CEIG – Spanish Computer Graphics Conference (2019)
A. Jarabo and D. Casas (Editors)

A Virtual Reality Front-end for City Modeling

E.Rando[1], C.Andújar[2] and G.Patow[1]
[1] ViRVIG, IMAE, Universitat de Girona, Spain
[2] ViRVIG, Computer Science Department, Universitat Politecnica de Catalunya, Spain

Abstract

Current tools for city modeling, including those supporting procedural techniques, have a steep learning curve and require substantial user input and/or skills to create realistic 3D models of cities. In this paper we propose a VR tool for the fast and intuitive creation of 3D models of cities through their main elements (buildings, blocks and streets). The key ingredients of our approach are: (a) intuitive creation of mass volume models for buildings, whose facades can be refined later on through procedural rules, (b) the ability to use arbitrary urban layouts, either created from scratch within the tool or imported from public map services, (c) algorithms to replicate and transfer user-generated blocks to arbitrary block shapes, so that a few template blocks suffice to cover the whole urban layout. The major benefit of our approach is that city design and inspection tasks are done simultaneously in a completely immersive environment.

CCS Concepts

•***Computing methodologies*** → ***Rendering;***

1. Introduction

One of the main challenges in Computer Graphics and interactive applications is content creation, which includes the generation of realistic and detailed geometry. On the other hand, end users demand increasingly interactive, user-friendly applications to allow a broader range of public to generate new content. The traditional approach is to let artists manually generate geometry with tools like Autodesk Maya and 3DS Max, which is a tedious, repetitive and time-consuming process, although it provides full control. In the last decade, grammar-based procedural modelling has emerged as a powerful technique for generating architectural geometry [WWSR03, MWH*06]. However, current procedural modelling systems are mostly text based and therefore impractical for the intended users, i.e., artists and technical artists. Recently, there have been several efforts to generate more user-friendly interfaces like visual editing of rules [LWW08, Pat12], even with some focus on sketching models [NGDA*16].

However, none of these works result in a truly intuitive user interface as they still require the user to think in terms of rules or to perform a tedious processing after the initial model is drafted. This results in a non-intuitive interface that deter non-experienced users, like art historians, architects or urban experts, from using these new tools, in spite of their tremendous potential. On the other hand, Virtual Reality (VR) simulations have long been proposed to explore both yet-to-built buildings. Unfortunately, most VR architectural applications only address the exploration of models created with external tools, and offer limited or no editing options [ABB*18]. Users of these applications thus have to create the designs with one desktop tool and switch to a VR system to explore the result. The major consequence is that VR benefits in terms of intuitive manipulation, view control and navigation (head-tracking, natural walking) and model understanding (proportion, scale, order) only apply to the exploration part of the design cycle.

Figure 1: *Our system allows inexperienced users to create complex cities in a few minutes. Users can design immersively both the street network and the overall shape and layout of buildings. Further detail is added through a procedural rule system.*

In this paper we present a new immersive system for the visual modeling of buildings, street blocks and whole urban layouts. See Figure 1. The main contributions of this paper are: An immersive technique based on VR tools for the intuitive generation of mass model volumes. An automatic mechanism to copy and adapt a block layout to any required block shape. A mechanism for filling a whole urban model with the created block layouts. The generated city model can be iteratively created and explored without leaving the environment.

DOI: 10.2312/ceig.20191210

2. Previous Work

Procedural urban modeling started with the seminal works by Wonka et al. [WWSR03] and Müller et al. [MWH*06] which resulted in some professional software packages like Esri's City Engine. Some improvements to the expressiveness and ease of use of CGA grammars were recently introduced [SM15, JCS16]. Some authors [Pat12] introduced a visual-based language to ease a little bit the burden of using a text-file ruleset editing paradigm. Our approach, instead, directly allows the user to model a building without any ruleset (either textual or visual) editing. Barroso et al. [BBP13] proposed a copy & paste system for procedural buildings. Recently, Nishida et al. [NGDA*16] presented a system using user-provided sketches and a machine-learning algorithm to match procedural snippets to the user sketch. Guerrero et al. [GJWW15] introduced systems for learning and propagating user-defined edit operations. We recommend the excellent surveys by Watson et al. [WMV*08], Vanegas et al. [VAW*10], and Smellik et al. [STBB14] for an in depth discussion of the techniques in the field.

