5
Lot --> extrude(height) Mass
Mass --> comp(f) { top : Roof.
  | front : Frontfacade
  | side : Facade }
Facade -->
  split(y) { heightG: Groundfloor
  / ~1 : UpperFloors }
Groundfloor -->
  case scope.sx > 10 :
  color("#cccccc")
  else : color("#ffcccc")
```

- Rules (may have parameters)
Lot, Mass, ...
- User-defined attributes and constants: height, heightG
- Boolean, float and string expressions
20, 8.5, ("#cccccc"),
scope.sx > 10
- CGA-specific keywords
attr, top, front, case
- CGA operations (may have parameters)
extrude(height),
comp(f)

CGA operations overview

Geometry creation



Geometry subdivision



Texturing



Transformations



User Interface in CityEngine

- Example building rule file



The screenshot displays the CityEngine interface with the following components:

- Rule File Editor:** Shows the following code:

```
13
14 attr windowWidth = 2.2
15 attr windowHeight = 2.8
16 attr floorheight = 4.0
17
18
19 Lot -->
20 extrude(height: comp(2) | side : Facade | top(2): Shape )
21
22 Facade -->
23 setupProjection(0, scope.xy, -2, -2)
24 split(y) ( $ : Door | -1: UpperFloors )
```
- 3D View:** Shows a 3D model of a building with a grid of windows, corresponding to the rule file output.
- Inspector Panel:** Shows the following parameters:
