Esri International User Conference
San Diego, California

Technical Workshops | July 25, 2012

Developing with Esri CityEngine

Simon Schubiger

Agenda

- Procedural Modeling
- CGA 101
- Reporting and Analysis

CityEngine

3D procedural modeling and design solution

- **3D City Content**
  - Model cities in 3D using parametric rules

Geometry + Attributes + Rules

- **3D City Design**
  - Rule driven design in 3D

Dynamic + Parametric editing

Rule based 3D content and design

Procedural modeling

3D model creation using rules / algorithms

- **Base geometry**

- **Procedural rules**

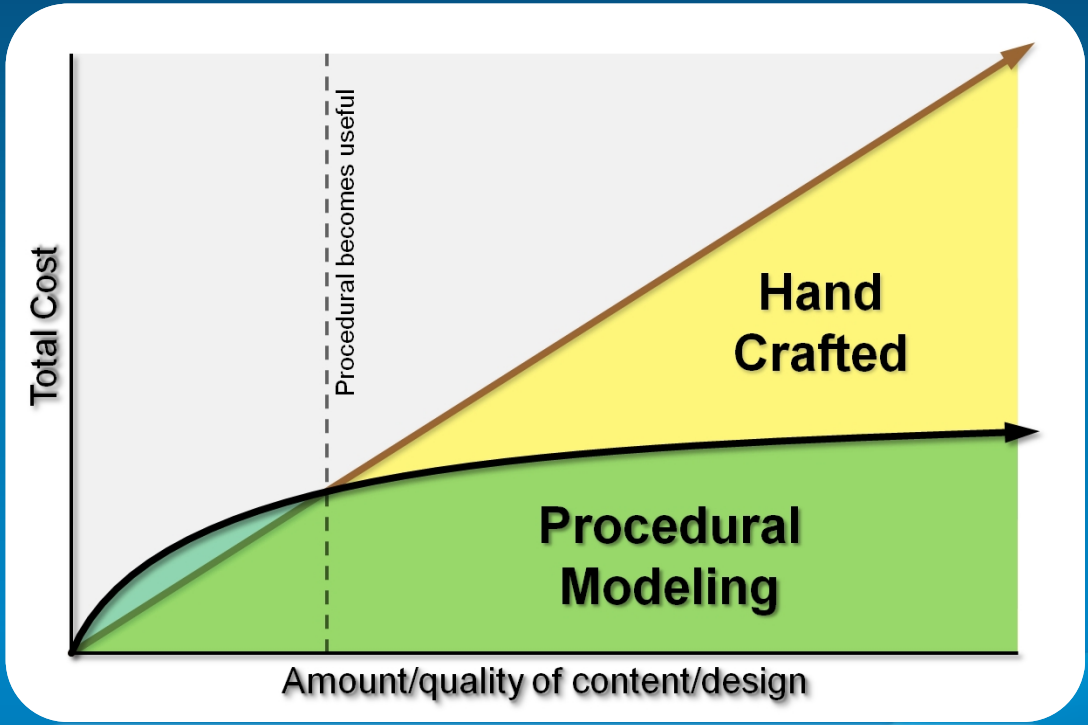
Base geometry

Final 3D model

Iterative refinement

iteratively refine a design by creating more and more detail

Procedural modeling vs. Manual modeling



Time reduction / cost saving

3D city design

3D procedural design

Parametric editing

Add a floor

Add a roof

Dynamic editing

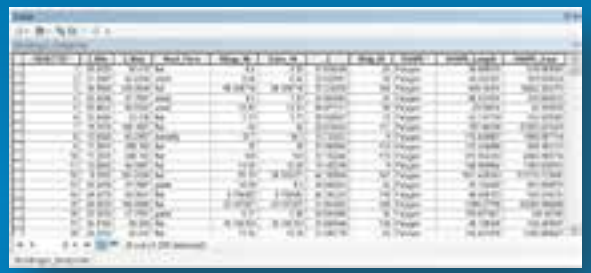
Procedural reporting

Rule based design

3D city content creation

procedural city modeling

Geometry

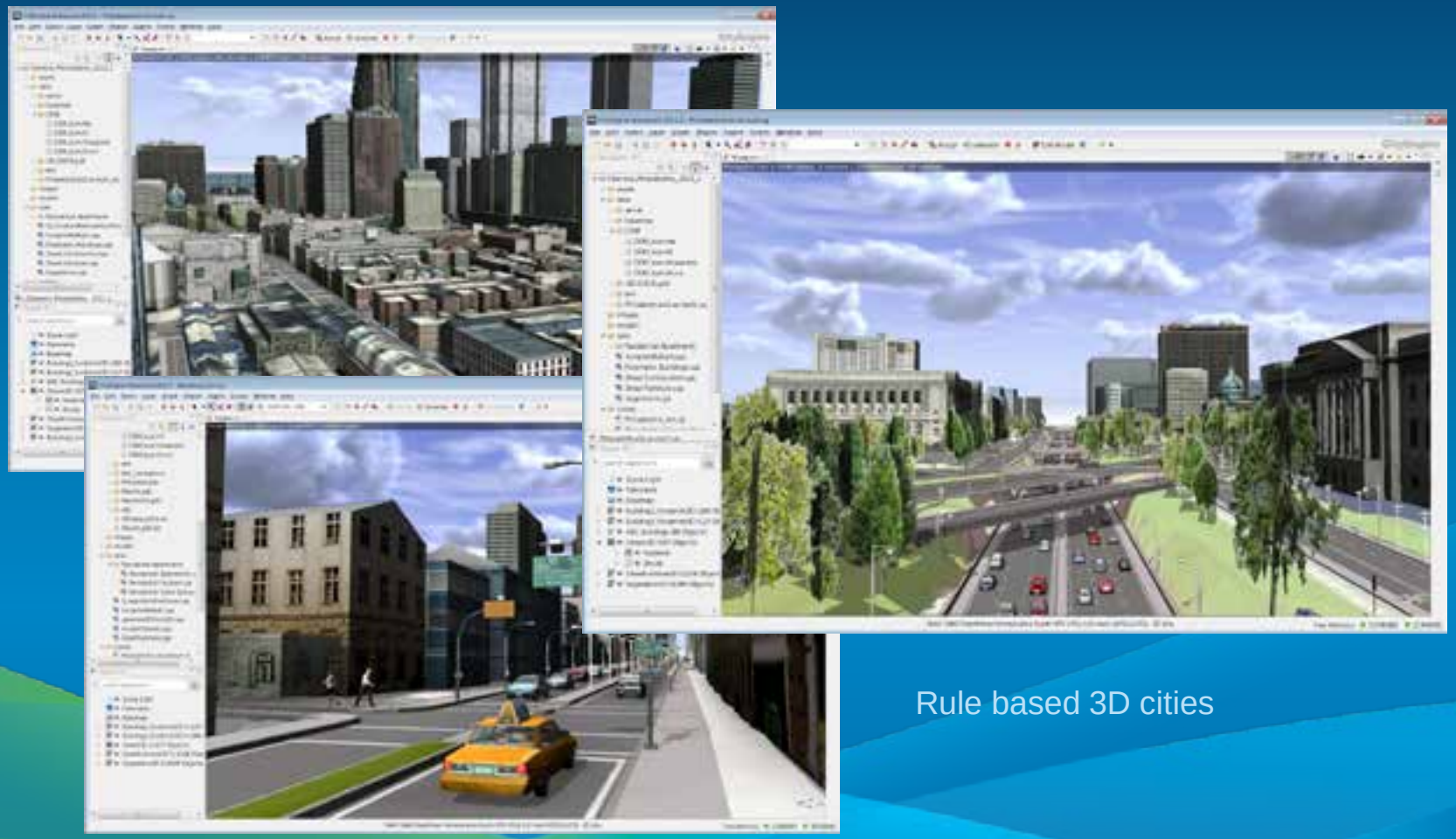


Attributes

Rules

3D city content creation

procedural city modeling



Rule based 3D cities

3D city (Geo)design