VR has shown to offer more natural interactions between the designer and the virtual environment, thus facilitating concept design creation and evaluation [SBS15]. Westerdahl et al. [WSW*06] found the VR model gave a fairly accurate representation of the real environment. Kuliga et al. [KTDH15] found very few experiential differences between the real and the virtual environment, that were related to atmospherics (warmth, attractiveness), detailing (e.g. missing furniture) and perception of distances/sizes.

3. Overview

Figure 2 shows an overview of the system. The *block creation* module allows users to create buildings and building blocks by providing their overall shape and layout. At this point, users are not required to provide roof/facade details; these will be added later on via procedural modeling. The *map creation* module allows users to define a street network. The network can be designed from scratch, or by importing an existing map from a public map repository (e.g., OpenStreetMap). Again, users only need to provide a minimalist input (street nodes and links); street details will be added through rules. The *city creation* module fills the map with variations of the user-provided blocks. The user can then generate a detailed city model by applying procedural rules to streets and buildings. In our current prototype, this is accomplished by sending the street network and the building shapes to an external procedural modeling tool, (we tested both CityEngine and Houdini), which automatically computes the detailed 3D model and sends it back to the VR application for inspection (through FBX export).

4. Block Creation

We define a block as a group of buildings bounded by user-defined streets. To create a block, the user defines the mass model of each building in the block. The mass model represents the general shape and size of a building. Our VR-system provides a set of tools to generate mass models and store them for later reuse.

A building is composed by simple volumes, that will be referred to as *subparts*, in order to support roofs at different heights. The

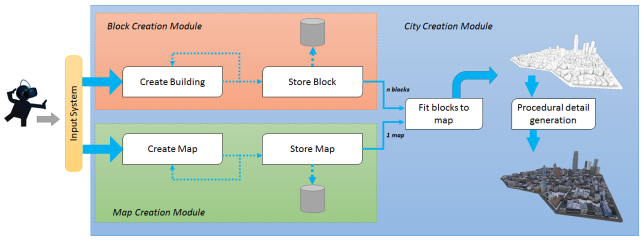

Figure 2: *Overall workflow of our system.*

Figure 3: *Moving, scaling and adjusting height of a subpart.*

creation of a building thus involves creating its subparts: the user starts by creating the shape of a subpart (a polygonal shape), and then adds more subparts. We now describe the interaction techniques for our current HTC Vive-based prototype (Figure 3):

- Moving a subpart by selecting it and specifying its new location. Our current interaction mapping for this task simply either use a 1:1 mapping of the hand-held HTC Vive controller position, or a virtual pointer to indicate the target location.
- Rotating a subpart, by selecting it and specifying a rotation around the vertical axis. Our current version requires the user to draw a circle arc with her hand or on the controller touch-pad.
- Scaling a subpart, by selecting it and specifying three scale factors along the application coordinate axes. The user defines the scale 3D vector using both hands (the roughly static hand defines the starting point, and the other one the direction and length of the scaling vector), similar to the pinch-to-zoom gesture.
- Assigning a height is a separate case of scaling, both for simplifying the gestures needed, and for smoothing the learning curve of the application. The user raises up the controller to increase the selected subpart height.

Using multiple subparts is useful to represent complex mass models, but may add overlapped surfaces and redundant faces that difficult to manage it as a single unit. A cleaning operation is also required for proper assignment and behaviour of the procedural rules, before merging all subparts into a single water-tight solid.