 - Shape:** Shape_2
 - Rule File:** solution/simple_building.cga
 - Start Rule:** Lot
 - Random Seed:** -93640
 - Object Attributes:** (collapsed)
 - Rule Parameters:**

Name	Source	Value
floorheight	Rule	4
height	Rule	18.9770...
tilewidth	Rule	3
windowHeight	Rule	2.800000
windowWidth	Rule	2.200000
 - Reports:** (collapsed)

2. Exporting and Using Rule Packages

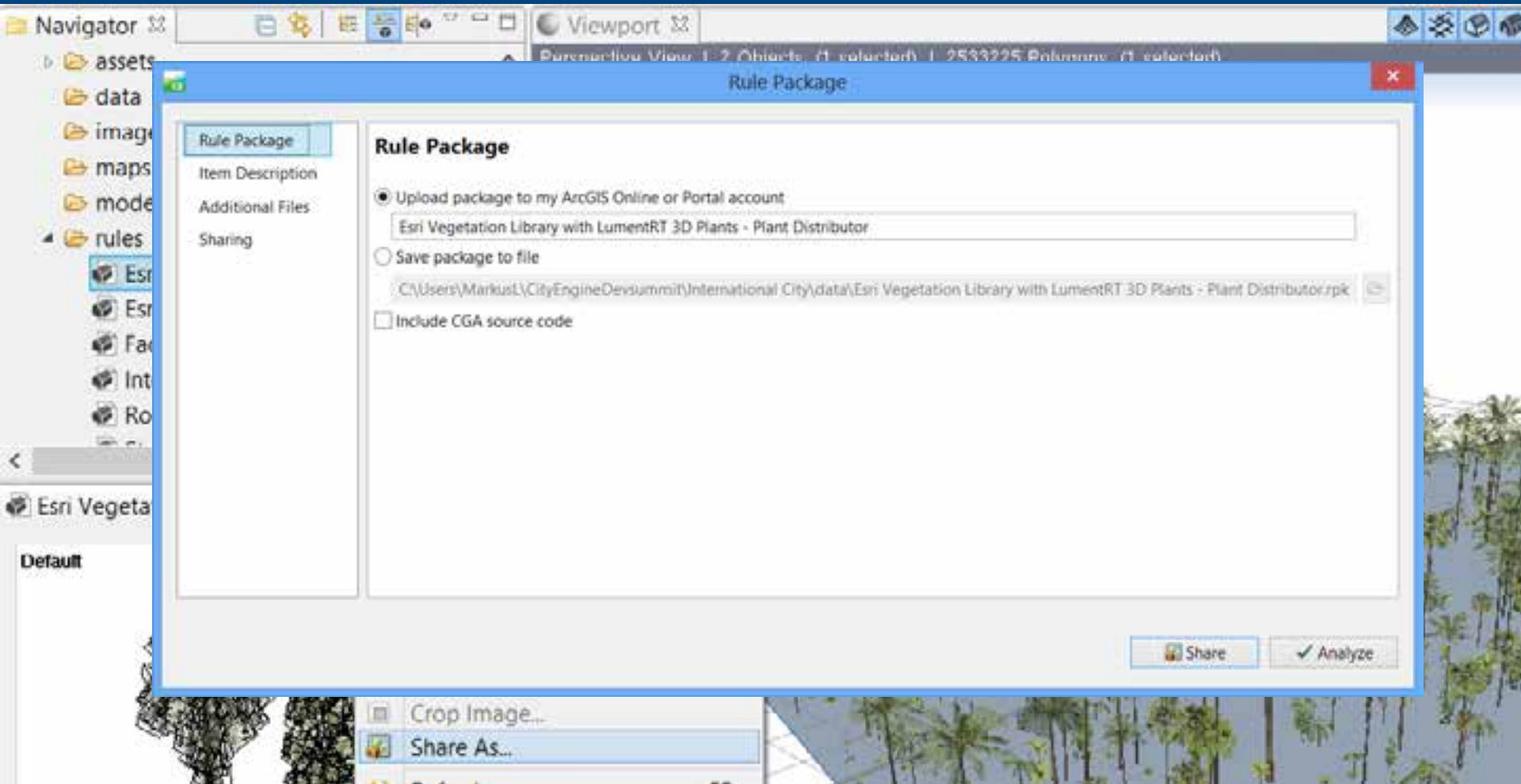
Recap: Rule package is:

- **Combination of CGA rules with assets**
 - Textures, meshes
- **Author in CityEngine, used in GP Tools or SDK**

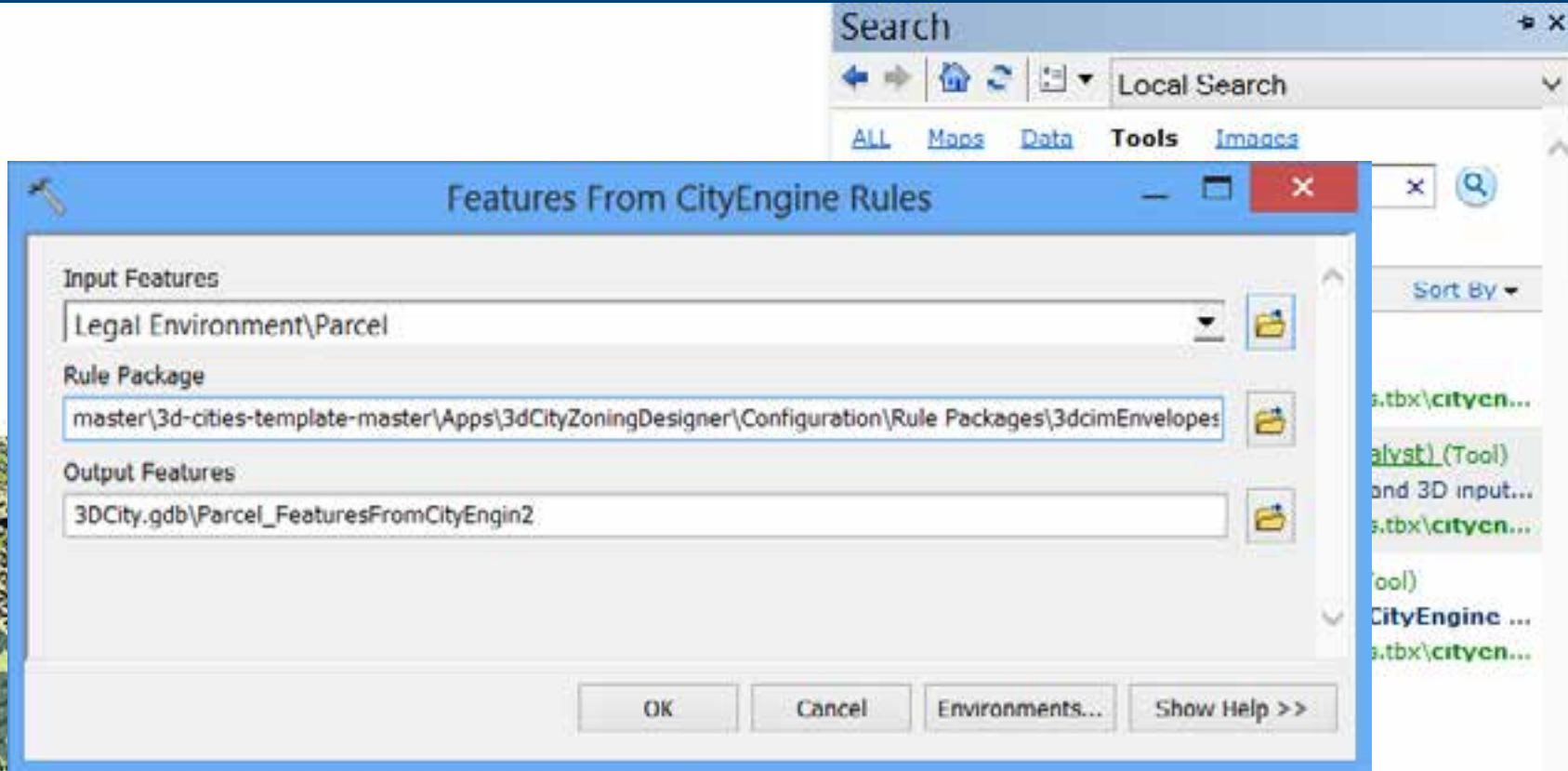