Iterative analysis while designing

Design → **Analyze** → **Compare**

Mass modeling

Visibility analysis

Shadow analysis

Façade design

Detailed Façades

Skyline Analysis

CGA 101

Simon Schubiger

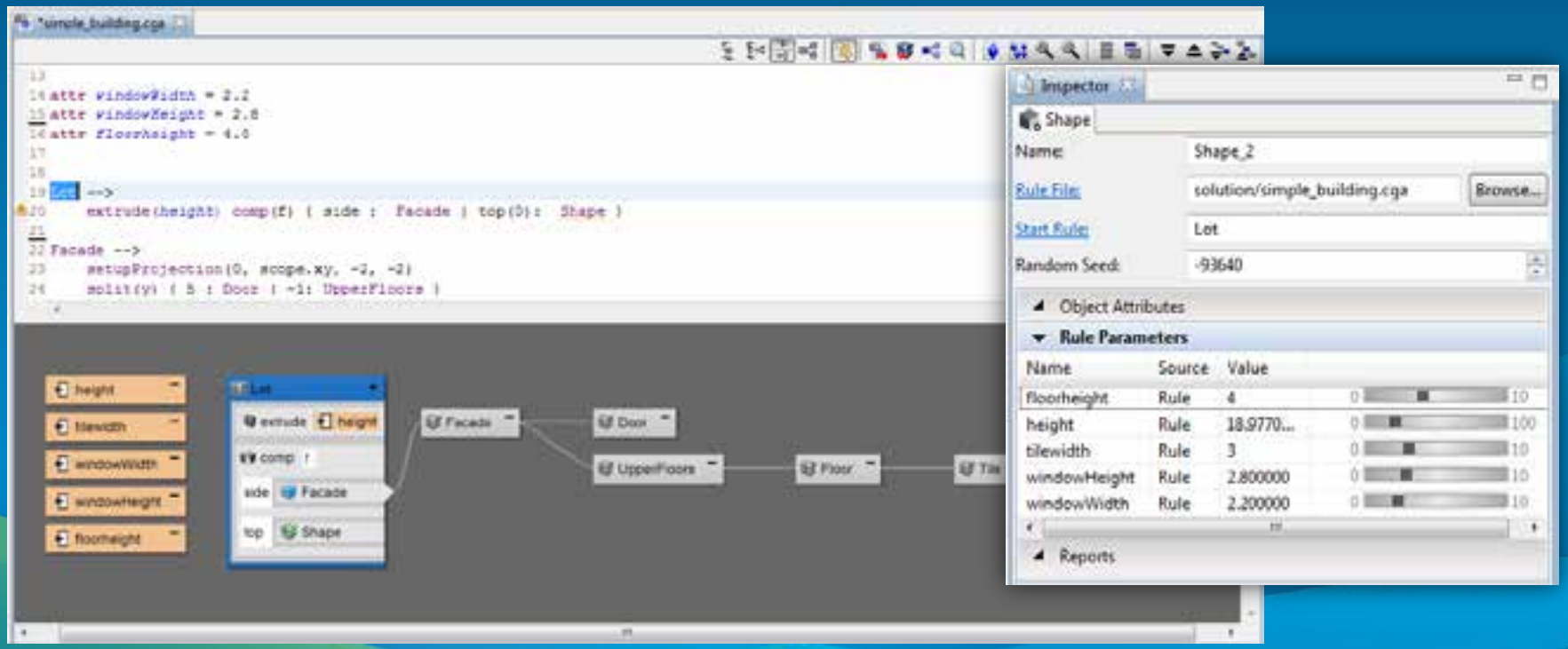
CGA: Shape Grammar

Scripting Geometries with Shape Grammar Rules:

- **Rule-driven modification and replacement of shapes**
- **Iteratively evolve a design by creating more and more details**

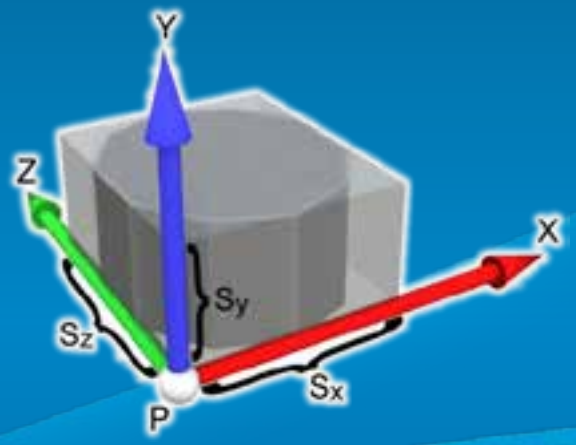
CGA: Example Building

- **Example building rule file**



CGA: Shape Grammar

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

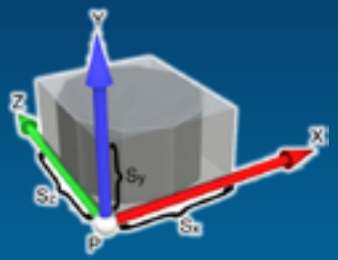


CGA: What is a rule

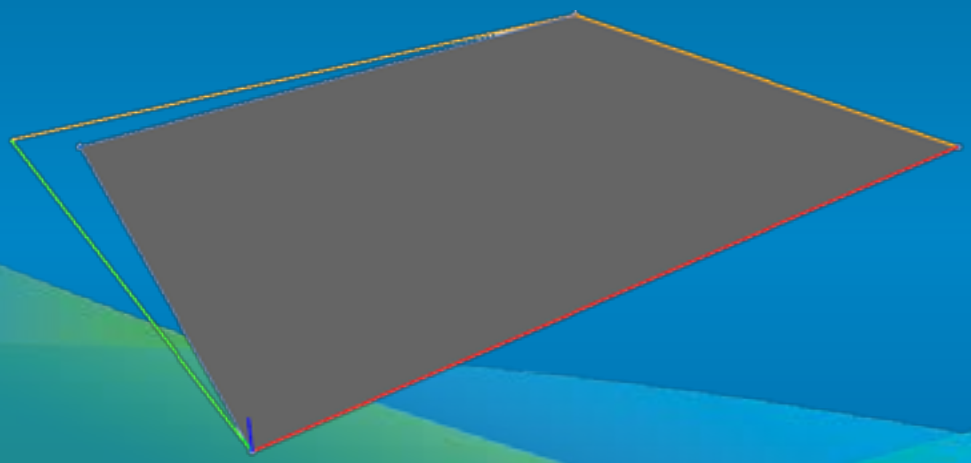
```
A --> extrude(10) B
```

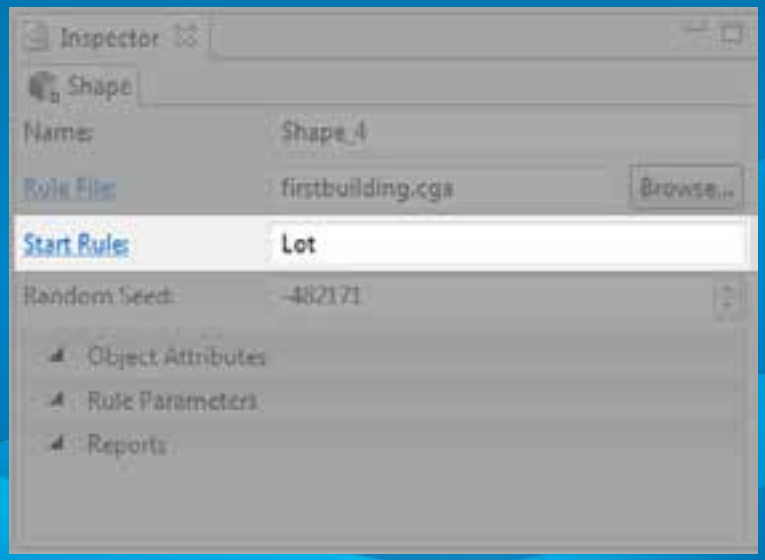
- A CGA rule is an instruction to process shapes
  - à CGA rules can modify shapes
- A and B are shapes
- A modified copy of shape A becomes shape B
- B is called a leaf shape

CGA: Lot = Shape



- A lot is a shape as well
  - Its geometry consists only of one face
  - Its symbol is displayed as Start Rule in the Inspector
  - Is also called Initial Shape because it is the first shape that is processed by the CGA rule set

