Modeling 3D Urban Spaces Using Procedural and Simulation-Based Techniques

Procedural Urban Modeling in the Industry

Pascal Mueller
Procedural Inc.

©2010 Fold 7

Procedural Urban Modeling in the Industry

INTRODUCTION

Industries

- **Media & Entertainment**
  Film, TV, Games, ...
- **Architecture**
  Urban Design, Architectural Visualization, ...
- **Geospatial Domain**
  Urban Planning, Navigation, Defence, ...
- **Others**
  Cultural Heritage, Education, ...

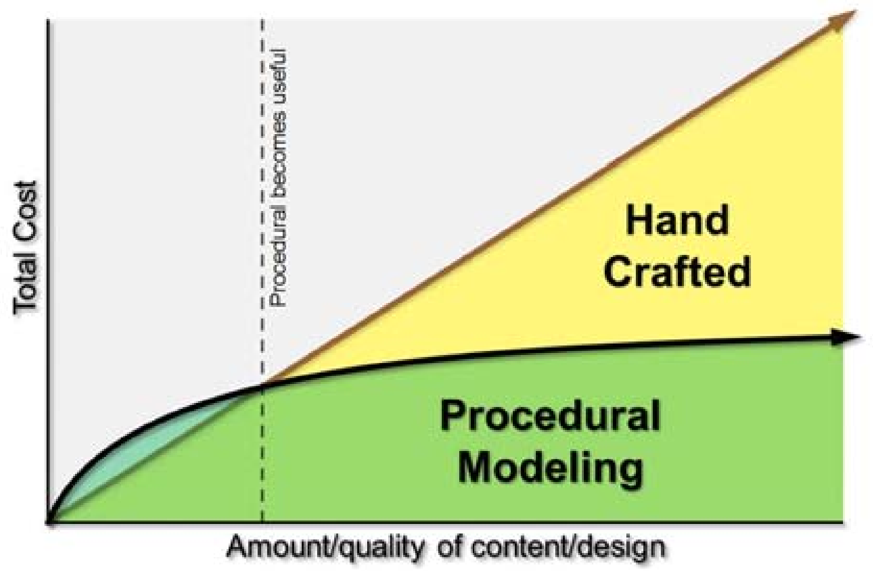
Applications

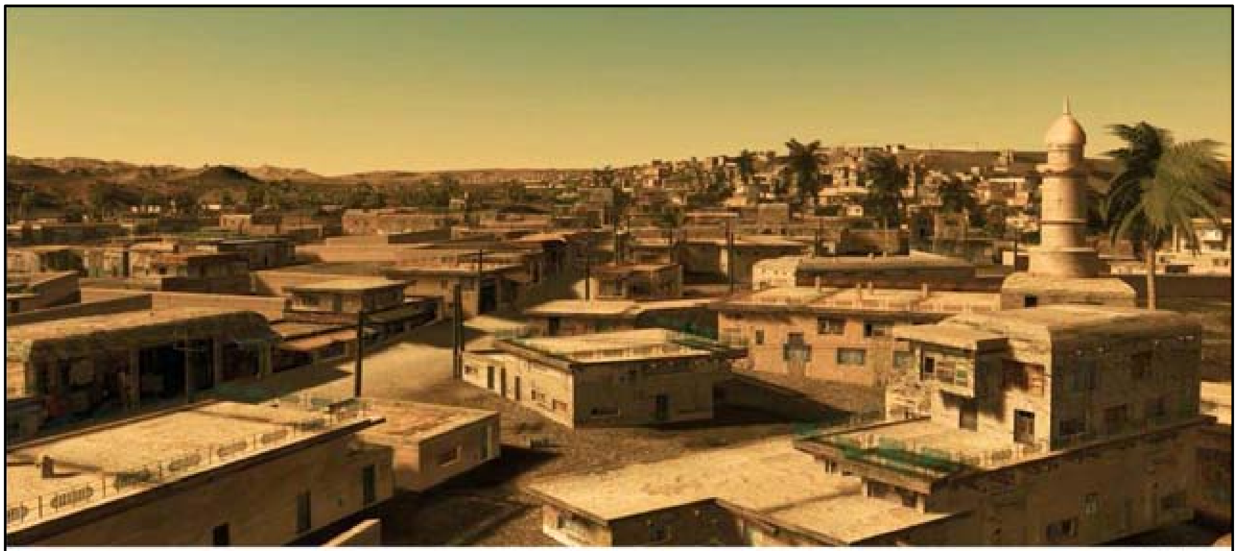
Wide application area for 3d urban models

- ranging from *virtual to real* enviroments
- at *small to city-wide* scales
- on *offline or realtime* rendering systems
- for *groundlevel or aerial* city views
- as *abstract to photorealistic* visualizations

Modeling Practices

Depend on application and production budget



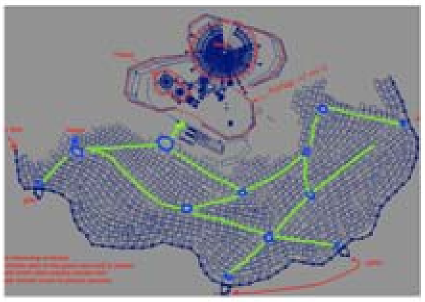


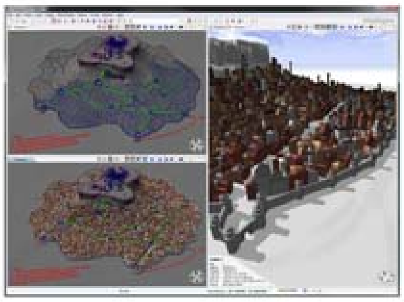
Procedural Urban Modeling in the Industry

CITY MODELING STRATEGIES

Designing Cities from Scratch

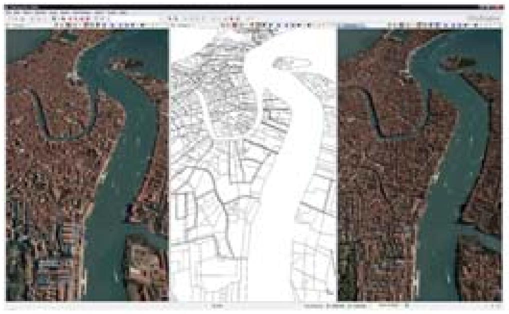
- Starting point is to sketch a streetnetwork
- Definition of land use areas & building types
- Design of height & density distributions
- Detailed modeling of streetnetwork





Cities based on Streetnetworks

- Real street data on OpenStreetMap.org
- Extract blocks from streetnetwork (using approx. street widths)
- Subdivide blocks into parcels



Often in productions a 3D model that only looks or feels like a real city is needed. These so-called 'atmospherically-correct' city models only mimic the architectural style of the building shapes and facades - but a building does not correspond exactly to a real-world opponent. Often productions do not have access to the corresponding building shapes/geometries nor do you want to buy the (expensive) building footprints.

As a consequence, a typical modeling strategy to reconstruct existing cities at low costs is to:

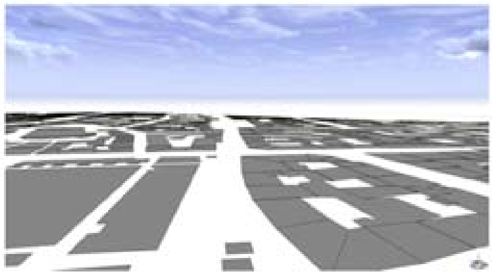
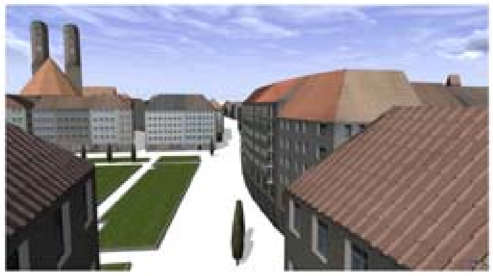
- Take street data from OpenStreetMap.org and a satellite picture.
- Generate blocks from the street data and procedurally subdivide the blocks into lots (to roughly the same size as on the satellite picture).
- Write simple rules which generate the shapes and texture the facades with photos of a few buildings common in such a city.
- Control the generation process via image maps to fine-tune e.g. landuse (open spaces or green spaces) to insert trees etc.
- Place manually landmarks at the right places.

Such a generation of approximate buildings in combination with the manual placement of landmarks makes it sufficent for bird view applications such as helicopter flight simulators.

The left picture shows the satellite picture on the left, the OpenStreetMap data in the center and the generated city on the right. The right picture shows a bird view of the model.