Export from CityEngine

right click on rule, “Share As...”

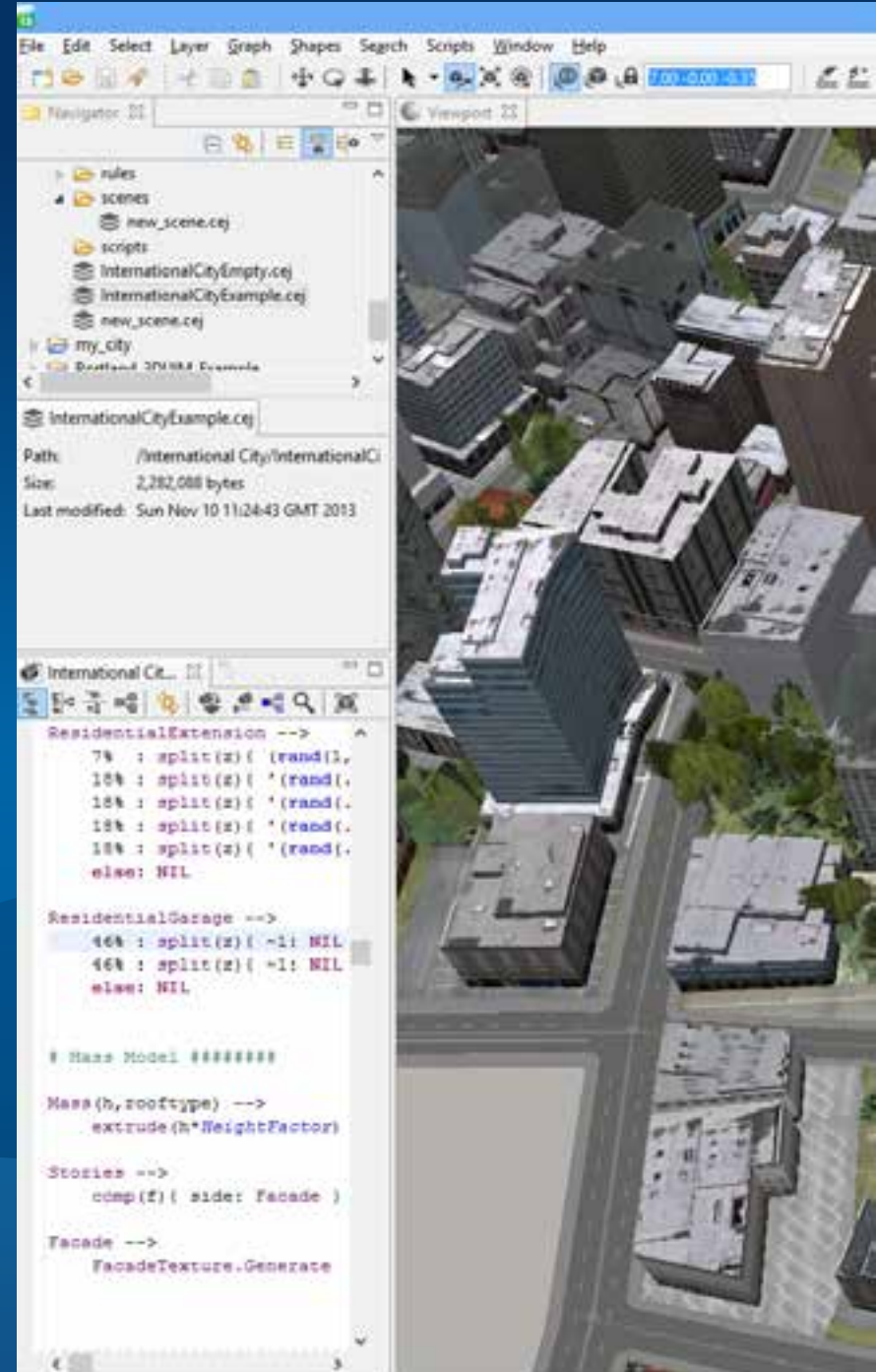


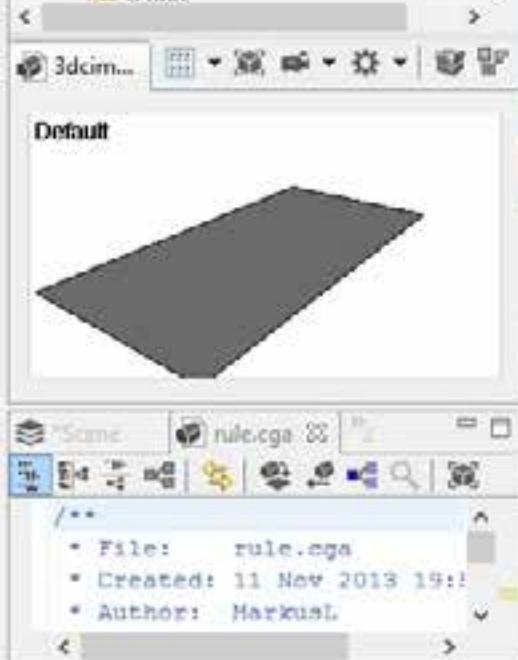
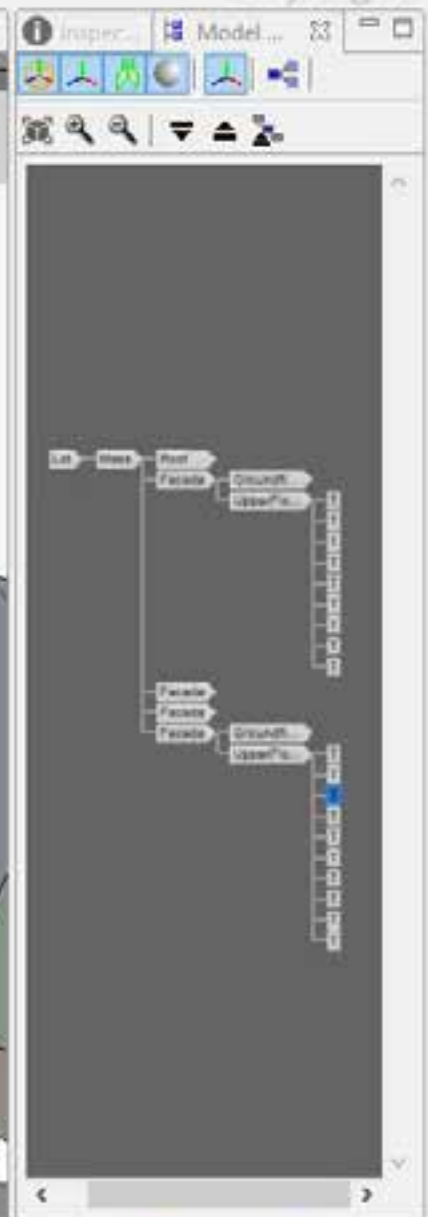
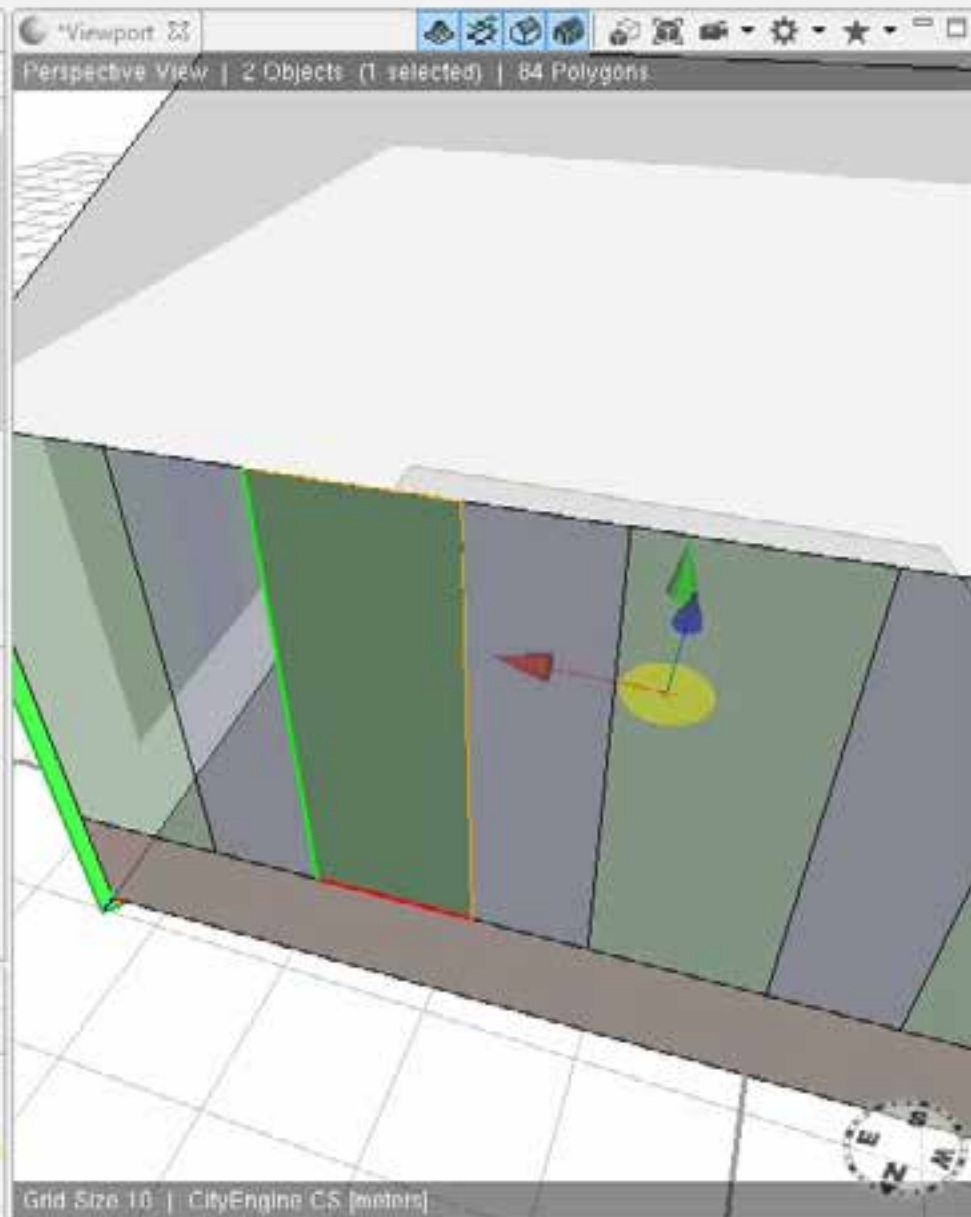
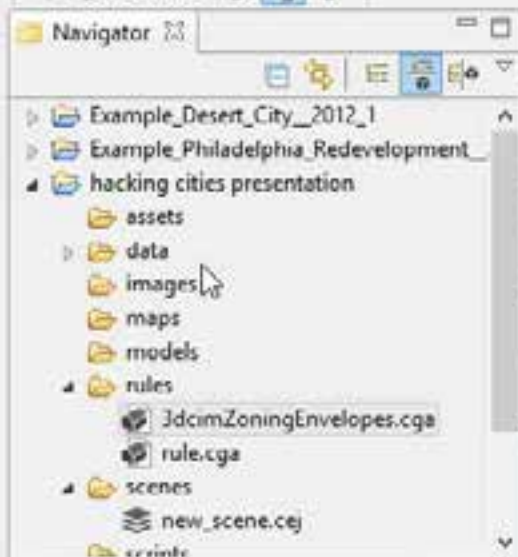
Using in ArcScene - CityEngine GP Tool



Demo

CityEngine 2013

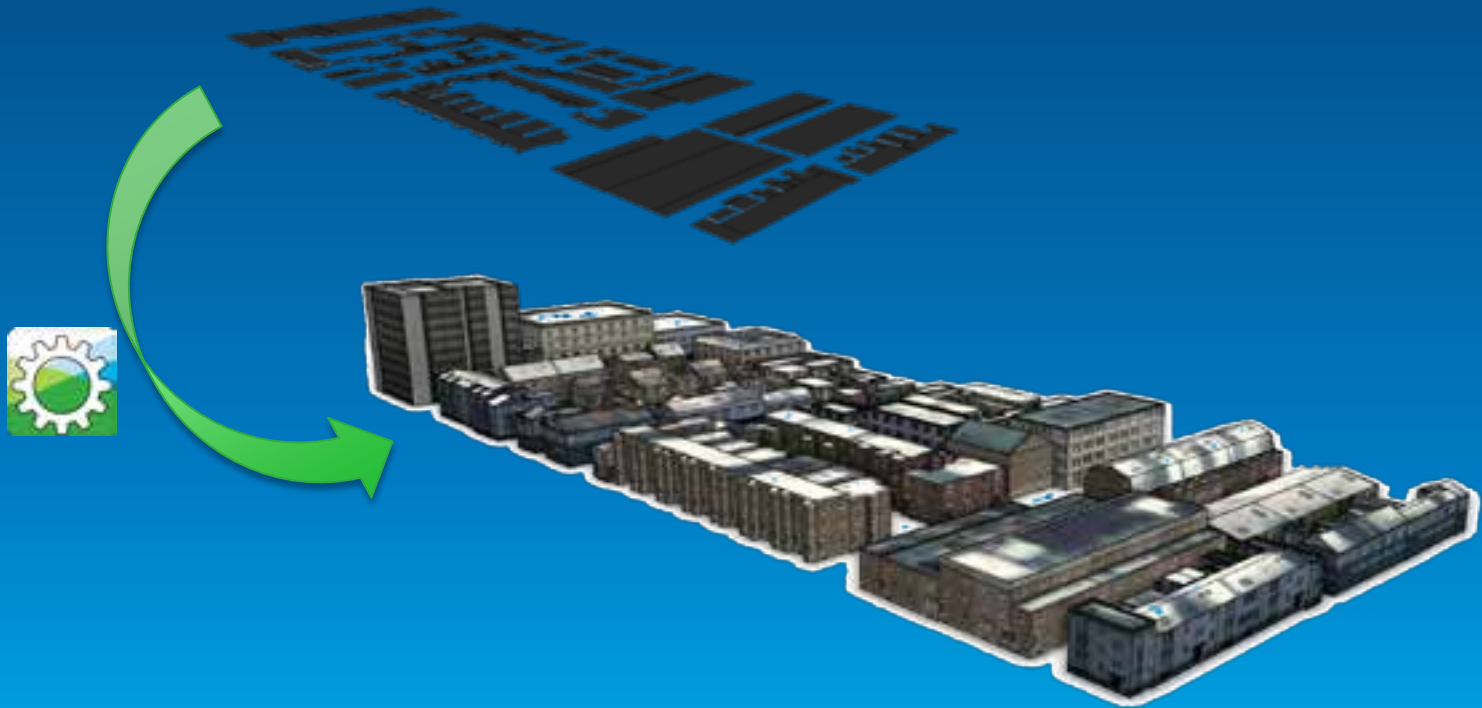




CityEngine GP Tool

Use Cases

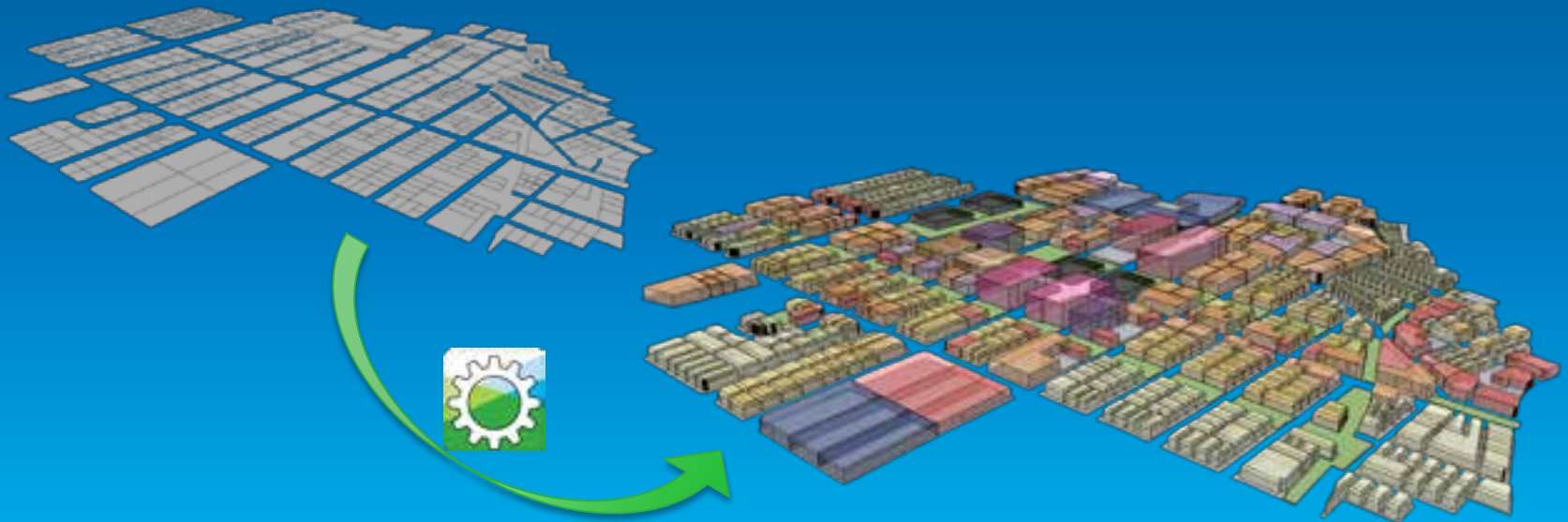
- 2D to 3D: automatic building generation from data model
 - E.g. visualize new development options



CityEngine GP Tool

Use Cases

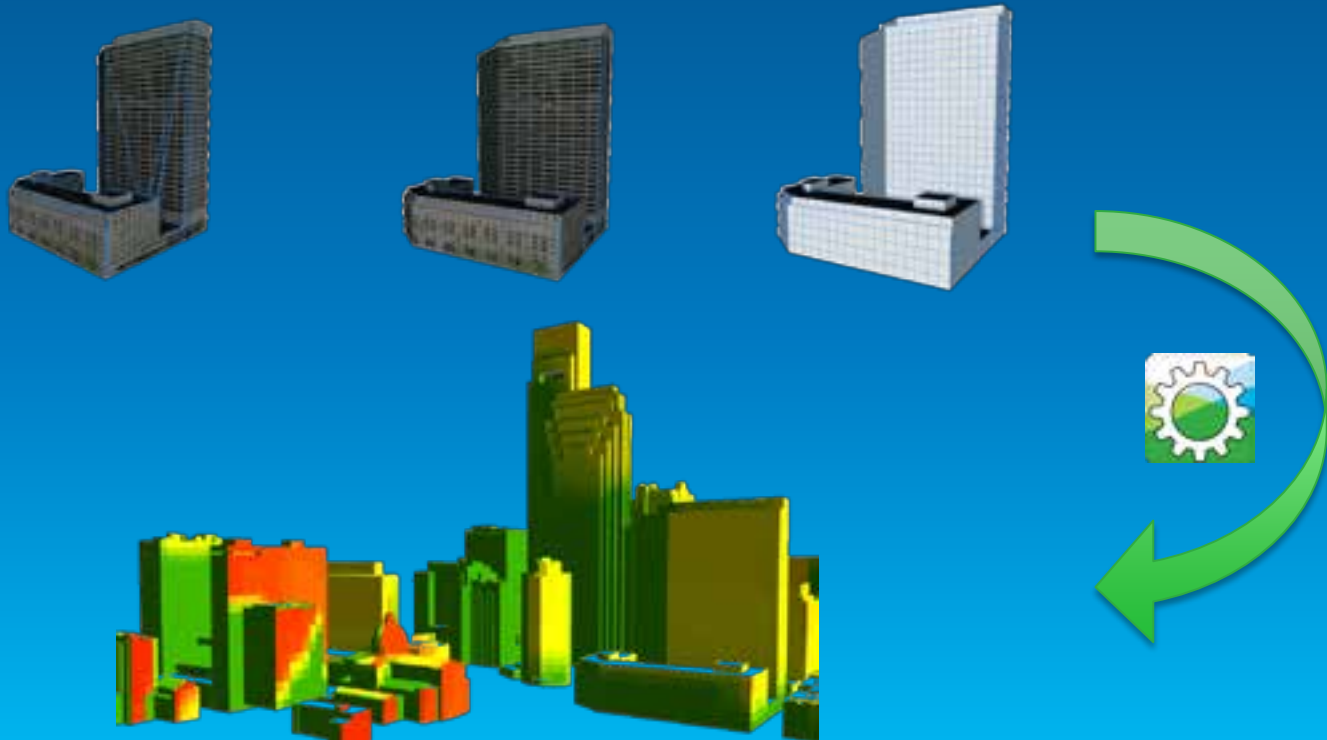