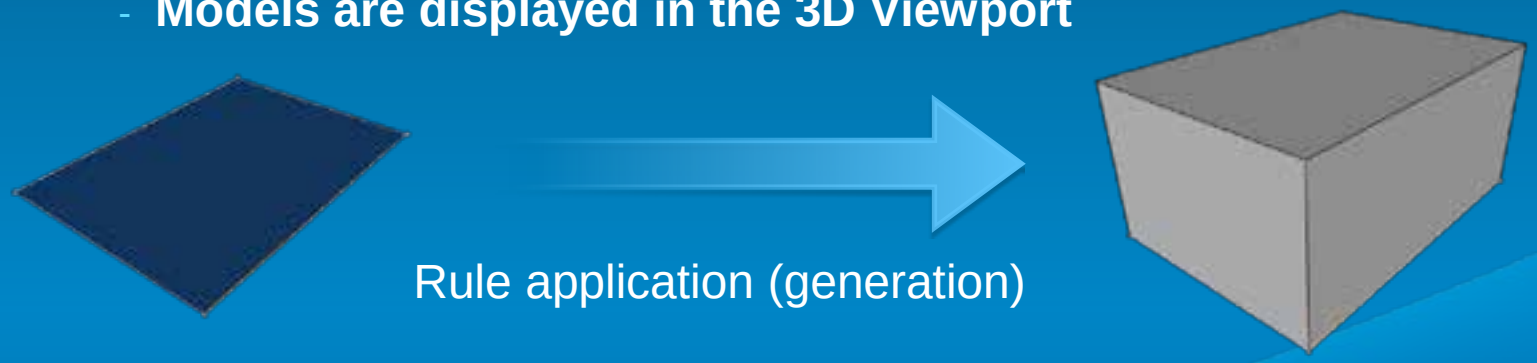




CGA : An actual rule

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

- **The resulting geometry of leaf shapes forms the Model (geometry)**
  - **Models are displayed in the 3D Viewport**



Lot with shape symbol **`Lot`**

Resulting shape **`Mass`**
Displayed geometry

CGA: Shape Replacement

```
Lot --> extrude(10) Mass      Rule #1
Mass --> C D                  Rule #2
```

- Rule #2 is a matching rule for Shape Mass
- Shape Mass is replaced by shapes C and D

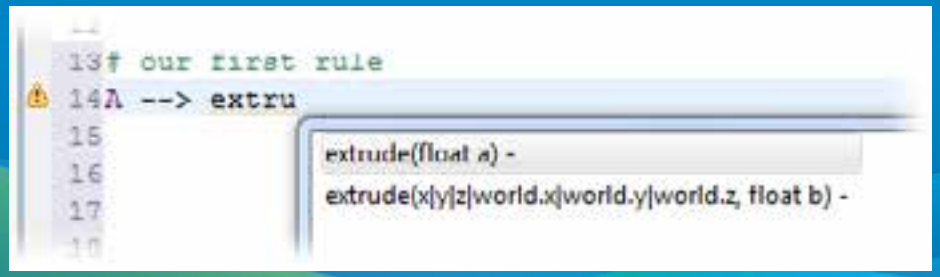
CGA: Syntax

```
attr height = 20

const groundfloor_height = 20

Lot --> extrude(height) Mass

Mass --> comp(f) { top : Roof.
         | front : Frontfacade
         | side : Facade}

# Facade

Facade -->
  setupProjection(0, scope.xy, 1,0.5, 1)
  split(y){groundfloor_height : Groundfloor |

  ~1 : UpperFloors}

Groundfloor -->
  case scope.sx > 10 : color("#cccccc")
  else : color("#ffcccc")
```

- Boolean, float and string expressions
  `1, 0.5, ("#cccccc"), scope.sx > 10`
- CGA-specific keywords
  `attr, top, front, case`
- CGA operations (may have parameters)
  `extrude(height), comp(f)`
- Rules (may have parameters)
  `Lot, Mass, Facade`
- User-defined attributes, constants and functions
  `height, groundfloor_height`
- Comments
  `#Facade, //, /* ... */`

CGA: Text Editor

- Opens with .cga files
- Ctrl-Space gives command completion
- Red underlines denote errors
- Yellow underlines denote warnings
- Split Screen with Visual CGA Editor (VCGA)



CGA: Create first rule

- **Menu : File à New… à CityEngine à CGA Grammar File**

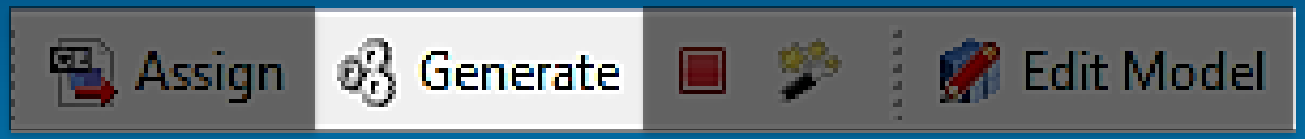
```
# my first CGA rule
Lot --> extrude(10) Mass
```

CGA: Assign rule file to lot

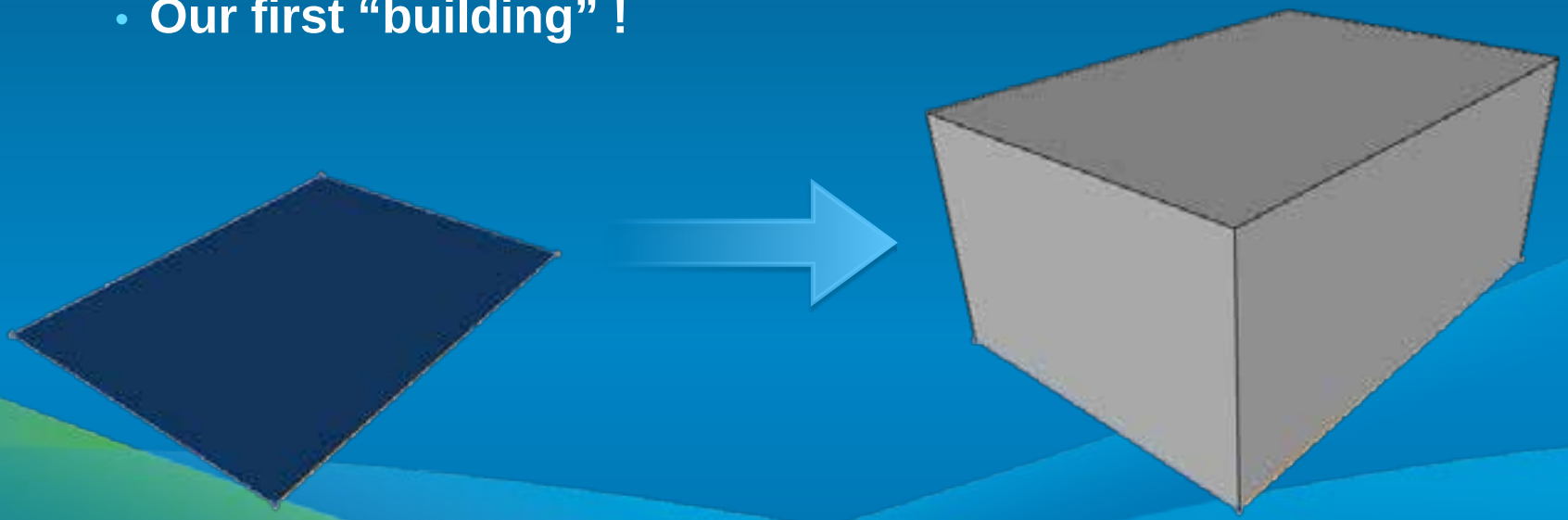
- Now we hook up our first rule to an actual building lot
- Rule File and Start Rule are shown in the Inspector
- The Start Rule defines the rule that is applied first

CGA: The first «building»