Cities based on Footprint Data

- Often with additional attributes (height, roof type, land use,...) - otherwise height & density distributions have to be modeled
- Procedural building generation using existing, approximate or random attributes

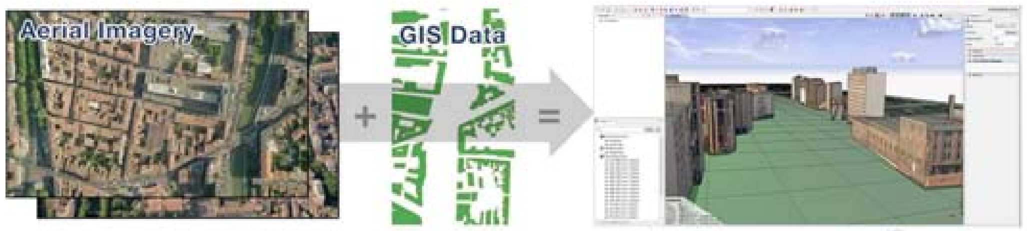
The pictures show the procedural modeling solution of GTA Geoinformatik GmbH based on CityEngine. Using GTA's detailed real-world GIS data (including façade attributes), buildings are generated in different levels of detail which can be adjusted interactively depending on the requirements. Buildings with high level of detail feature 3D facade assets as well as roof bricks. The resulting city scene consists of 3209 buildings in the center of Munich, Germany.

Existing Cities based on Photos

3d models from groundbased/aerial imagery:

- Covered in previous section of this course
- Not yet production ready

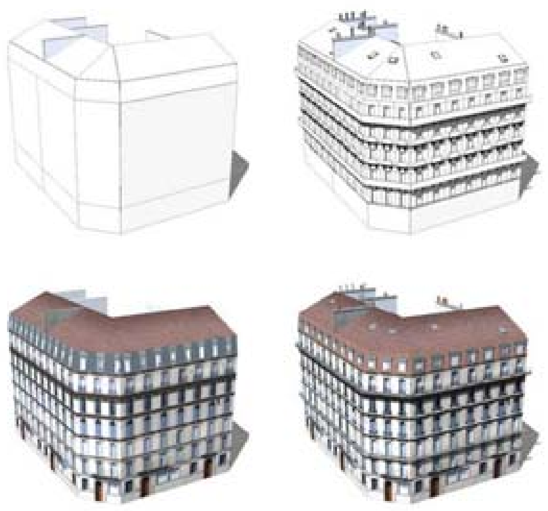
Combination of aerial imagery with GIS data:



V-City is a project supported by the European Commission and aims to research, develop and validate an innovative system integrating the latest advances in Computer Vision, 3D Modeling and Virtual Reality for the rapid and cost-effective reconstruction, visualization and exploitation of complete, large-scale and interactive urban environments. The focus of the project on urban environments is not only made possible by the latest technological advances, but also justified. Urban environments represent one of the most important and valuable cultural heritage as acknowledged by the UNESCO. More info on http://vcity.diginext.fr.

Levels of Detail 1/2

- Photo textures (realistic looks)

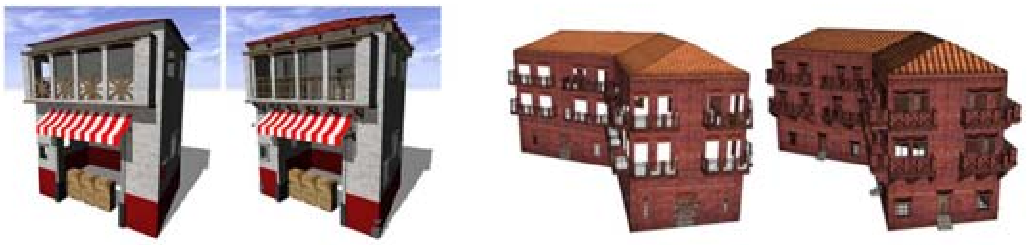


Generally, textures from photos lead to more realistic looks and should be used whenever possible. The figures depict possible level of detail strategies:

- *Top left:* For real cities, airborne 3d models come with low polygon counts and low resolution of textures (not suited for groundlevel views). However, since the textures cannot be reused, airborne city models require a large texture memory footprint.
- *Bottom left:* For pseudo-real cities, groundbased photos on simple geometry (=one quad-polygon per facade with UVs adjusted to the corresponding texture) can be used. A few dozens of recitfied facade textures are needed to texture a city i.e. the textures can be reused.
- *Right:* Groundbased photos aligned with more complex geometries generate realisitc models suited for groundlevel views.

Levels of Detail 2/2

- Asset library (rendering performance)



If no photos are available or (realtime) rendering performance is important, an asset library with prepared models and textures has to be applied. Based on the quality of assets detailed and realistic groundlevel views can be generated (see figures below)

The figures on top depict possible level of detail strategies using assets. 3d facade models can be easily simplified by:

- Making the facades 2d e.g. by not modeling three dimensional window openings
- Leaving out small assets completely in low levels of detail
- Preparing the assets in different level-of-detail representations i.e. the procedural generator selects which level-of-detail of the asset should be inserted

©2011 Designworks

Procedural Urban Modeling in the Industry

BUILDING PARAMETERIZATION

The following obstacles make the formalization and parameterization of building designs difficult:

- *Variations*: The appearance of buildings is often based on personal design choices of architects. They reuse parts of existing design patterns and change other elements according to their aesthetic perception. As a consequence, we encounter a lot of arbitrary design variations and exceptions, which make a general formalization or classification difficult.
- *Incompleteness*: Architectural sketches or descriptions typically only convey selected measurements of interest and do not cover elements of less importance. Another source of incompleteness is ruination: ancient buildings often have been destroyed or significantly damaged which limits the possibility to gather additional geometric data. Hence, the missing data has to be retrieved (often in a tedious task) from multiple sources.
- *Conflicts*: Architectural reference literature often has conflicting data. This makes it difficult to combine measurements from different sources. And unfortunately, the naming conventions often differ in the literature.

A good strategy to formally break an architectural design down comes from one of the most famous architects of the 20th century, Le Corbusier:

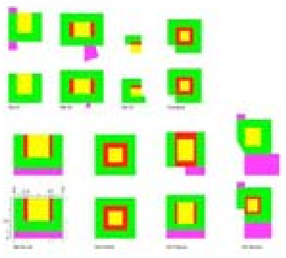
> *"Mass and surface are the elements by which architecture manifests itself. Mass and surface are determined by the plan. The plan is the generator." [Le Corbusier 1923, page 47]*

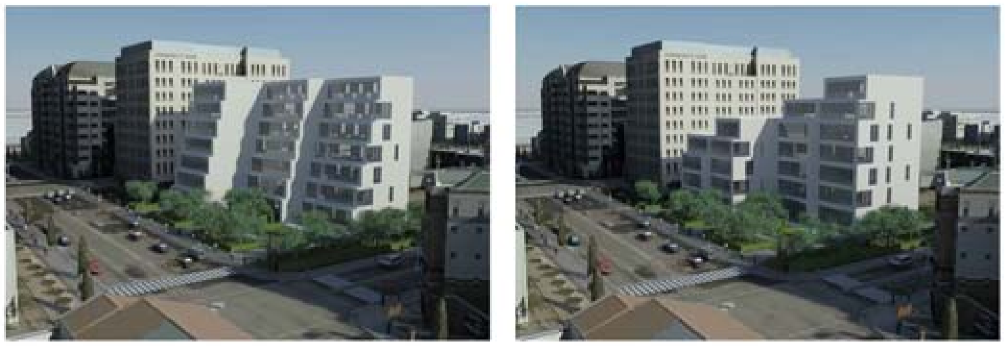
which means in the first place that usually an architectural design can be classified into mass and surface. General concepts for mass and surface modeling have been presented already in the previous sections of this course (e.g. mass models are usually constructed as an assembly of simple solids to model the basic L-, H-, U-, T- shaped building types, and surface modeling usually follows the general subdivision scheme (facade -> floor -> tile -> wall/window). In the following, we will present a few guidelines and examples.

Parcels

General Patterns & Zoning Laws

- Distance of building to street
- Distance inbetween buildings



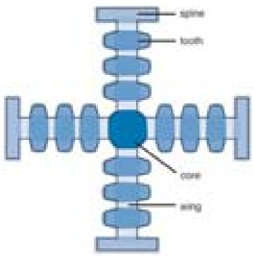


Top right: General Parcel Subdivision Scheme by José Duarte, TU Lisbon.

Bottom: Same procedural building style on same parcel with different procedual footprints guided by parameters such as distance to street or buildings.