5. Block parameterization

Users can repeat the building creation procedure described above to create the buildings forming a block. These user-designed blocks will need to be transferred to arbitrary *circuits* defined by the street network. This way users only need to design a few *template blocks*, that will be transferred to fill hundreds of circuits defined by the street network. We thus need to encode blocks so that their shape can be re-used and transferred to fit different shapes.

For each vertex in the block, we compute its projection over the ground-plane and compute its barycentric coordinates with respect

to the convex hull of the block, see Figure 4. We use Mean Value Coordinates [HF06] which behave correctly in arbitrary polygons.

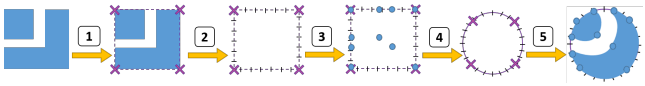

Figure 4: *Codification of an input block (in the figure, composed of two blue building footprints) by using barycentric coordinates: 1: Compute the footprint convex hull (CH); crosses in the figure represent the corners of the CH, 2: Partition the CH into N segments, 3: Compute barycentric coordinates of each block vertex (blue dots), 4: Parameterize segmented CH onto a circle, 5: Cast to Euclidean Coordinates on the circle and obtain the final, deformed outlines (blue areas in the figure).*

We segment the boundary of the convex hull into a fixed number of segments to facilitate block transfer. A refined convex hull also results in more accurate shape morphing when using the barycentric coordinates, and make these coordinates less sensitive to the convex hull complexity. Finally, we re-parameterize the (segmented) convex hull onto a circle (using a simple arc-length parameterization) and compute, with the barycentric coordinates, the new block vertices inside the circle. See Figure 4.

6. Map Creation

Once a few template blocks have been designed, the user can provide a map for the city representing the street network and (implicitly) the circuits that will accommodate the template building blocks. The street network is represented by a graph where the nodes represent intersections between streets (joints) and the links represent the street segments connecting them. In order to create the streets, users have to add intersection points and link them. Alternatively, a map can be defined by importing it from a public map service (e.g. OSM files from OpenStreetMap).

7. City Creation

At this point the user has designed a small collection of n template blocks, and a street network defining m circuits; typically $m >> n$ since the creation of block mass models is more involved than designing/importing the road network. The generation of a city consists of filling the circuits detected on the city map with varying instances of the template blocks. Recall that we store the blocks using their own barycentric coordinates. Transferring the mass models of a block to a circuit involves inverting this process: given a target circuit, we segment it, parameterize it onto a circle, compute its barycentric coordinates and, using the barycentric coordinates and the segmentation of the circuit, compute the final block vertices.

We avoid a synthetic look on our cities by preventing neighboring circuits from re-using the same template block. This is analogous to the classic problem in Graph Theory of coloring a map with a finite set of colors. So, we use a paint algorithm to prevent the appearance of homogeneous areas where one block predominates over the rest. This optional step assumes that the user has provided at least four template blocks, as it is demonstrated on the four color theorem for graph coloring [AH77].

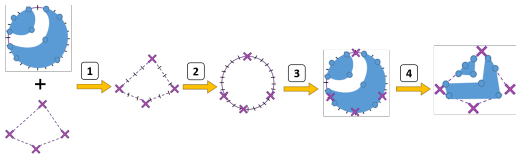

Figure 5: *Transferring a template block I to a map circuit: 1: Partition into segments, 2: Parameterization onto circle, 3: Compute Barycentric Coordinates of I, 4: Cast to Euclidean Coordinates.*

Once a city has been designed, the user can select among a predefined set of procedural rules to add realistic detail to the buildings. We use an external tool (CityEngine, Houdini) to refine streets, roofs and facades. The user can also decide to fill some circuits with procedural rules, besides using the template buildings.

8. Results

We implemented the described system as C# scripts for Unity3D Engine and python scripts for CityEngine and Houdini. Figure 6 shows some screenshots of the user interface. The virtual working environment reproduces a minimalist architecture studio with a virtual table allowing the inspection of a miniature replica of the current city model. Mass models are created on a grid-decorated virtual ground plane. The street network is designed in a virtual vertical wall. After creating template blocks and the street network, the system fills the map circuits with the templates and the resulting city can be explored on the virtual table. The user can then trigger the application of procedural rules to create detailed geometry and textures, and then explore the complete 3D model at real scale.