- **Menu : Initial Shape à Generate**

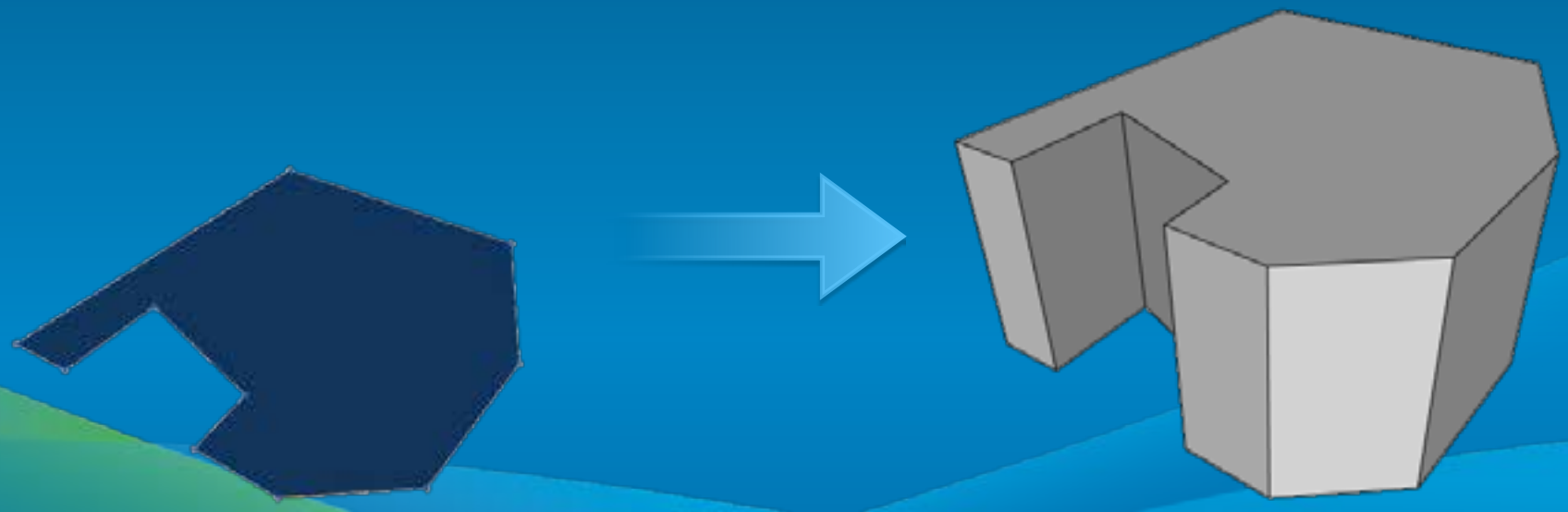


- **Our first “building” !**



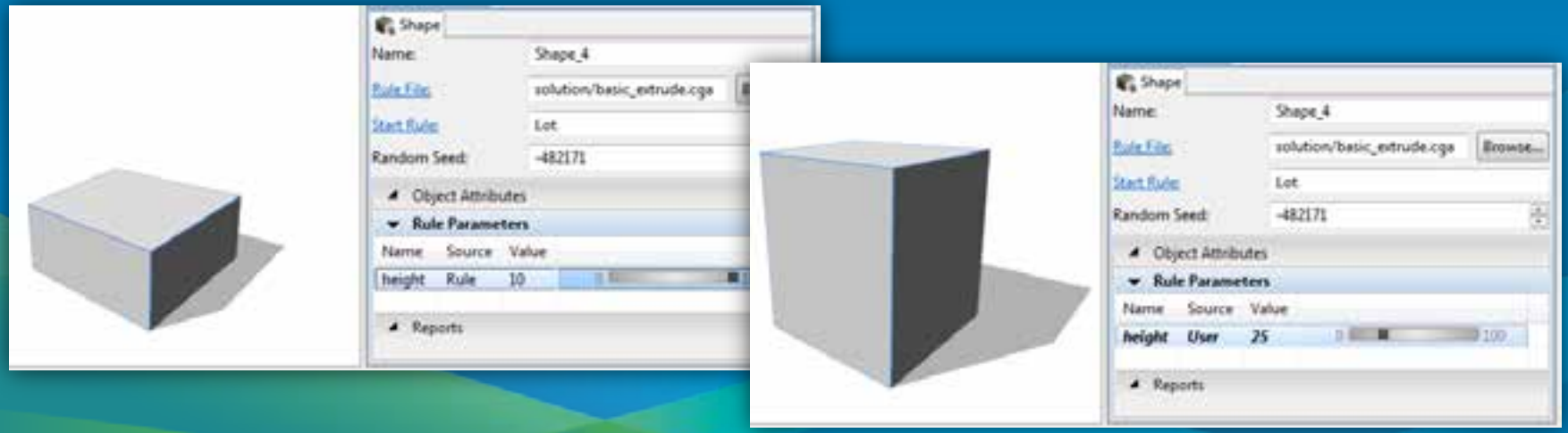
CGA: Live Mode

- Use transform tools to update the lot shape
- The generated model is updated on-the-fly
- Helps to analyze and test rules



CGA: Attributes

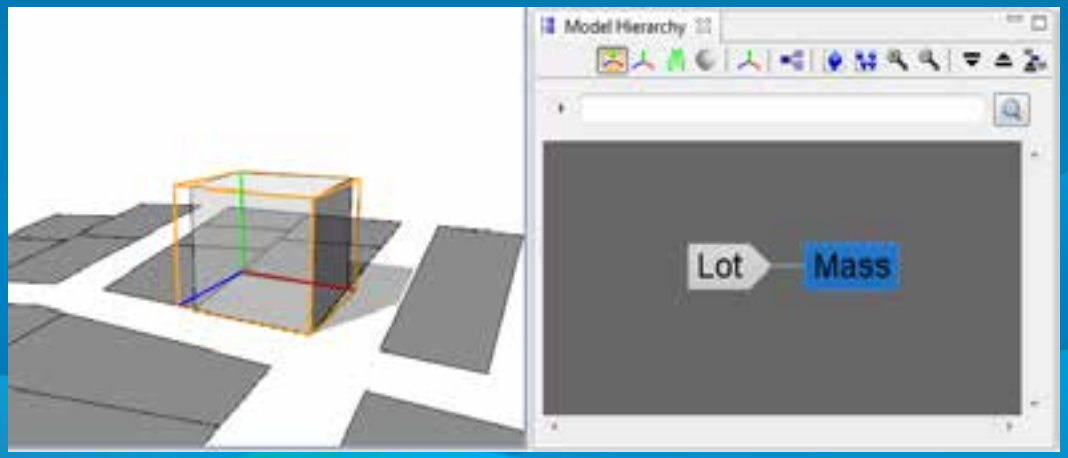
- Add attribute *height* to CGA file
- `attr height = 10`
  `Lot --> extrude(height) Mass`
- Attribute appears in rule parameters in the Inspector
- Rule attributes can be externally controlled (e.g. through Inspector)



CGA: Model Hierarchy Viewer

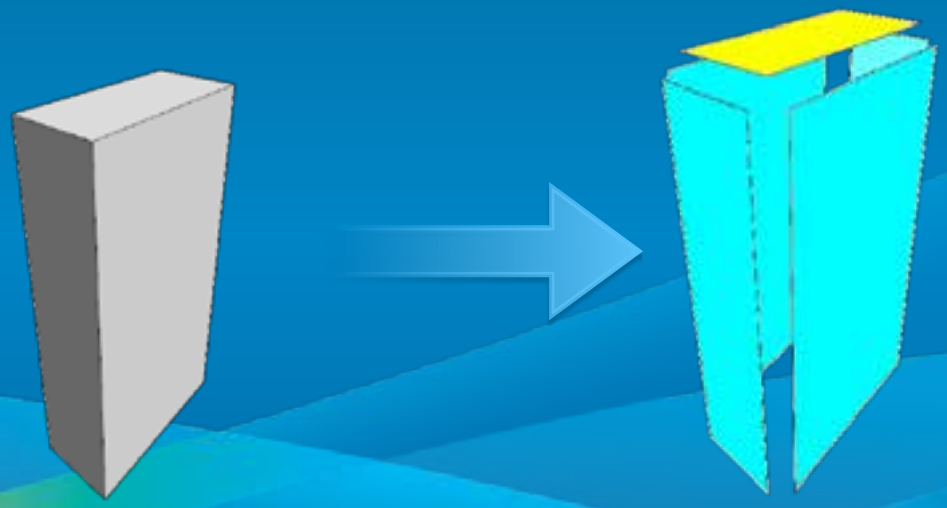
- Generated Model can be viewed in Model Hierarchy Viewer