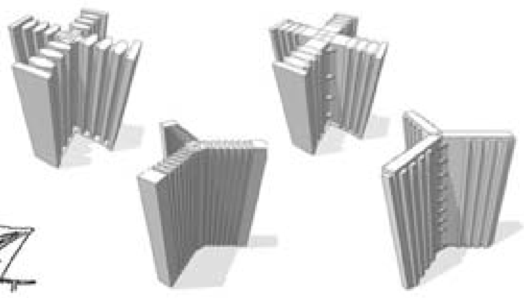
Mass Models 1/2

"Cruciform Skyscraper"
by Le Corbusier



Control attributes for volume:

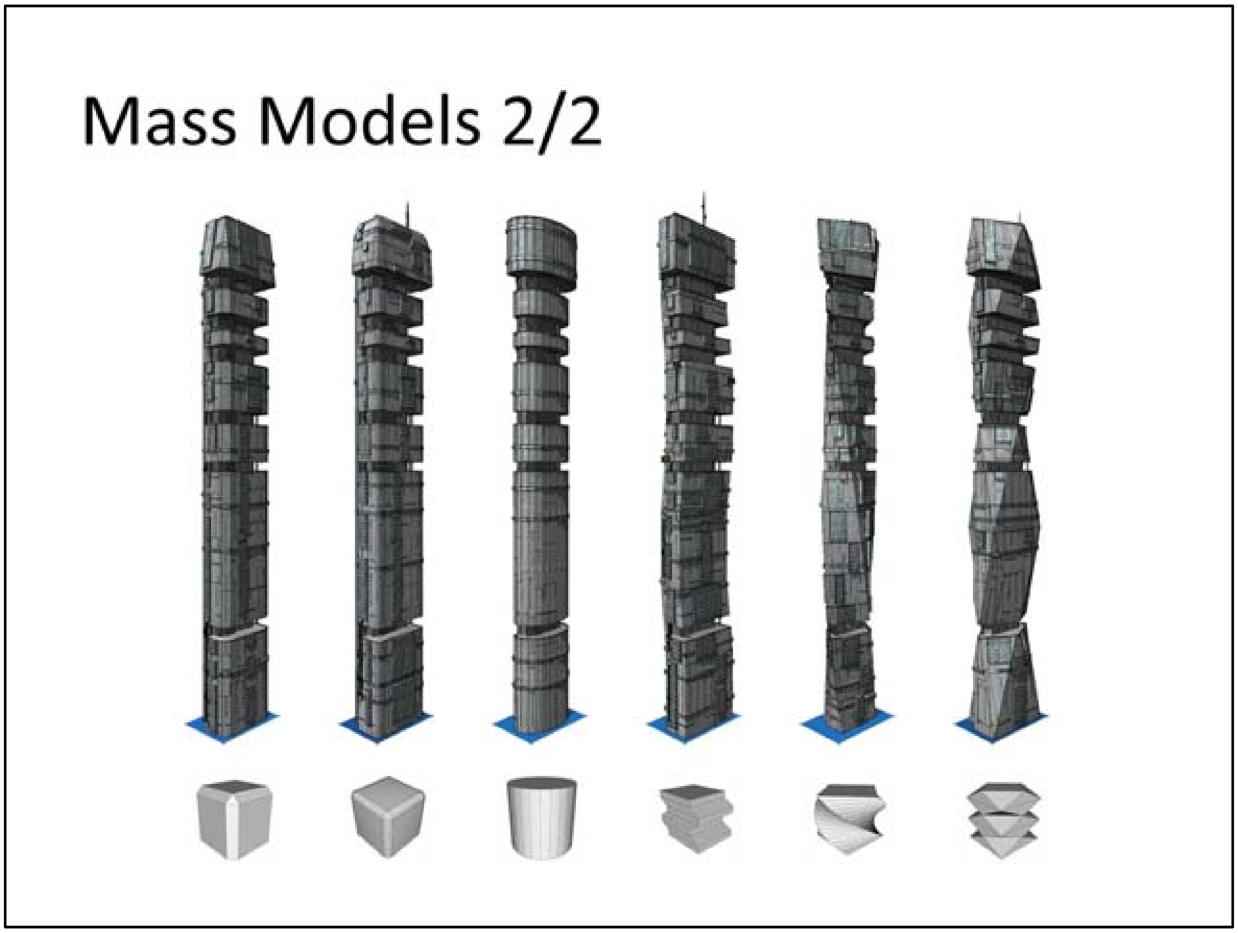
```
attr Building_height = 220
attr Wings_n = 4
attr Wing_length = 100
attr Wing_width = 16
attr Spine_width = 50
attr Tooth_projection = 10
attr Tooth_distance = 12
...
```



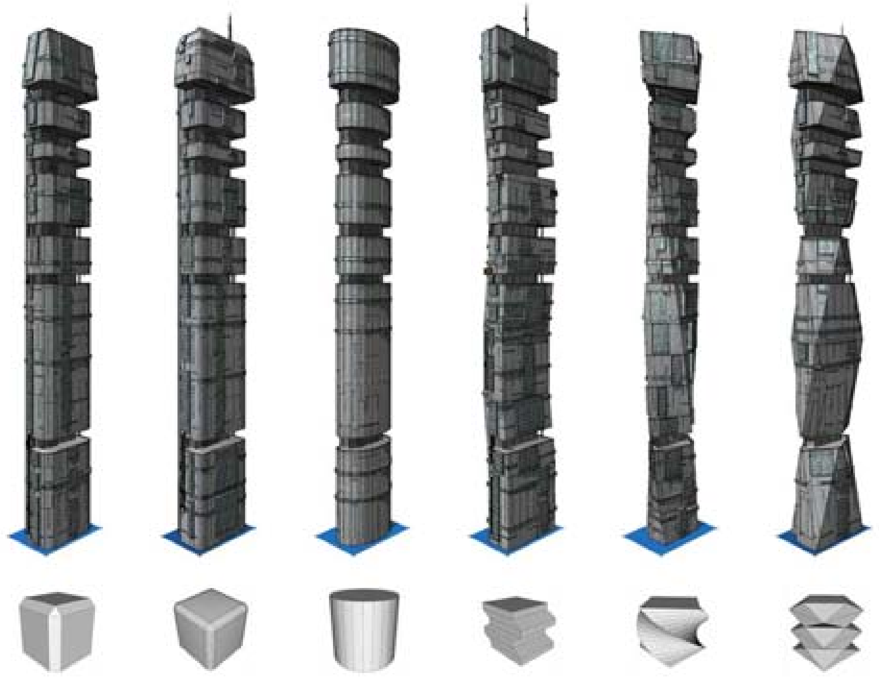
The Cruciform Skyscraper of Le Corbusier are taken as example. Le Corbusier designed, in great detail, several plans of different variations between 1920 and 1930. Most famous incarnations can be found in the master plans for his Contemporary City (1922) or the Plan Voisin (1925). The (back then) enormous sixty-storey cruciform skyscrapers are built on steel frames and encased in huge curtain walls of glass. They housed both offices and the flats of the most wealthy inhabitants of the cities. These skyscrapers were set within large, rectangular park-like green spaces.

In a first step, it is important to name individual elements of the design. In the example, we choose the names core, spine, wing, and tooth for the mass model (see figure 5.9). After the components have been identified, their spatial dependence has to be analyzed and formalized. Often it is clearly obvious how they are assembled, for instance a tooth element is repeated along the wing in our example.

In the second step, the important parameters of the design have to be identified. It is recommended to declare these parameters as control attributes in the rule set. The box shows the control attributes for the mass model of the Cruciform skyscraper. For example, the control attribute *building_height* is the overall height of the skyscraper and *Wings_n* defines how many wings are arranged around the core (note that there are paintings from Le Corbusier of the skyscraper design with only three wings). Afterwards, the length and width of the wing and spine element are defined and so on.



Mass Models 2/2

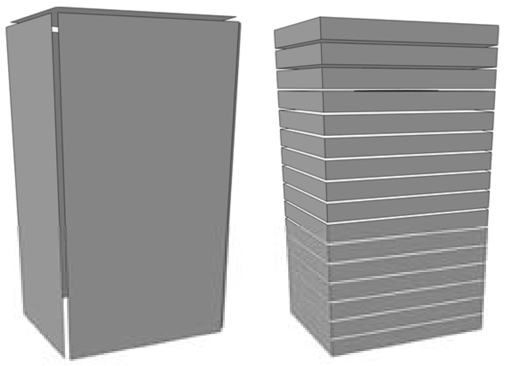


In some productions mass models are already available which then has to be refined using procedural techniques. The figure above illustrates a hybrid concept where a massing vocabulary has been modeled as simple polygonal assets (shown on the bottom row). These have then been inserted at the very beginning of the rule set. Note that the different volumes do not influence the overall building design e.g. the horizontal sections are the same - regardless of which volume is applied at the beginning.

Floors and Facades

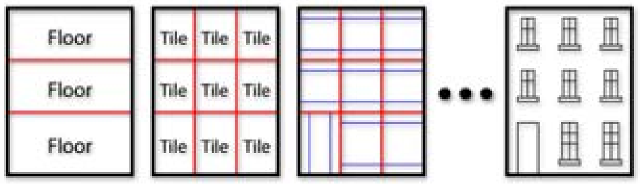
Two split strategies:

- Facade surfaces (simpler models)
- Floor volumes (required for interiors)



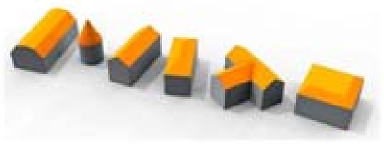
Refinement using subdivsion schemes

(covered in previous seciton of course)

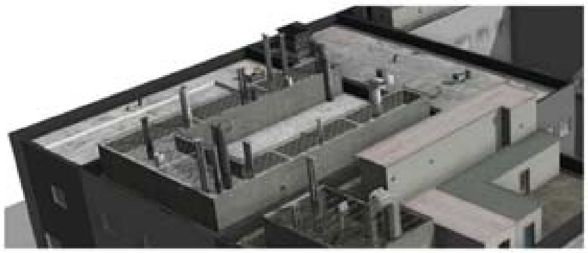


Roofs

- Hipped roofs



- Flat roofs



Hipped roofs have been classified (gable, hip, mansard etc) and can be parameterized accordingly. The resulting geometry can be easily extended with additional assets.