- **2D to 3D: generation of zoning volumes from data model**
 - Intuitive visualization of zoning regulations
 - Analyze impact of regulation changes



CityEngine GP Tool

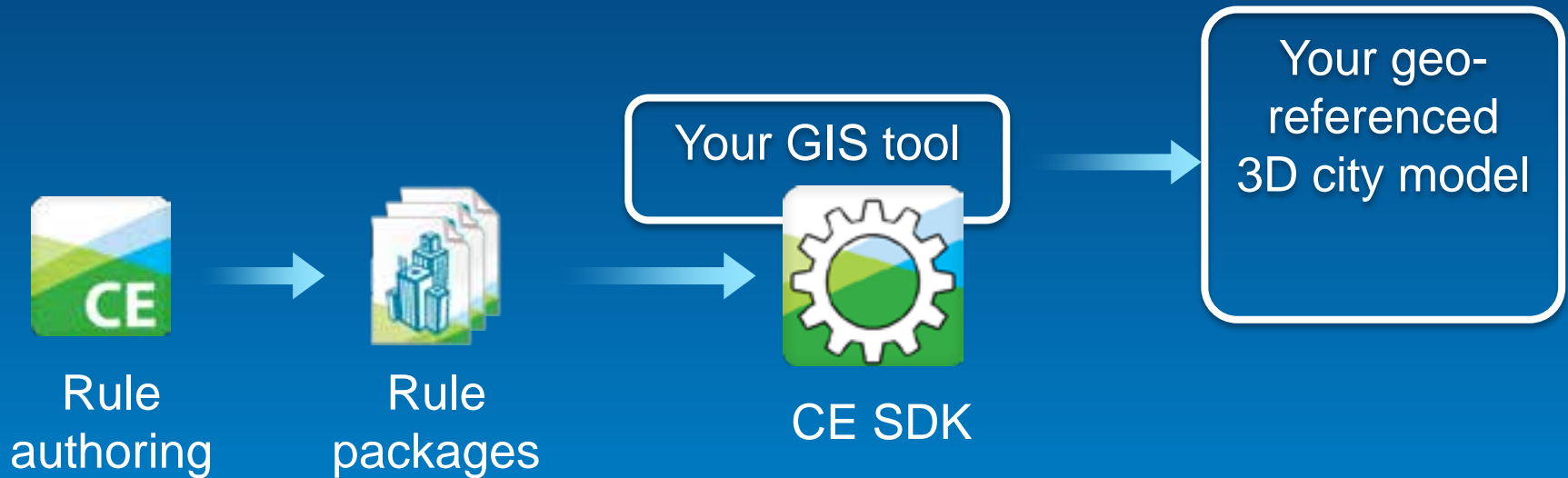
Use Cases

- **3D to 3D: Generate panels on 3D multipatches**
 - Generic rule that subdivides geometry, places point features and/or generates attributes
 - Distribute patches on 3D geometry



3. CityEngine SDK

“Proceduralize” your in-house modeling pipeline



CityEngine SDK

Basis for an Eco-System



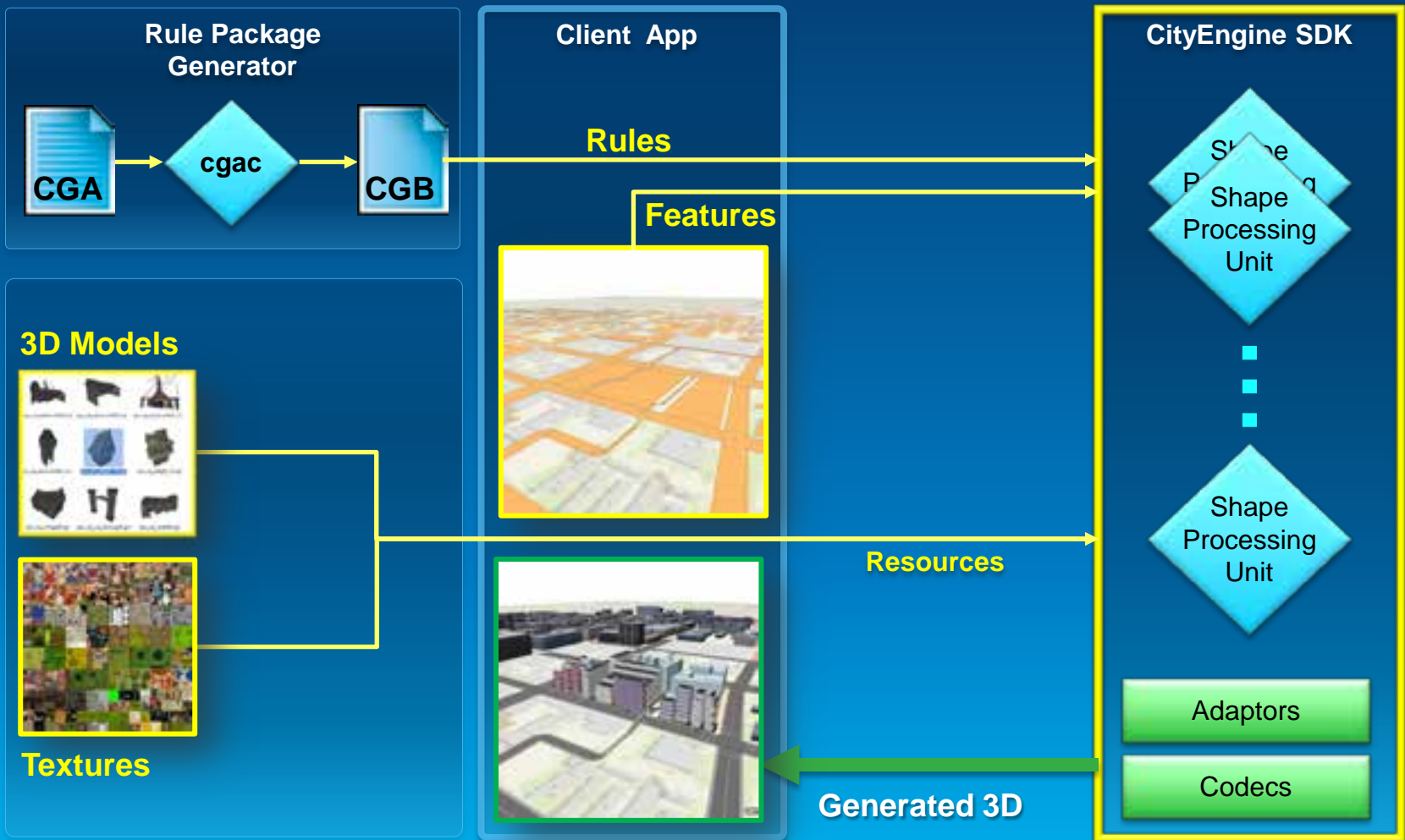
CityEngine SDK

System Architecture

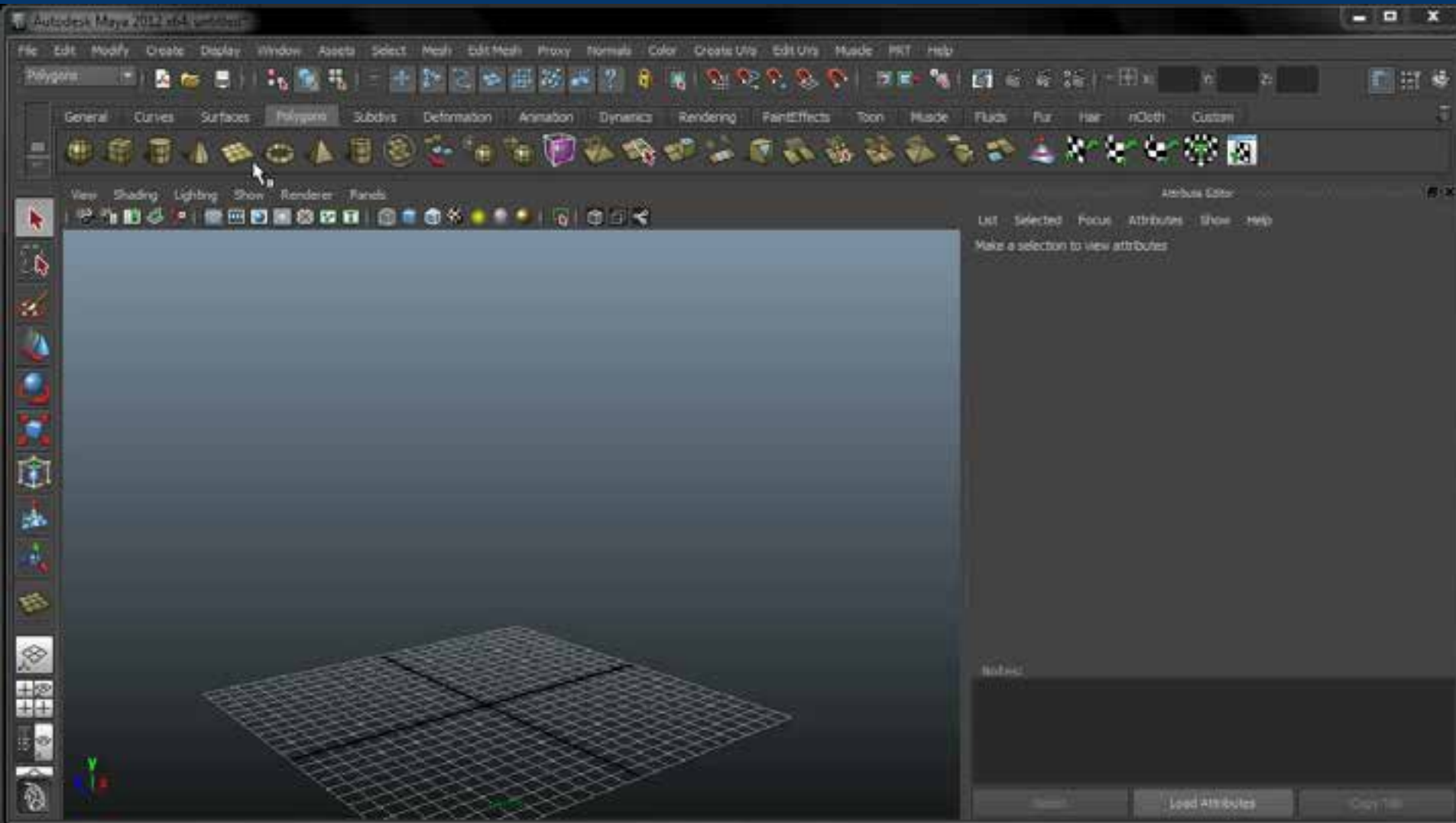