  - Menu : Window à Show Model Hierarchy
  - Toolbar : Edit Model
- Very helpful for writing and analyzing rules
  - Displays additional info (e.g. scope)
  - Shows generated structure (tree)



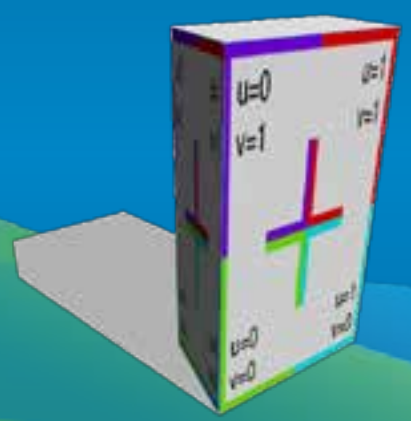
CGA: Component split

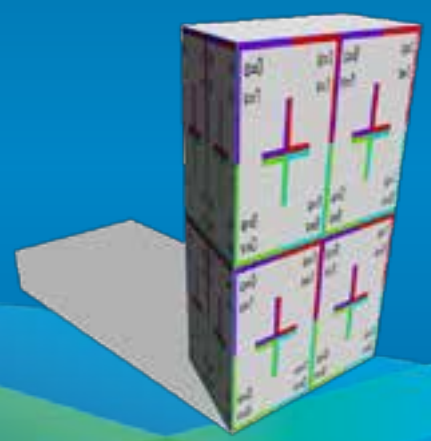
- Get face components:
  - Component split : `comp(f){top : Roof | side : Facade}`
  - Also works for edges and vertices (`comp(e), comp(v)`)
- Different semantic selectors such as `top, side, vertical, left, aslant, …`

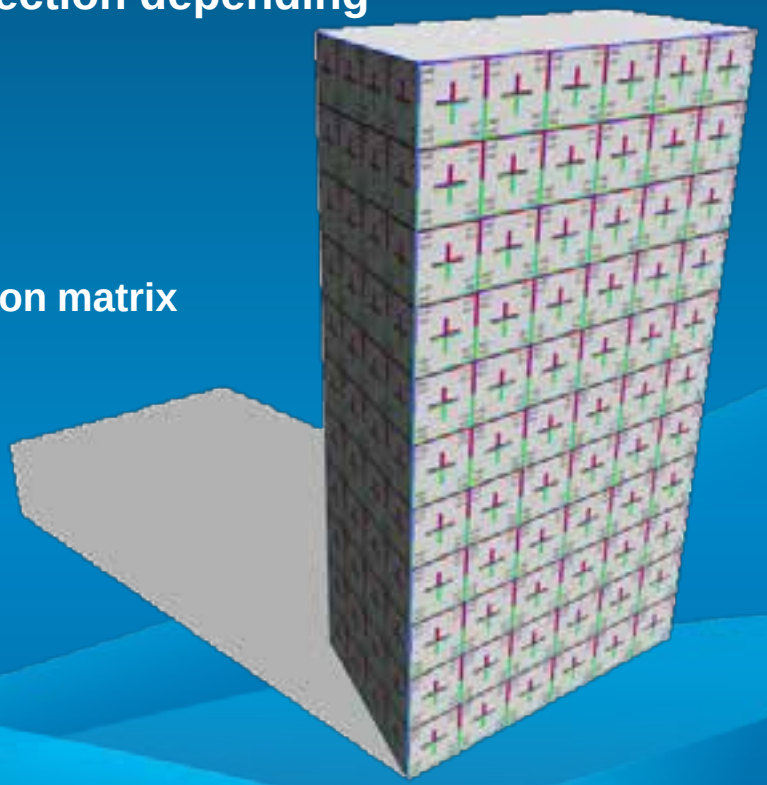


CGA: Texturing

- `texture("builtin:uvtest.png")`
  - Loads a texture
- `setupProjection(uvset, axes, width, height)`
  - Sets the projection matrix for later UV projection depending on the current scope
  - UV scaling is controlled using the `width` and `height` arguments
- `projectUV(uvset)`
  - Creates texture coordinates by applying projection matrix





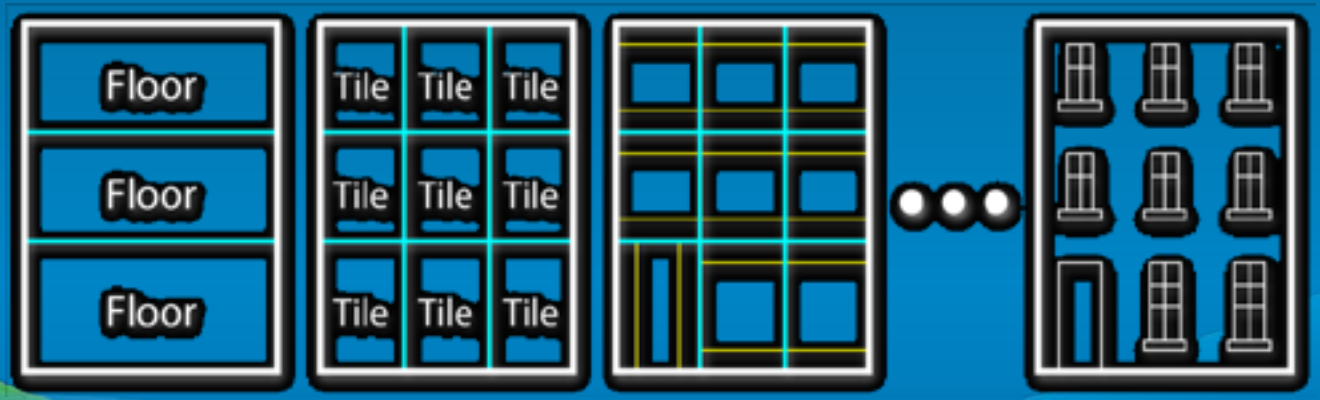


CGA Shape Grammar: Façade Modeling

CGA: General Facade Schemes

Most common subdivision scheme:

Facade 4 Floor 4 Tile 4 Wall & Window/Door



CGA: Repeat split

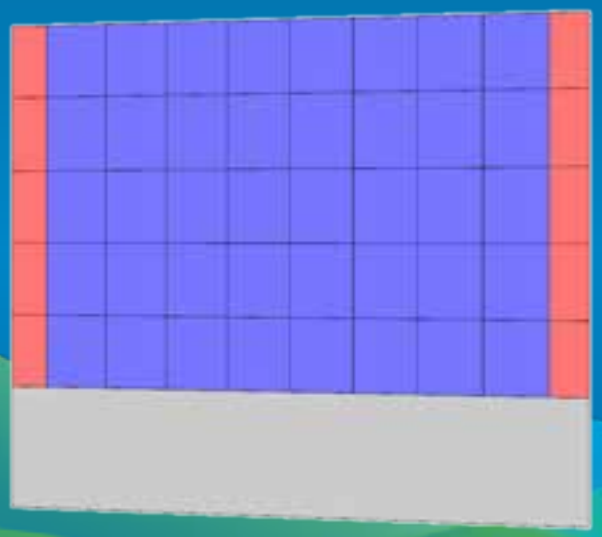
- Asterix marks a repeating split
  - `split(y){~width : A}*`
  - Floating operator ~ ensures fitting sizes
- Normal and repeating splits can be nested:
  - `split(y){groundfloorheight : Groundfloor`
    `| {~ floorheight : Floors}* }`

CGA: Rhythm split

- **Nesting normal and repeating splits à rhythms**

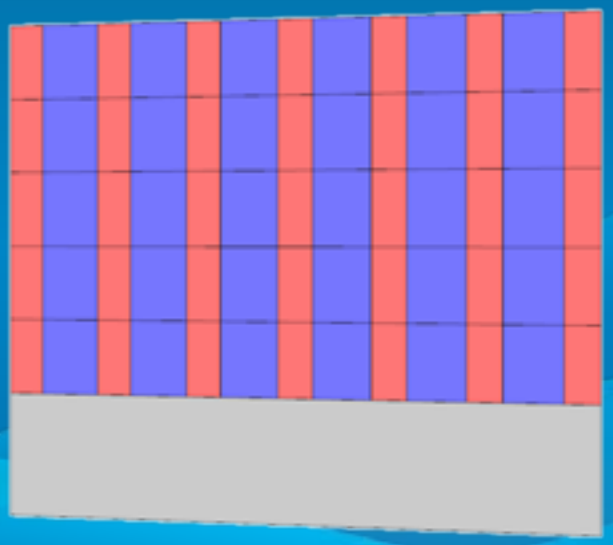
§ A B* A

```
split(x){ widthA: TileA
        | {~ widthB : TileB}*
        | widthA : TileA }
```