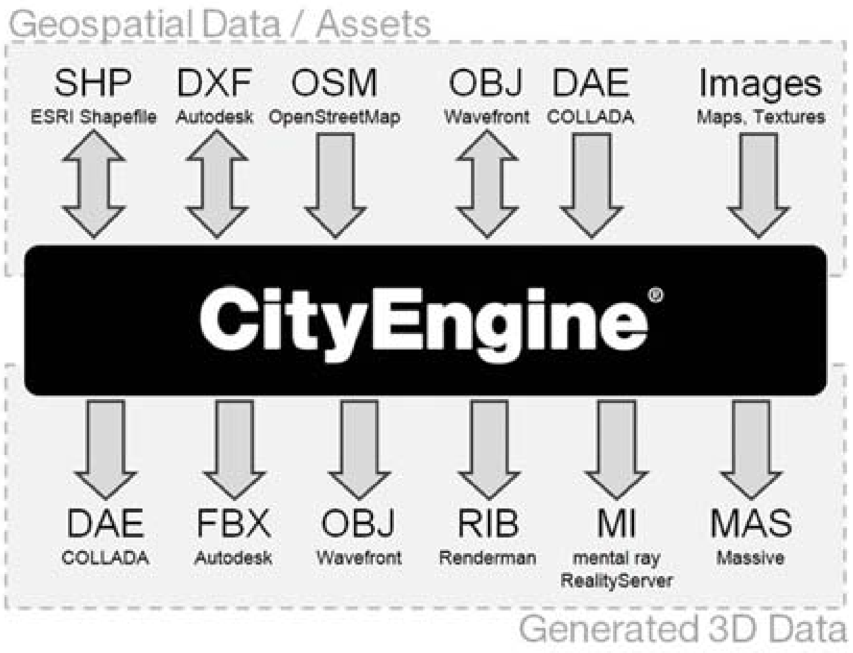
Modern flat roofs consists of a significant amount of different assets ranging from small pipes up to large vents and facility constructions.

Procedural Urban Modeling in the Industry

PRODUCTION PIPELINES

In the remaining sections different use case strategies are presented by describing applied pipelines. Although CityEngine is used as procedural content generation tool in the presented examples, the pipelines apply to other tools as well i.e. CityEngine could be substituted with other procedural generation solutions.

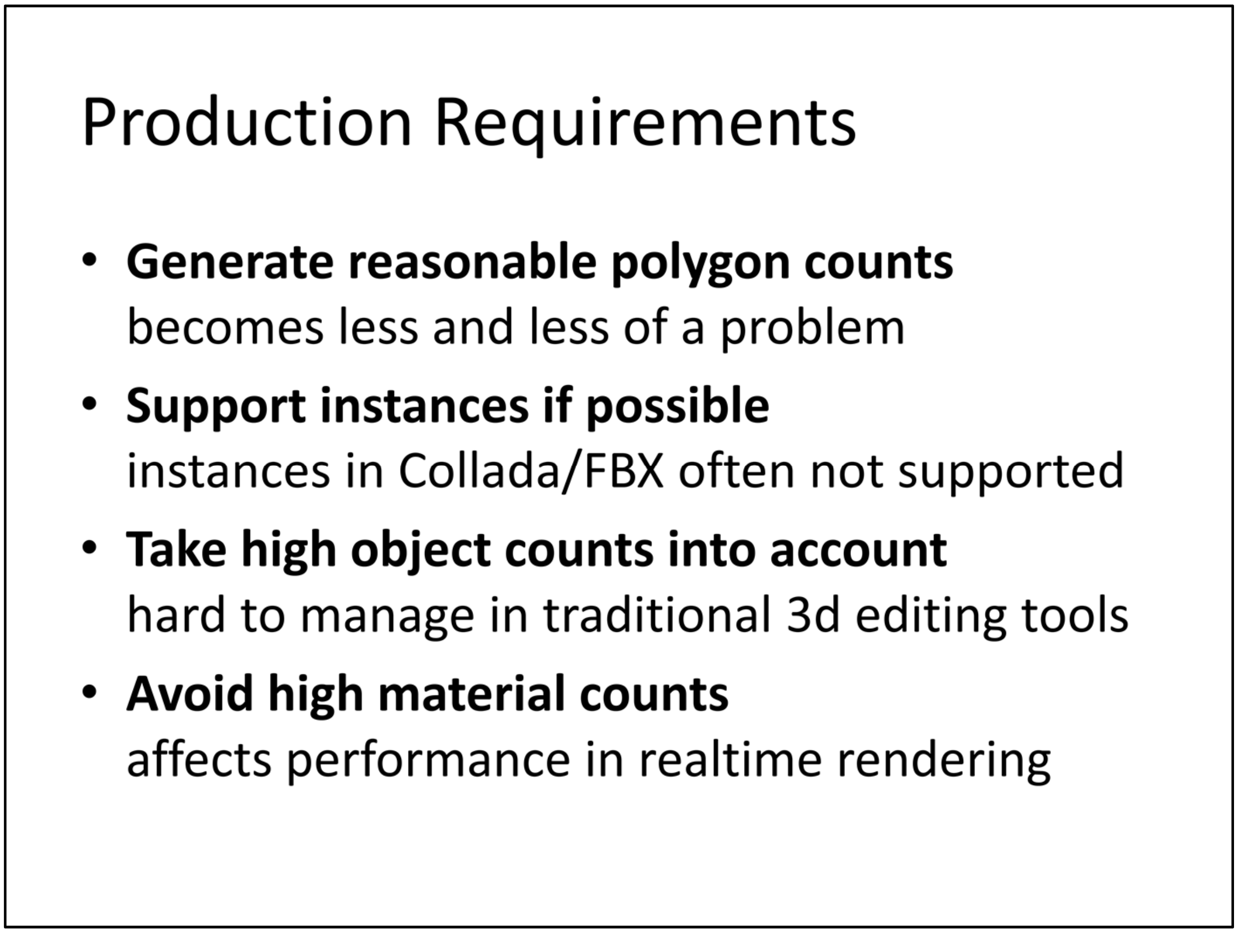
General Procedural Pipelines



The general pipeline of procedural geometry generators:

- Input: assets and geospatial data (if any)
- Output: 3d models generated procedurally by rules or code

The figure shows the import/export capabilities of CityEngine which includes the most common industry-standard GIS and 3d formats.



Production Requirements

- **Generate reasonable polygon counts**
  becomes less and less of a problem
- **Support instances if possible**
  instances in Collada/FBX often not supported
- **Take high object counts into account**
  hard to manage in traditional 3d editing tools
- **Avoid high material counts**
  affects performance in realtime rendering

High polygon counts become less and less of a problem due to today's graphics hardware.

Depending on the pipeline and end application, the usage of instances is strongly recommended. However, please note that it might conflict with point 3, high object counts. In most 3d editing tools, each instance is handled as an object (even if the geometry is shared). As a consequence, 100'000's of instances might affect heavily the performance of such tools and are hard to manage.

High object counts could be avoided by e.g. merging objects with same materials. Therefore, depending on the pipeline and end application (again), it is sometimes recommended to expand instances by 'freezing' their vertex coordinates and merging them into a few objects (as many as corresponding materials). However, as a consequence, local coordinate systems of the merged objects are lost.

High material counts have to be avoided generally. Procedural operations such as for example thousands of *set(material.diffuse.r,rand(0.0,1.0))* operation calls, where the red diffuse channel is set with a random number have to be avoided. Instead use discrete random values such as for example *set(material.diffuse.r,floor(rand(0.0,11.0))/10.0)* where only 10 different materials are generated.