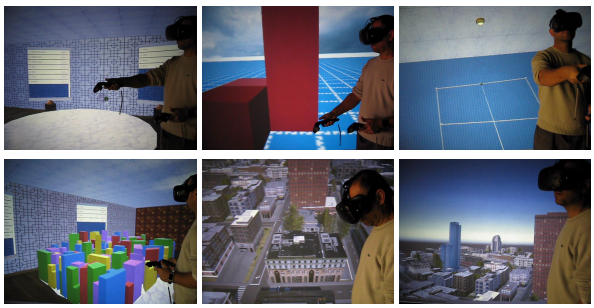

Figure 6: *Some screenshots of the user interface.*

Figure 7: *Creation of a simple city: street network, one user-designed block, map circuits to be filled (shown with streets), generated city blocks, and final city with procedural detail.*

Figure 7 shows the different steps in creating a simple city consisting of a grid-like street network, and different block templates filling the map circuits.

Figure 8 shows some city models created from scratch with our

tool. Observe the many different block shapes generated from a few sample blocks, especially in the last row of Figure 8 where several different blocks have been generated. All models shown in Figure 8 were created in about 5 minutes by one of the authors.

Finally, the user can select a rule set from a menu to give a specific appearance to the final detailed city, as shown in Figure 9.

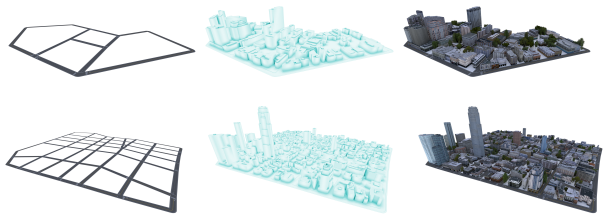

Figure 8: *City models created from scratch with our tool: road networks (textured), blocks and fully-detailed models.*

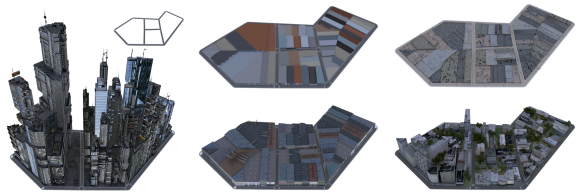

Figure 9: *City models created from the same street network (top-left inset) by refining blocks with different rule sets.*

9. Conclusions and Future Work

We have presented a VR application for the fast creation of city models. The whole user interface has been designed to minimize user input to allow the creation of complex cities within minutes. We allow the user to design template city blocks determining the overall shape of the buildings, and import/create the street network. The system automatically distributes and adapts template blocks to the map circuits defined by surrounding streets. The resulting mass models can be refined through procedural rules. Our current system only allows direct control on street network, building mass volumes and block layouts. Concerning the procedural rules for detailing streets, roofs and facades, we only allow users to choose from a set of predefined rule sets trough a visual menu. An interesting research avenue is how to allow more user control on the final appearance of the facades, while keeping the simplicity of the user interface. Also, an in-depth user study comparing this system with state-of-the-art software tools, such as Blender or Sketch-up is needed to assess fundamental aspects such as long term use of the system (e.g., whether it possible to spend 2 or 3 hours using the system, or how does it compare to a similar tool implemented in a traditional desktop applications).

Acknowledgements

Bruce Branit's science fiction film *World Builder* was a source of inspiration for the UI of our system. This work was partially funded by the TIN2017-88515-C2-1-R and TIN2017-88515-C2-2-R projects from Ministerio de Economía y Competitividad, Spain.