CityEngine SDK

Data & Control Flow



SDK Usage Example – Maya Plugin

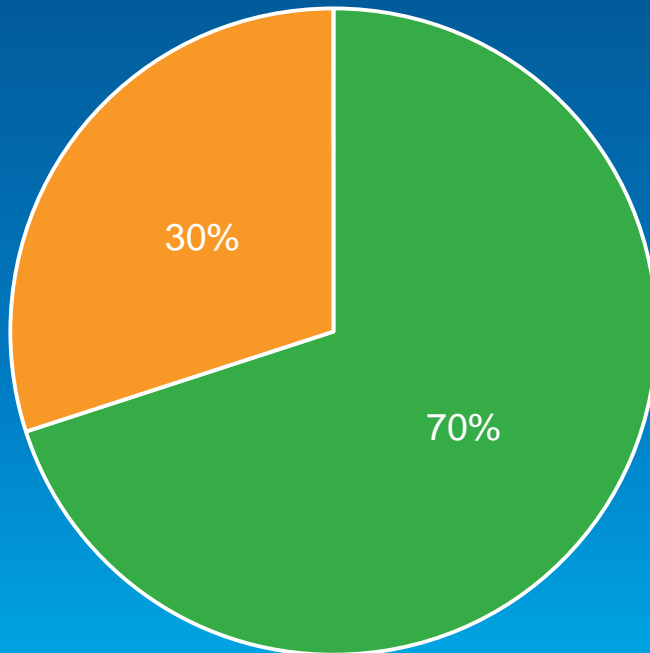


4. Python Scripting



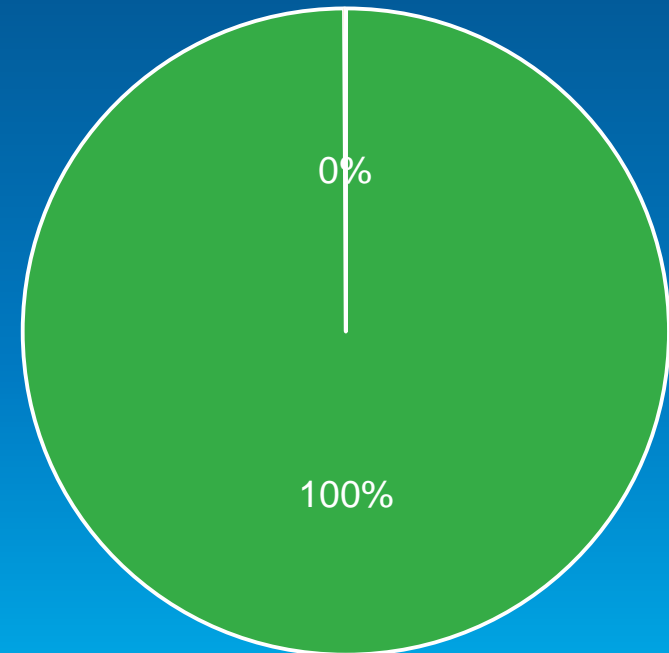
- Automate UI tasks
- CE 2013: All of functions accessible in Python

CityEngine 2012



■ Tools available in Python ■ Unsupported tools

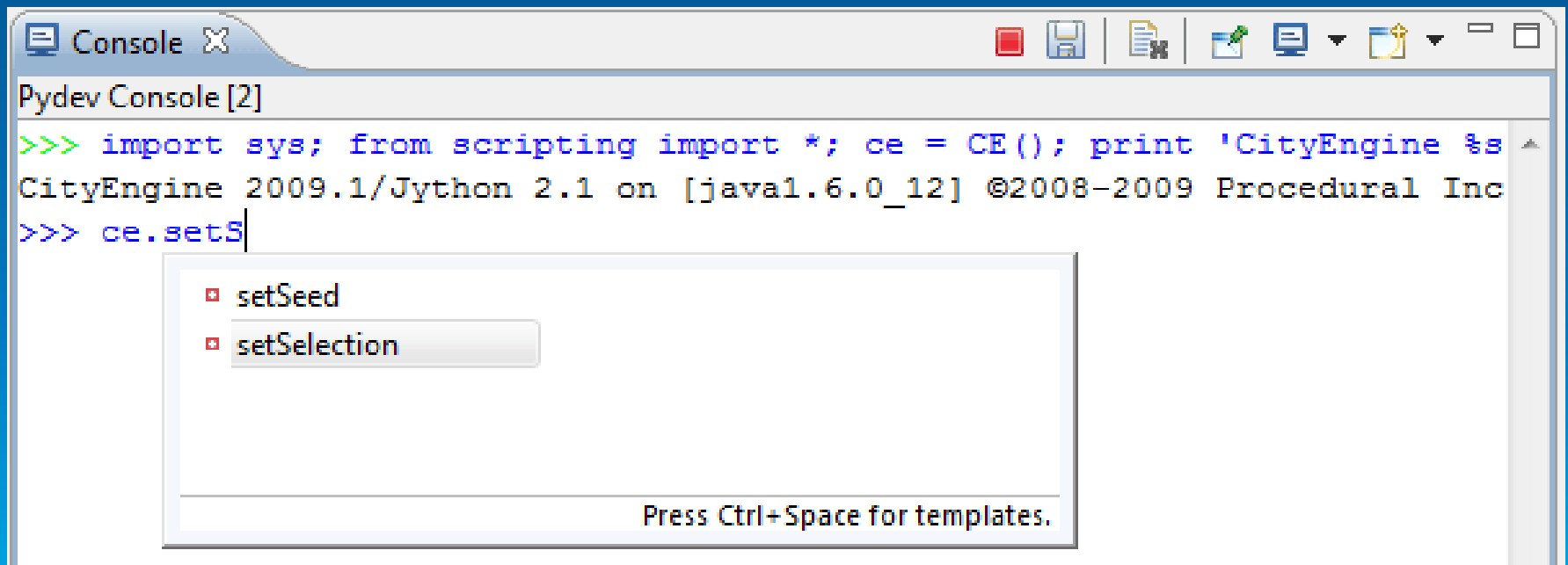
CityEngine 2013



■ Tools available in Python ■ Unsupported tools

Python Scripting

- Python Console:
 - Call CE or conventional Python commands interactively
 - Command completion



The screenshot shows a Pydev Console window with the following content:

```
Pydev Console [2]
>>> import sys; from scripting import *; ce = CE(); print 'CityEngine %s
CityEngine 2009.1/Jython 2.1 on [java1.6.0_12] ©2008-2009 Procedural Inc
>>> ce.setS
```

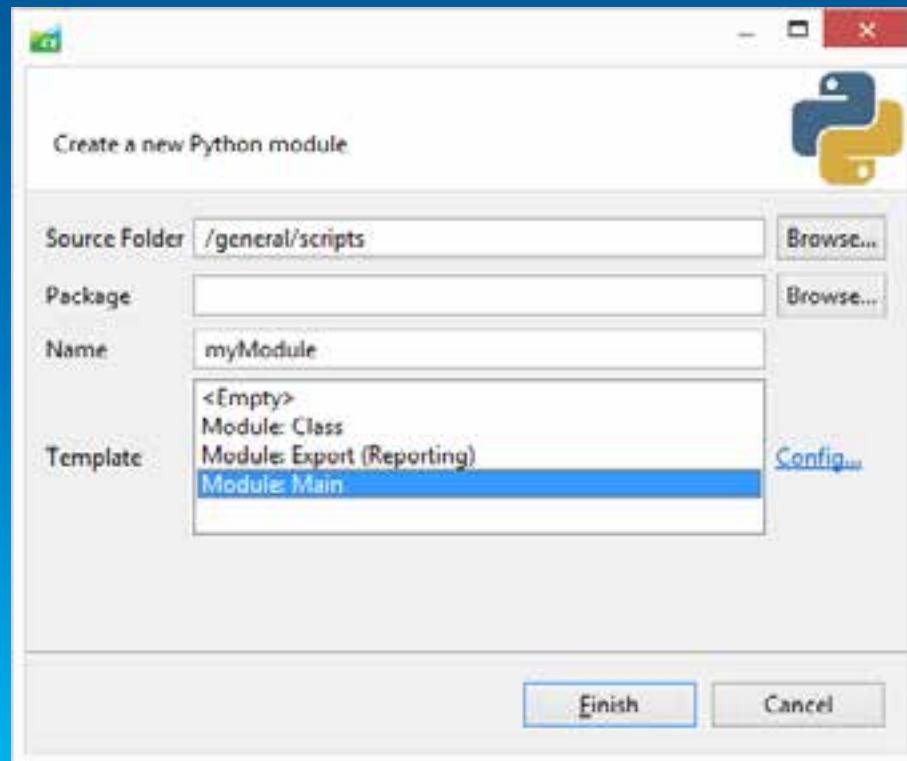
A command completion popup menu is displayed below the code, listing the following options:

- setSeed
- setSelection

At the bottom of the popup, it says: "Press Ctrl+Space for templates."


Python Scripting

- Python Editor
 - Convenient editor
 - Edit and execute



Python Scripting

- Extensive command set
see CityEngine Help for reference
- Use your own Python modules