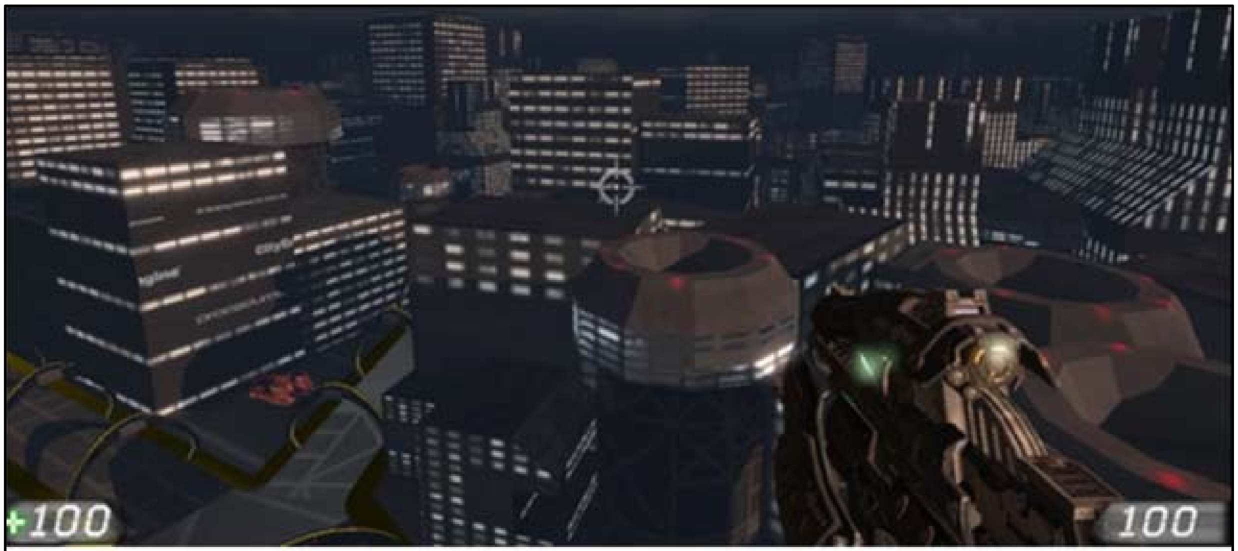
Procedurally Generated Materials

- Complex materials using different UV sets
- Material exchange capabilities often limited

Procedural modeling tools allow for the simple generation of complex material networks using different UV sets (texture coordinate layers on geometry). However, the material exchange capabilities via industry-standard formats are still limited and furthermore not many formats support multiple UV sets. In case complex material setups are generated, it is therefore recommended to export the output geometry in the native format of the targeted 3d application. If this is not possible, the widely popular Collada or FBX formats are recommended.

Collada and FBX support both a.o.:
- Common shading models
- Ambient, diffuse, specularity as scalar or image
- Reflectivity as scalar or image
- Bump mapping (not normal mapping)
- Opacity as scalar or image (some tools interpret it as transparency (=1-opacity))
- Multiple UV sets

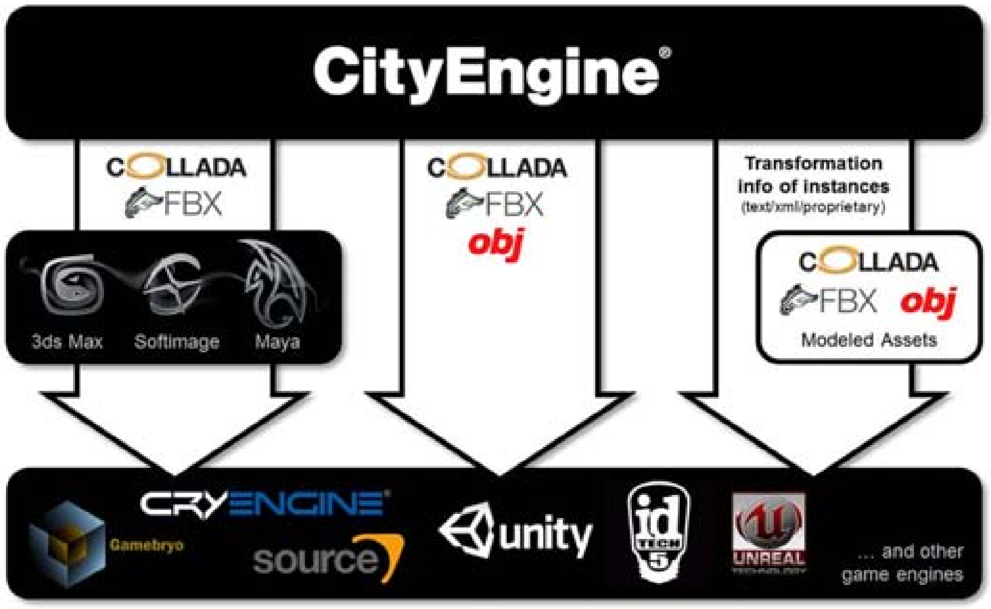
Note however, although specified, that not all tools read these attributes.



Procedural Urban Modeling in the Industry

GAME DEVELOPMENT

Game Development Pipelines



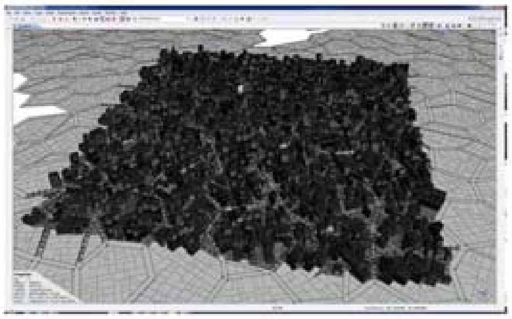
Procedural Content in Games

Different use cases for game engines:

- Procedural map generation
- Procedurally distributed asset instances
- Preprocessed procedurally generated geometry (by 3rd party tools or engine editor)
- Realtime procedurally generated geometry (usually hardcoded due to diff. requirements)

Example: Instancing Assets 1/2

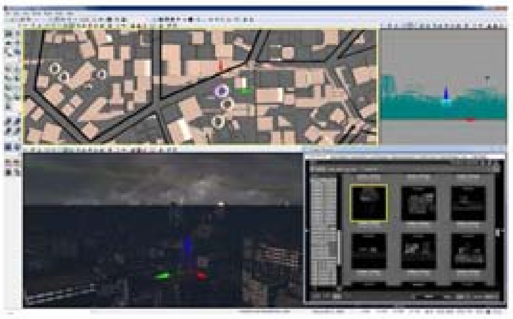
- Procedural creation of streets and positioning of buildings assets

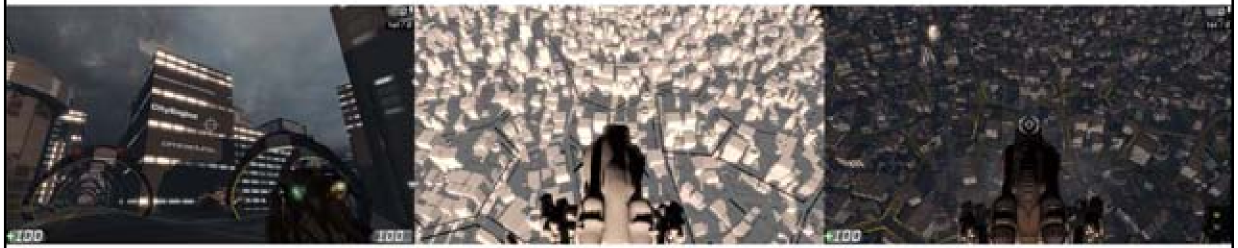


Note: To procedurally distribute and position prepared model assets, their ratios and scales have to be taken into account. Therefore, the procedural generation consists mainly of a selection mechanisms deciding which asset to place on a given parcel and how much it can be scaled (if any).

Example: Instancing Assets 2/2

- Import assets & other elements as geometry
- Import positioning info of assets into engine





Epic's Unreal Engine is one of the most popular and successful game engines. Epic’s Unreal Development Kit (UDK), an free easy-to-use integrated graphic environment for authoring games or interactive architectural content, is capable of creating standalone applications/games.

Instanced assets can be export from CityEngine to UDK via Unreal's t3d format using the Python-based exporter. The t3d format is a simple text-based format where the transformations (position, rotation, scale) of the assets is described.

A possible production pipeline from CityEngine to UDK could consist of the following steps:

1. Prepare your own set of textured building assets and street elements
2. In CityEngine, generate a city scene using these assets
3. Define report variables to your CGA file that store necessary information such as asset ID, position, scale and rotation
4. Define additional render-specific attributes such as shadow, collision or light parameters
5. Prepare an exporter Python script that generates a t3d file reading the reported values.
6. Export from CityEngine with the python export script enabled
7. In the Content Browser of the Unreal Editor, prepare static meshes from the building assets.
8. In Unreal Editor, import the generated t3d file into your scene.

With such a pipeline, procedural city scenes can quickly be imported into UDK which allows to iteratively design and test different versions of the city layout. And besides instanced geometry, additional Unreal-specific metadata such as collision or lighting attributes can be defined arbitrarily and are generated and exported per asset instance in the same step.