References

[ABB*18] ANDUJAR C., BRUNET P., BUXAREU J., FONS J., LAGUARDA N., PASCUAL J., PELECHANO N.: VR-assisted Architectural Design in a Heritage Site: the Sagrada Familia Case Study. In *Eurographics Workshop on Graphics and Cultural Heritage* (2018), pp. 047–055. 1

[AH77] APPEL K., HAKEN W.: Every planar map is four colorable. part i: Discharging. *Illinois J. Math. 21*, 3 (09 1977), 429–490. 3

[BBP13] BARROSO S., BESUIEVSKY G., PATOW G.: Visual copy & paste for procedurally modeled buildings by ruleset rewriting. *Computers & Graphics 37*, 4 (2013), 238 – 246. 2

[GJWW15] GUERRERO P., JESCHKE S., WIMMER M., WONKA P.: Learning shape placements by example. *ACM Trans. Graph. 34*, 4 (July 2015), 108:1–108:13. 2

[HF06] HORMANN K., FLOATER M. S.: Mean value coordinates for arbitrary planar polygons. *ACM Trans. Graph. 25*, 4 (Oct. 2006), 1424–1441. 3

[JCS16] JESUS D., COELHO A., SOUSA A. A.: Layered shape grammars for procedural modelling of buildings. *The Visual Computer 32*, 6 (2016), 933–943. 2

[KTDH15] KULIGA S. F., THRASH T., DALTON R. C., HOELSCHER C.: Virtual reality as an empirical research tool— exploring user experience in a real building and a corresponding virtual model. *Computers, Environment and Urban Systems 54* (2015), 363–375. 2

[LWW08] LIPP M., WONKA P., WIMMER M.: Interactive visual editing of grammars for procedural architecture. *ACM Transactions on Graphics 27*, 3 (Aug. 2008), 102:1–10. 1

[MWH*06] MÜLLER P., WONKA P., HAEGLER S., ULMER A., VAN GOOL L.: Procedural modeling of buildings. *ACM Trans. Graph. 25*, 3 (2006), 614–623. 1, 2

[NGDA*16] NISHIDA G., GARCIA-DORADO I., ALIAGA D. G., BENES B., BOUSSEAU A.: Interactive sketching of urban procedural models. *ACM Trans. Graph. 35*, 4 (2016). 1, 2

[Pat12] PATOW G.: User-friendly graph editing for procedural modeling of buildings. *IEEE Computer Graphics and Applications 32* (2012), 66–75. 1, 2

[SBS15] SCHMIDT S., BRUDER G., STEINICKE F.: A layer-based 3d virtual environment for architectural collaboration. In *Proceedings of the EuroVR Conference* (2015). 2

[SM15] SCHWARZ M., MÜLLER P.: Advanced procedural modeling of architecture. *ACM Transactions on Graphics 34*, 4 (2015), 107:1–107:12. 2

[STBB14] SMELIK R. M., TUTENEL T., BIDARRA R., BENES B.: A survey on procedural modelling for virtual worlds. *Computer Graphics Forum 33*, 6 (2014), 31–50. 2

[VAW*10] VANEGAS C. A., ALIAGA D. G., WONKA P., MÜLLER P., WADDELL P., WATSON B.: Modelling the appearance and behaviour of urban spaces. *Comput. Graph. Forum 29*, 1 (2010), 25–42. 2

[WMV*08] WATSON B., MÜLLER P., VERYOVKA O., FULLER A., WONKA P., SEXTON C.: Procedural urban modeling in practice. *IEEE Computer Graphics and Applications 28* (2008), 18–26. 2

[WSW*06] WESTERDAHL B., SUNESON K., WERNEMYR C., ROUPÉ M., JOHANSSON M., ALLWOOD C. M.: Users' evaluation of a virtual reality architectural model compared with the experience of the completed building. *Automation in construction 15*, 2 (2006), 150–165. 2

[WWSR03] WONKA P., WIMMER M., SILLION F., RIBARSKY W.: Instant architecture. *ACM Transaction on Graphics 22*, 3 (July 2003), 669–677. Proceedings ACM SIGGRAPH 2003. 1, 2