```
ge = {}

''' generate the street via
the length
'''
def generateStreetFromData(
    selectedSegments = ce.g.selectAll().select
    for segment in selectedSegments:
        width = ce.g.getAttribute(segment, "/ce/s
        width = width.getAttribute(
            segment, "/ce/street/width

''' generate street and sidewalk, width of all sel
the width:
'''
def generateStreetAndSidewalk(
    selectedSegments = ce.g.selectAll().select
    for segment in selectedSegments:
        width = ce.g.getAttribute(segment, "/ce/street/width
        width = width.getAttribute(segment, "/ce/street/width
        width = width.getAttribute(segment, "/ce/street/width

''' generate sidewalk of width of street'''
```

Python
Scripting Reference

Python: Export via script

```
def exportToObj(shapes, exportName):  
    # create new export settings class, define  
    export format  
    objExportSettings = OBJExportModelSettings()  
    # specify export settings  
    objExportSettings.setGeneralName(exportName)  
    # do the export  
    ce.export(shapes, objExportSettings)  
if __name__ == '__main__':  
    exportToObj("pythonExported")
```



scripts/export.py

Python: Export to a set of files

```
def exportMulti(shapes, exportName):  
    for i in range(10,20):  
        # set value of height attribute  
        ce.setAttribute(shape, "/ce/rule/height", i)  
        # call export function  
        exportToObj(shape, exportName + str(i))  
  
if __name__ == '__main__':  
    exportMulti("pythonExported")
```



scripts/export.py

Python: Script Based Export

- Python scripts can run parallel to the export
- Can process arbitrary report data via callback functions
- Powerful mechanism in combination with CGA
`report()`



```
# Called before the export starts.  
def initExport():  
  
# Called for each initial shape before  
generation.  
def initModel():  
  
# Called for each initial shape after  
generation.  
def finishModel():  
  
# Called after all initial shaped are  
generated.  
def finishExport():
```

Python: Write report data to file 1

```
def finishModel(exportContextUUID, shapeUUID,
                modelUUID):
    shape = Shape(shapeUUID)
    model = Model(modelUUID)

    # get report variable 'LotArea' of generated model
    reports = model.getReports()
    shapeName = ce.getName(shape)
    lotAreaSum = sum(reports['LotArea'])

    # storing data to global variable
    global REPORT
    REPORT += "%s,%f\n" (shapeName, lotAreaSum)
```

```
def finishExport(exportContextUUID):
    # write collected report data to file
    global REPORT
    filename = ce.toFSPath("data/report_LotAreas.txt")
    file = open(filename, "w")
    file.write(REPORT)
    file.close()
```

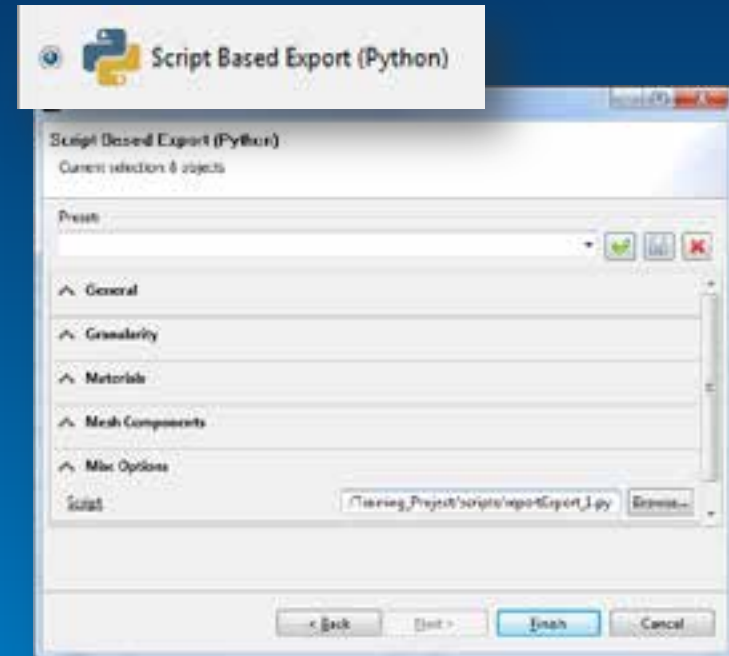


scripts/reportExport_1.py

Python: Write report data to file 2

- Start the script based exporter with python script containing the callback functions
- Collected report data is written to file *data/report_LotAreas.txt*

```
Lot_3,2615.475098  
Lot_2,2573.790283  
Lot_7,1753.116943  
Lot_4,2815.327881  
Lot_1,1365.432495  
Lot_6,2164.343994  
Lot_5,2069.638184  
Lot_0,2551.697510
```

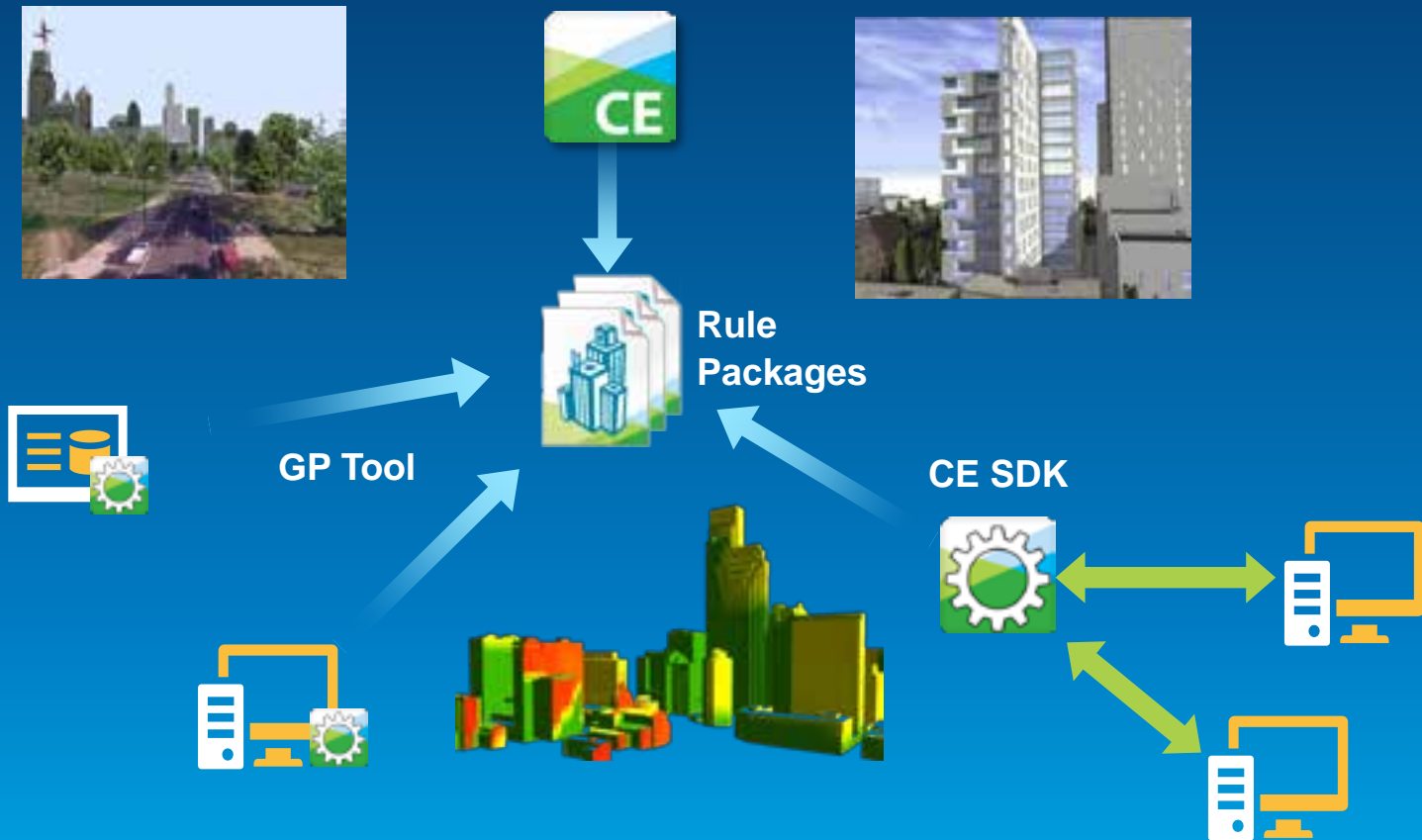


CityEngine 2013 timeline

- **November**
- **SDK: Binaries in CE2013 – coming in Nov**
- **SDK Headers, Documentation, Examples (incl Maya Plugin) TBR in GIT repository over the next 2-3 months**



Summary – CityEngine 2013 great for Developers





Understanding our world.

ArcGIS integration



3D WebScene



Feature from
CityEngine Rules



3D WebScene