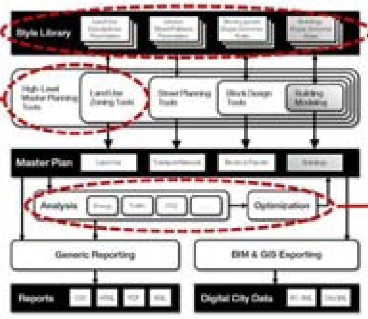
©2010 ETH Zurich

Procedural Urban Modeling in the Industry

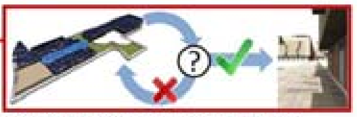
URBAN DESIGN

Urban Design Pipelines

Touch-based, interactive high-level interfaces for collaborative planning

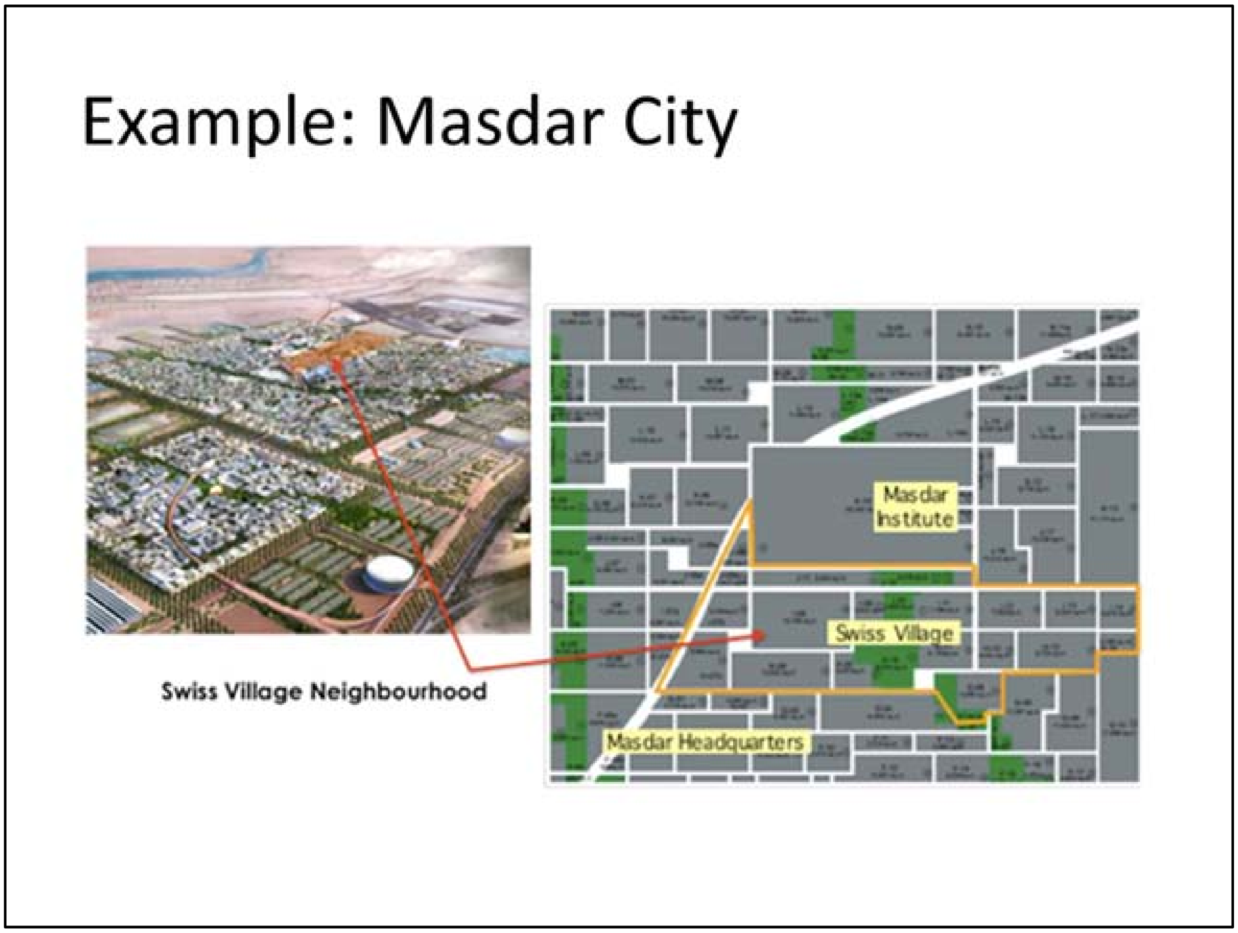


Library of generic urban patterns (incl. metadata and environmental context)


Procedural representation enables optimization to constraints

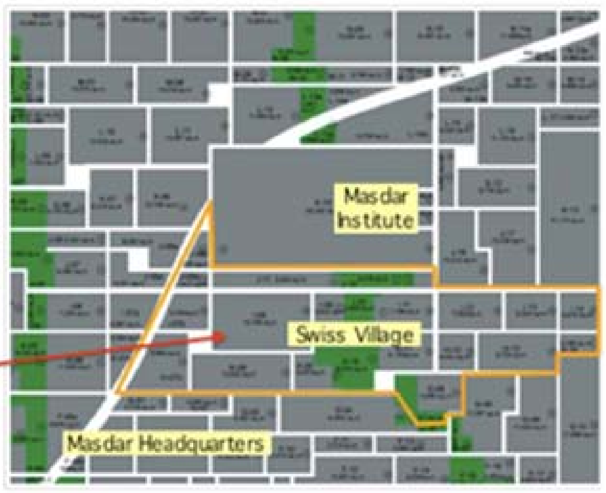
Urban patterns (=library of rules) serve as knowledge basis and allow for iterative design workflows:

- Simulation and evaluation of generated design
- Parameter optimization of procedural model



Example: Masdar City

Swiss Village Neighbourhood

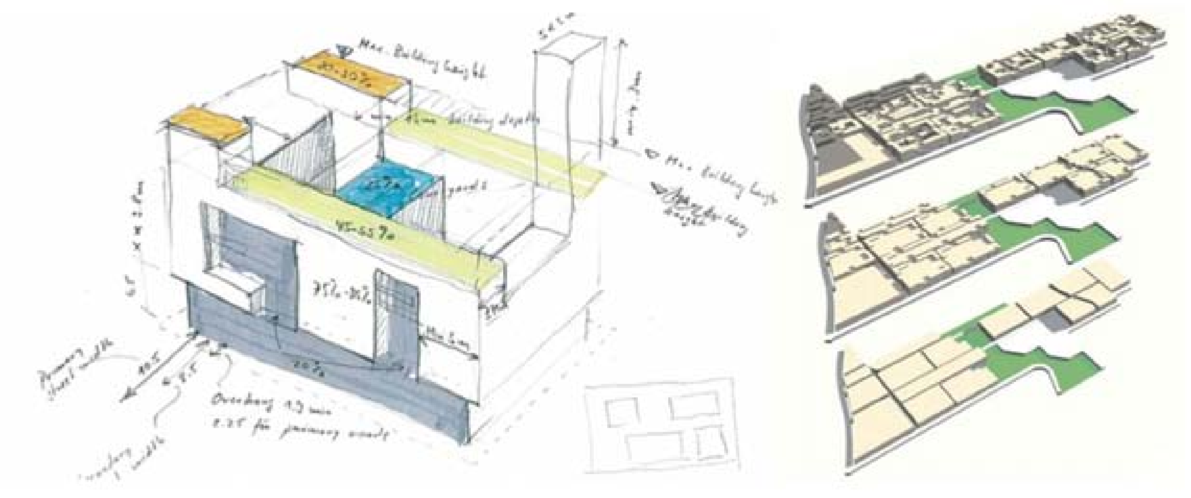


The ambitious Masdar City will rely entirely on solar energy and other renewable energy sources, with a sustainable, zero-carbon, zero-waste ecology. The city is being constructed 17 kilometers south-east of the city of Abu Dhabi. The master plan for the whole city (6km2) was created by Foster + Partners, combining traditional Arabic city building principles, combined with current and future technologies.

In the following, the applied procedural design principles for the planning of Masdar's quarter 'Swiss Village' are described. The Swiss Village will be a distinct neighborhood within Masdar City, which will integrate and serve as the home of Swiss companies with expertise in clean technology (210'000m2 of built up area).

Building the Parametric Rule Base

- Encoding the volumetric principles which are (narratively) defined in the master plan

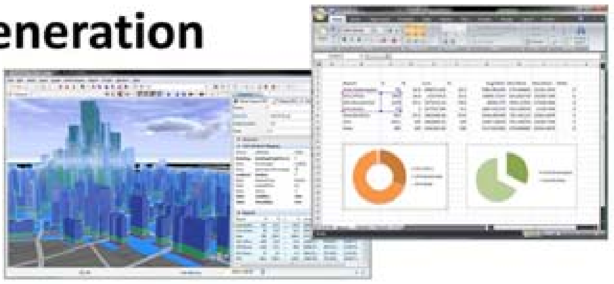


The development of the architectural gestalt of the Swiss Village is being researched at the Chair For Information Architecture at the ETH Zurich. The team around Prof Dr. Gerhard Schmitt developed an architectural grammar which incorporated the main volumetric principles defined by Foster & Partners.