§ { A B }* A

```
§ split(x){ { widthA: TileA
          | ~ widthB : TileB }*
          | widthA : TileA }
```



CGA : Facade assets

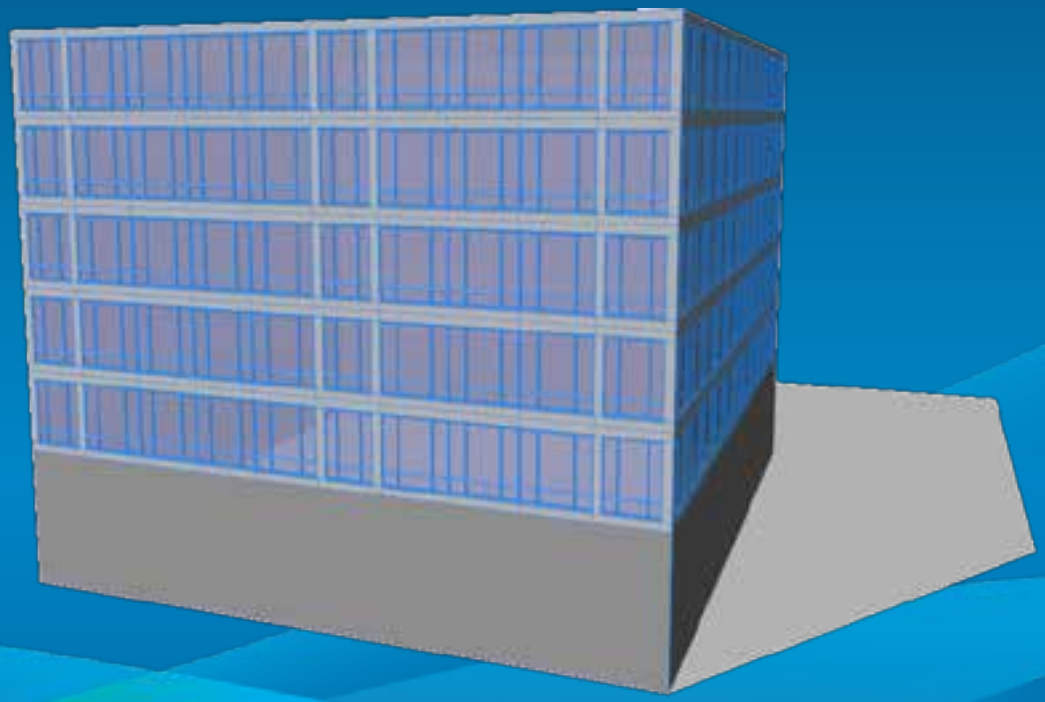
- `i("facade/window.obj")`
- Inserting a window asset
  - LOD : Texture or geometry asset

Texture

Low LOD

Geometry

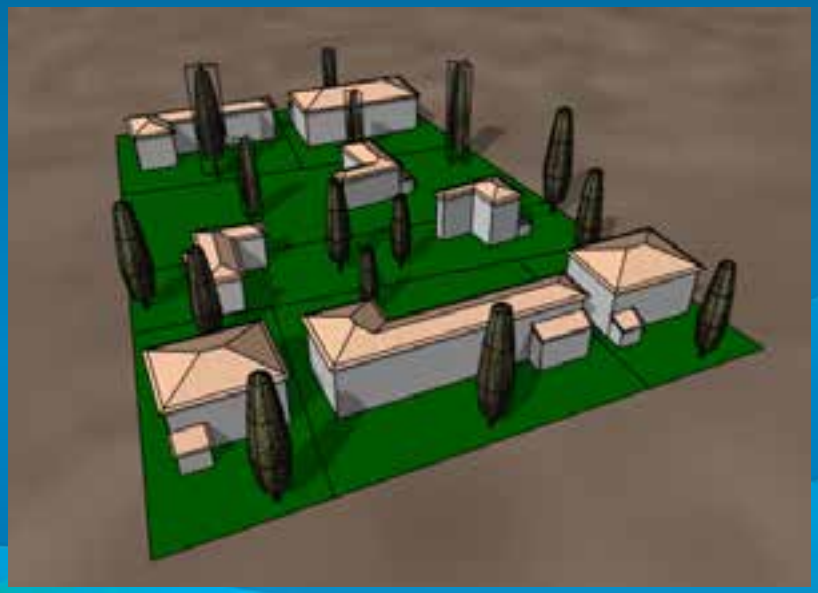
High LOD



CGA Shape Grammar: Residential Building

CGA: Residential Building

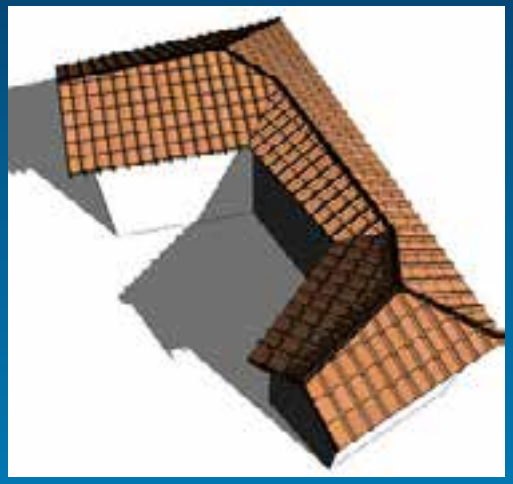
- Additional CGA commands and concepts:
  - Roof command : `roofHip(), roofShed()`
  - Find inner Rectangle : `innerRect`
  - Placing assets

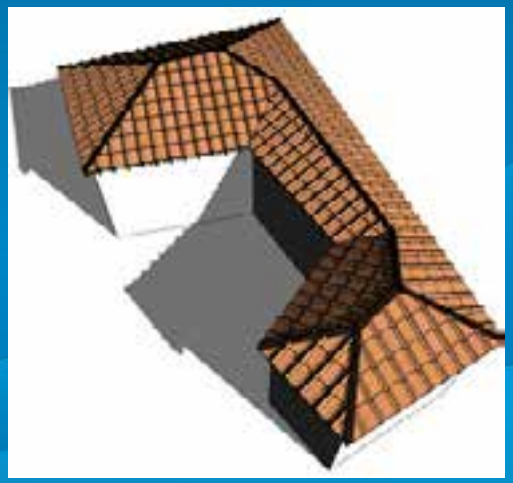


CGA: Roof commands

- Special commands create roof shapes:
  - `roofGable()`
  - `roofHip()`
  - `roofPyramid()`
  - `roofShed()`