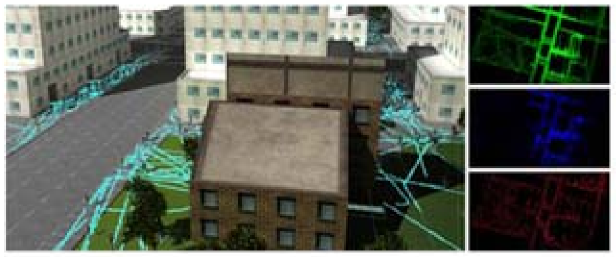
Once the main volumetric language was found, the design has been encoded as CGA Shape rules inside CityEngine and the buildings were automatically generated accordingly. The rule-based top-down modeling of CityEngine inherently maintains a consecutive order in the levels of detail of the created building shapes.

Simulation and Evaluation 1/2

- **Rule-based report generation**
  Automatically calculate quantities such as GFA, FAR, or material usage



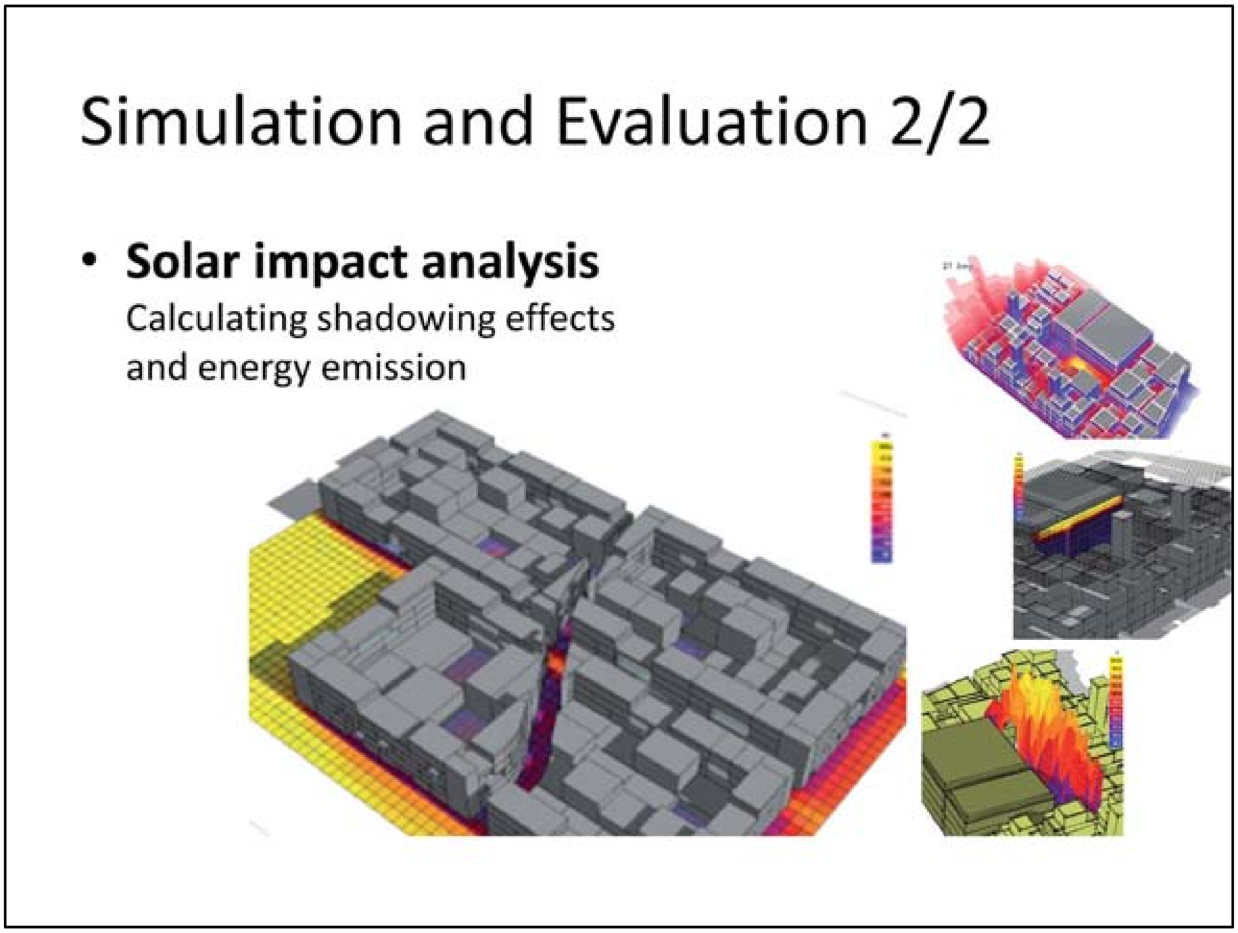
- **Agent-based simulation**
  Analysis, prediction and visualization of occupant behavior in urban spaces



To statistically evaluate the quality of the designs, rule-based reports have been generated using CityEngine e.g. the impact of changes in the mass model parameters to the gross floor area has been measured and accordingly the parametric mass model has been optimized to fit the targeted needs.

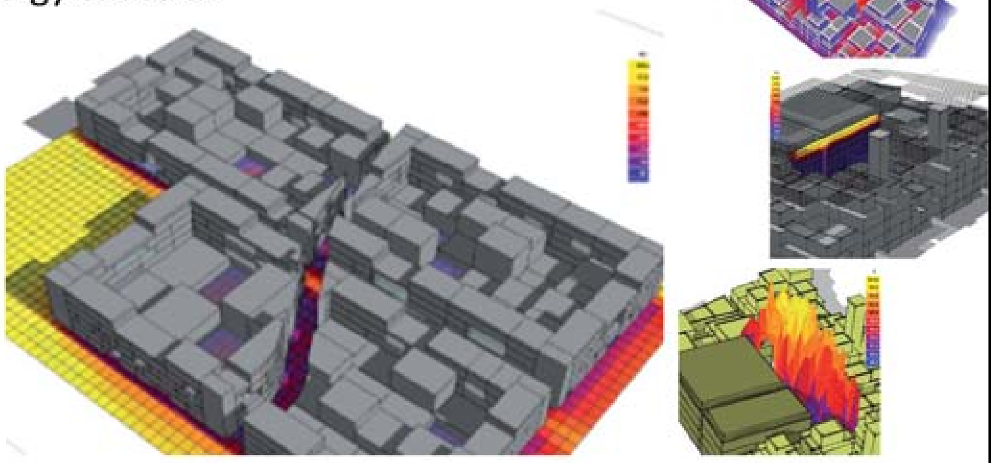
Furthermore, agent-based simulation methods have been applied to study the behavior of (note that the figure does not show the Masdar model). More information can be found in:

> *Aschwanden G, Haegler S, Halatsch J, Jeker R, Schmitt G and Van Gool L. Evaluation of 3D City Models Using Automatic Placed Urban Agents. CONVR Conference Proceedings, University of Sydney, 5-6 November 2009.*



Simulation and Evaluation 2/2

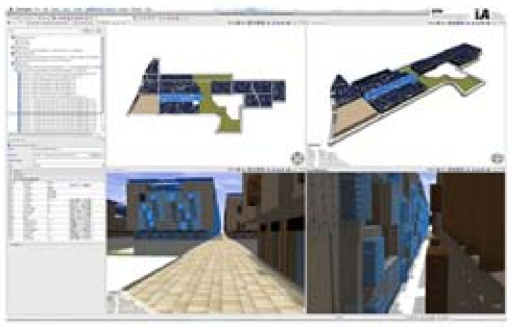
- **Solar impact analysis**
  Calculating shadowing effects and energy emission

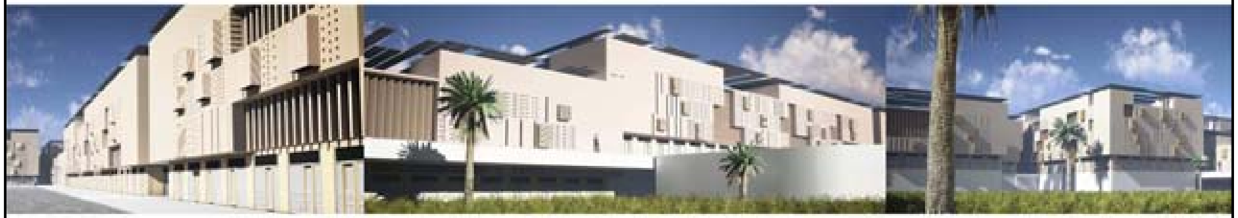


Solar impact analysis can be done in various commercial tools, in this example Autodesk's Ecotect has been applied.

Architectural Visualization

- Encoding detailed facade patterns
- Photorealistic rendering of final models





After the final parameters for the volumetric design have been determined, the basic facade patterns have been encoded as rules as well and offline renderings have been created in Vue from Eon Software.

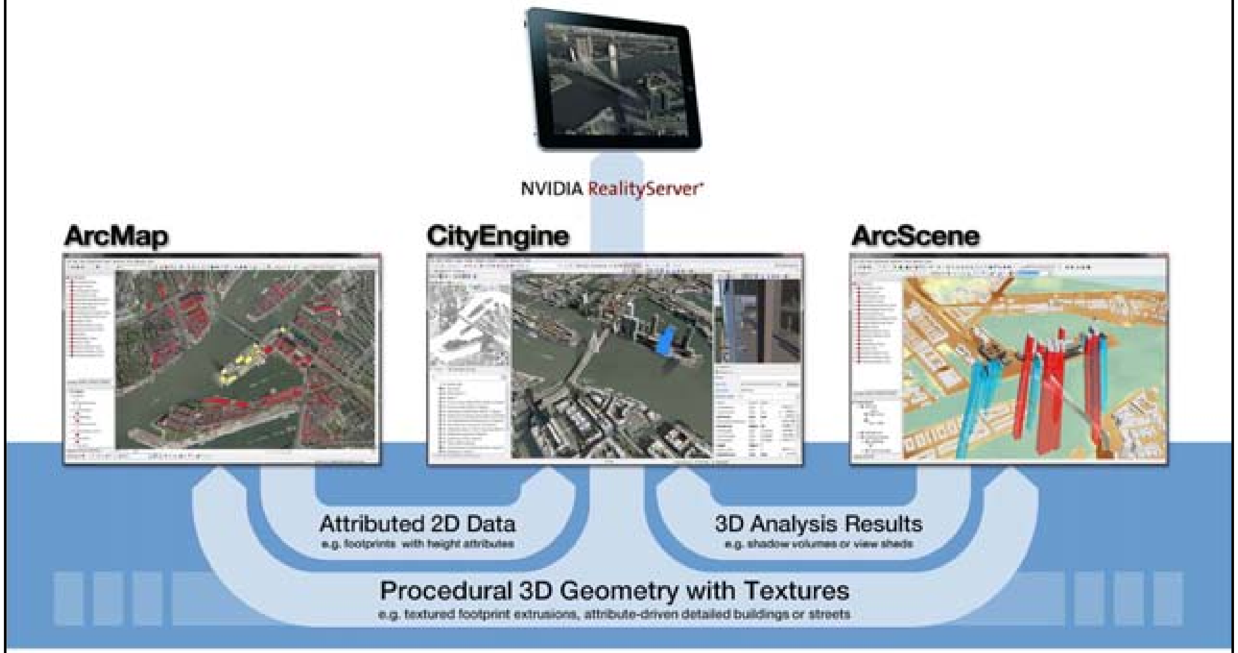
Note that it is important in such architectural visualizations (where only early concepts are communicated) to produce different design alternatives. Otherwise the client gets the impression of seeing a visualization of the final construction.

©2010 Nvidia

Procedural Urban Modeling in the Industry

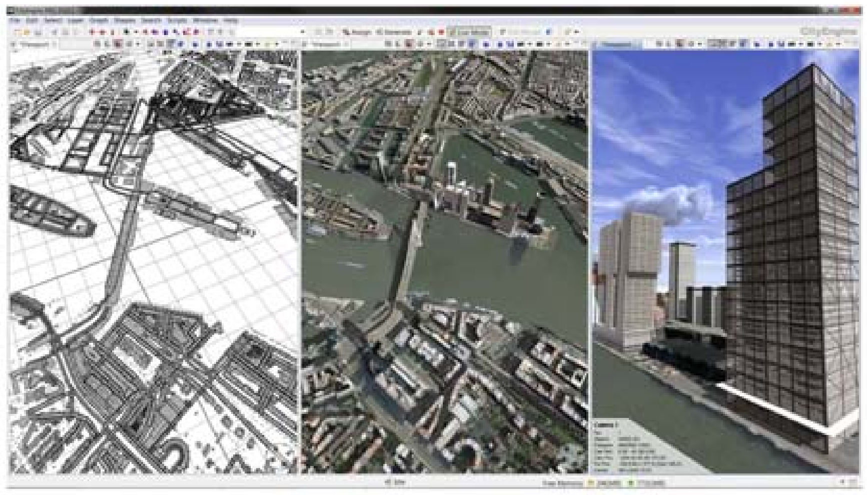
URBAN PLANNING

Urban Planning Pipelines


NVIDIA RealityServer
ArcMap
CityEngine
ArcScene
Attributed 2D Data
e.g. footprints with height attributes
3D Analysis Results
e.g. shadow volumes or view sheds
Procedural 3D Geometry with Textures
e.g. textured footprint extrusions, attribute-driven detailed buildings or streets

Example: City of Rotterdam

- Model aggregation from various 2d and 3d sources using procedural techniques



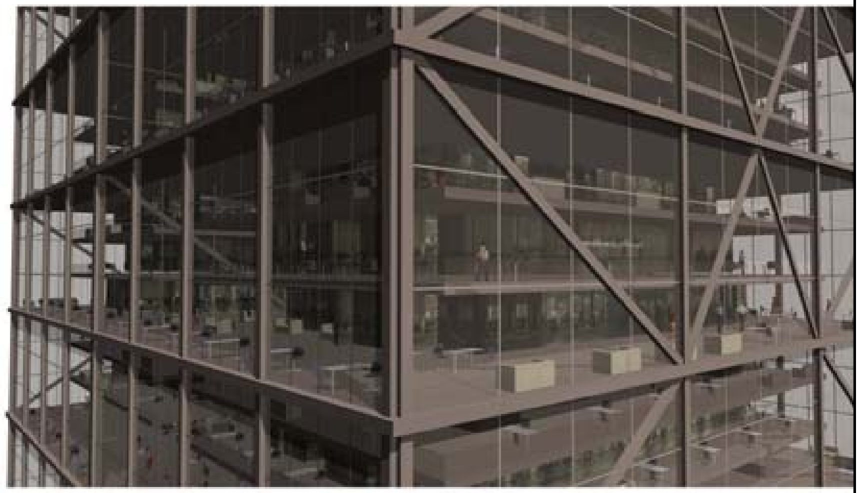
In this urban planning example, traditional 3D and procedural models as well as geo-referenced GIS data are aggregated into a detailed model of the city of Rotterdam. The CityEngine screenshot shows on the left the given GIS data and in the middle the whole city model with reconstructed 3D buildings as well as procedural streets, trees, and buildings. On the right, the new developments are depicted.

This example showcases a typical urban planning example where various sources of data at different representations and quality are available which have to be combined using procedural techniques. The following layers have been included into the city model in CityEngine:

- Traditional 3D layer: Landmarks and building models with oblique textures (Collada files) are available in the center of the city and are placed in the scene according to their geo-referenced footprints.
- Building extrusions: Based on given building footprints with height attributes, a ruleset is used to extrude the buildings of the outer area. Generic random texturing of the facades is applied for a more realistic look.
- Vegetation: A ruleset selects and inserts corresponding tree assets on the given real-world locations. Each feature point consists of several attributes such as the type and age of the tree. Since the height of a tree is not provided, the ruleset has to compute the height according to its age and type.
- Procedural streets: Street center lines are used as base for the generation of procedural street geometry with the CGA shape grammar. Depending on the needed level of detail, additional street elements such as vehicles, pedestrians, street lamps, traffic lights and other street furniture can be arbitrarily enabled or disabled for specific areas using control attributes.

Procedural New Developments

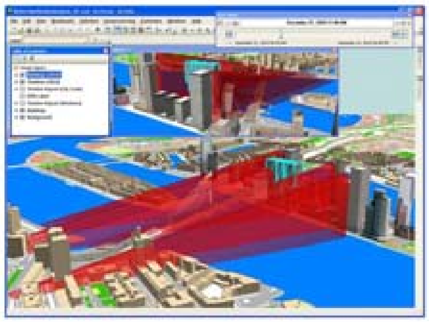
- Encoding designs of new buildings for analysis and visualisation studies



On the Wilhelminapier, the peninsula in the center of Rotterdam new buildings are constructed. These new developments have been designed by magnificent architects Rem Koolhaas, Norman Foster, or Alvaro Siza. From these building designs, procedural models have been created with the CGA shape grammar. They consist of four levels of detail: extrusions, textured facades, detailed facades and floor plans, and detailed interior with tables, sofas, people etc. To model the interiors, the rule set first generates the building core and subdivides afterwards the floor plan into rooms and open areas. In the next step, the CGA shape grammar is applied to distribute furniture assets along the generated walls and place human assets between sofas and chairs and other furniture. A close up of the resulting building model is shown in the figure.

Analysis and Visualization

- Shadow analysis in GIS
- Cloud-based rendering in web browser for visualization



Afterwards, shadow volumes of existing and planned buildings have been calculated with ArcGIS 3D Analyst and exported for visualization.

The resulting city model has been exported to mental images' RealityServer which renders the scene at interactive rates using GPUs in the cloud. Such cloud-based rendering solutions allow for a simple communication of the new developments to citizens using standard browser technology.