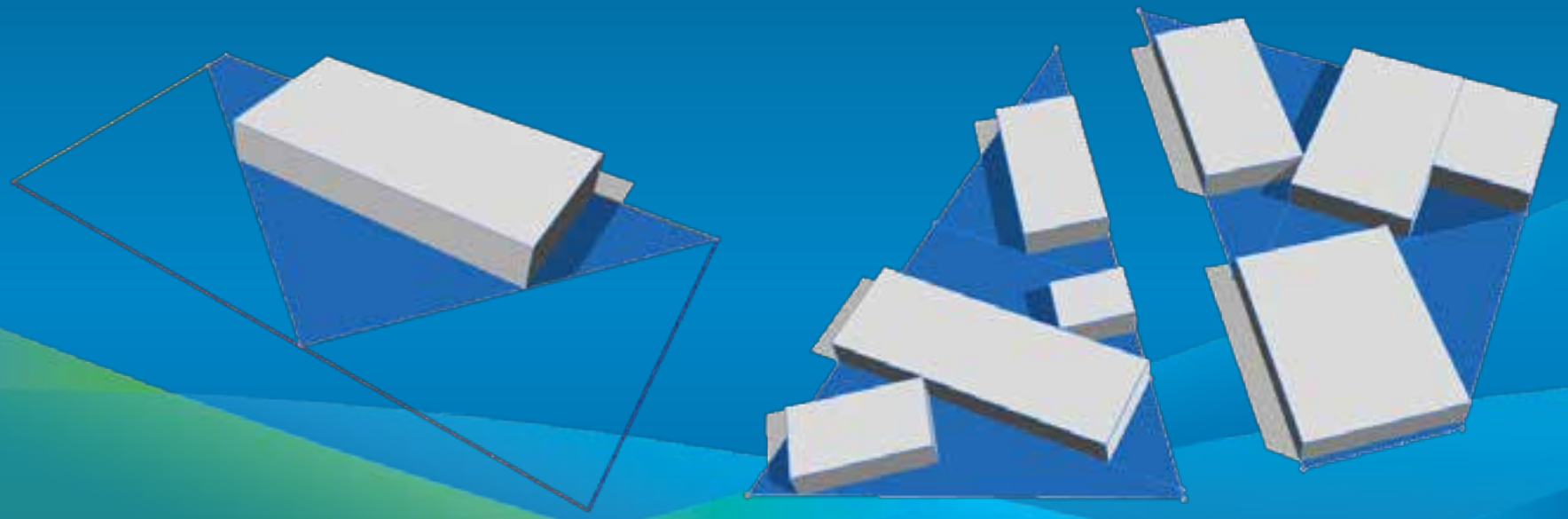
- With additional settings such as angle and overhang





CGA: innerRect

- `innerRect`
- Transforms shape into a rectangle fitting into current geometry

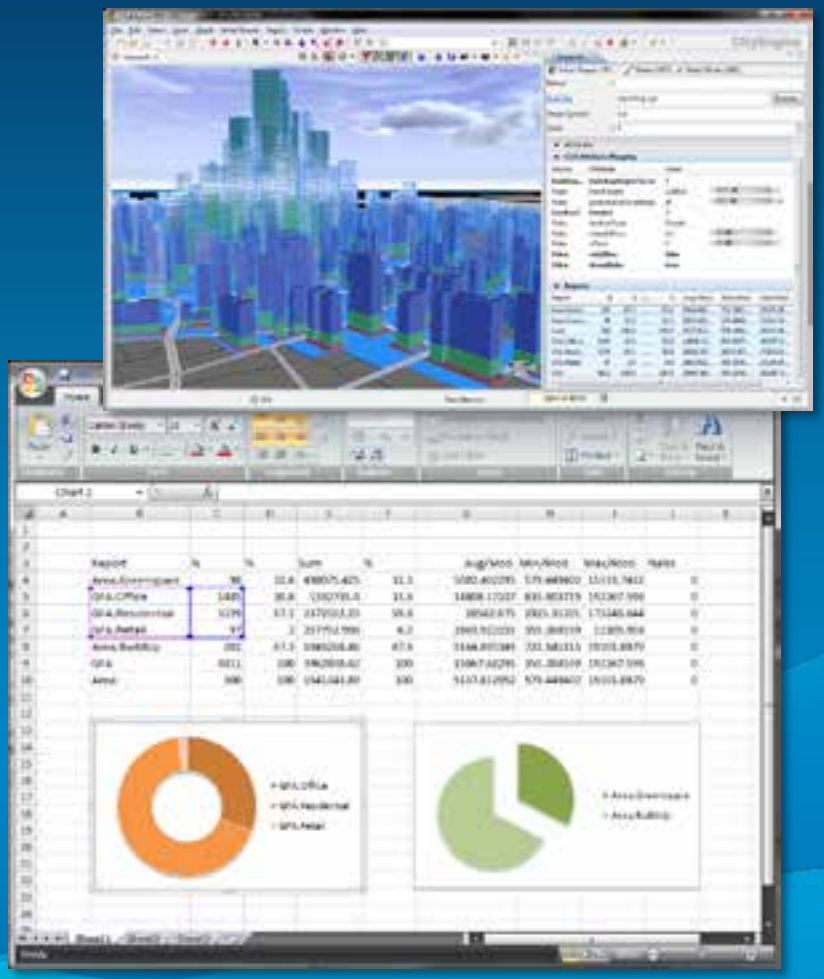


Reporting and Analysis

Rule-based Report Generation

Usage:

- **Automatic report generation to analyze and verify the city design with statistics**
- **For example automatically calculate GFA, FAR, or material quantities on a city-wide scale**



The Report Operation

- Collecting *arbitrary* data during model generation
- Synopsis:

```
report(key,value)
```

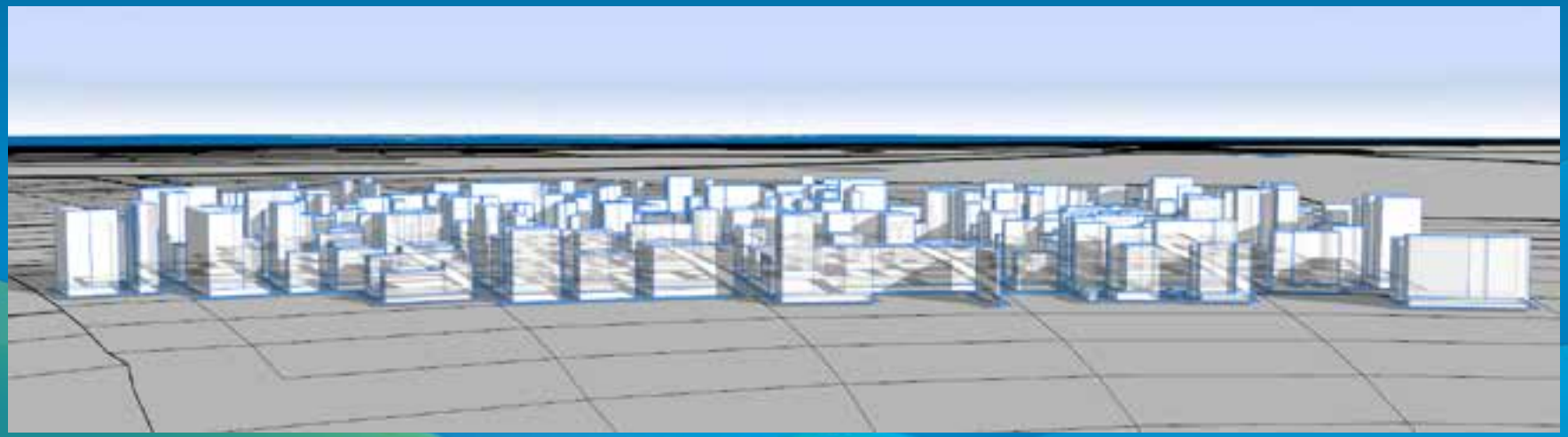
- Example:

```
OfficeFloor -->
 report("GFA.Office",geometry.area)
```

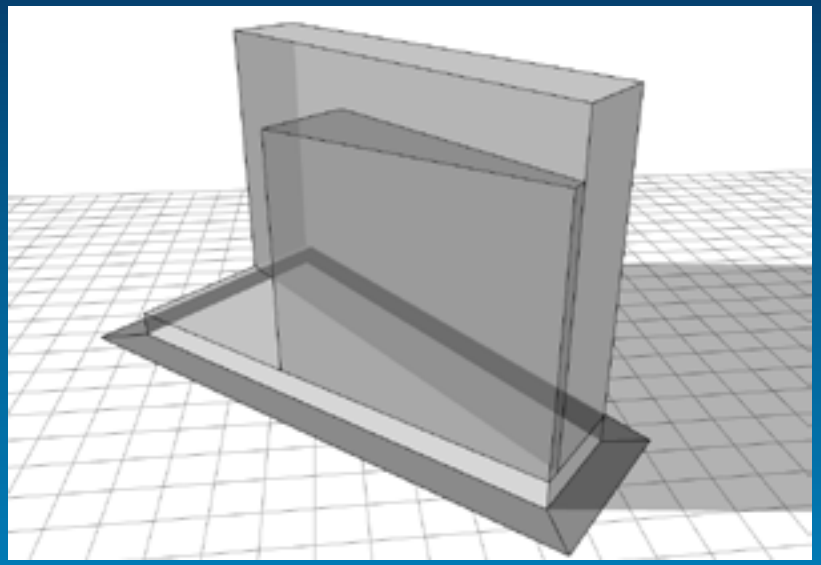
- **Values can be reported anywhere in any rule**

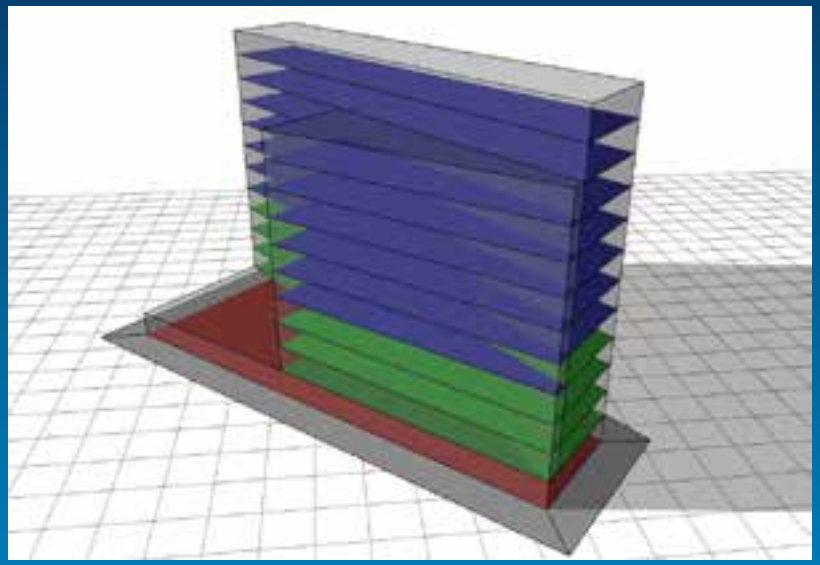
Report Project

- Dubai street-network from OpenStreetMap.org
- Quickly extended using a spiral street network growth



Simple Building Design





Attributes:

- **# Floors & their Height**
- **Facade Setback**
- **L, T, or U-Shape**
- **Mixed Usage Ratio**

Visualization:

- § Transparent: Shell
- § Red: Retail Floors
- § Green: Office Floors
- § Blue: Residential Floors

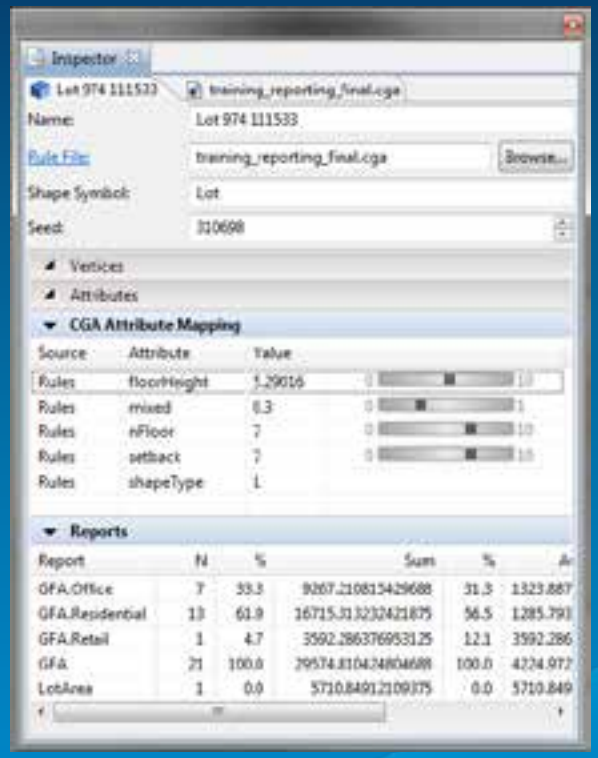
Add Reporting Variables

- **Single value per building:**

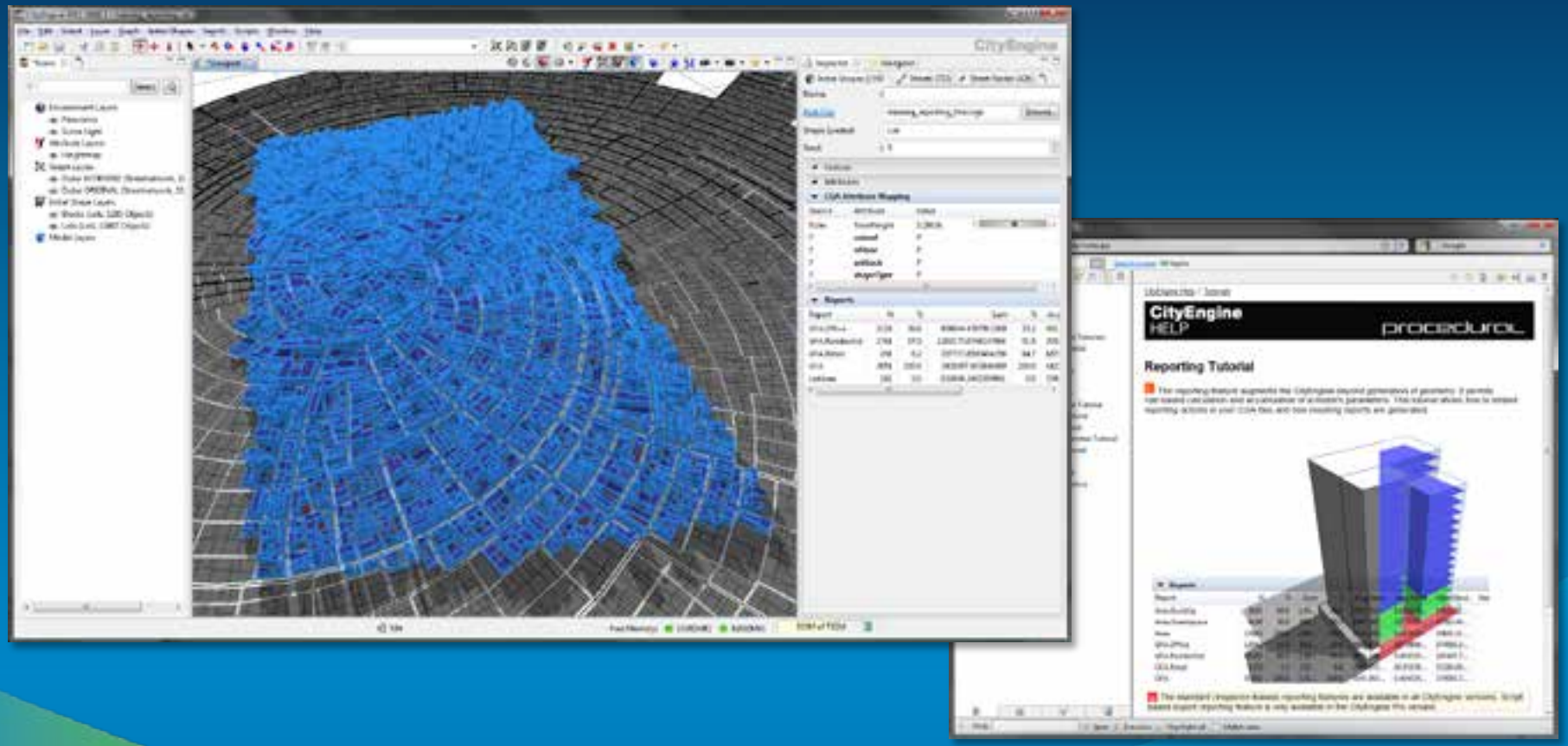
```
Lot -->

report("LotArea",geometry.area)
   ...
```

- **Multiple values (for each floor):**

```
FloorBottom(type) -->
  case type == "Retail":
    color("#ff4444")

report("GFA.Retail",geometry.ar
ea)
  case type == "Mixed"...
    ...
```



City-wide Report Generation



- For a more advanced example see CityEngine Tutorial 11

Where to go from here

- Get a free trial version of CityEngine at `www.esri.com/cityengine`
- Have a look at our tutorials
- Check out the CityEngine help
- Meet the team at the Esri Lab booth

esri