International Conference on Artificial Reality and Telexistence
Eurographics Symposium on Virtual Environments (2022)
J.-M. Normand and H. Uchiyama (Editors)

ProGenVR: Natural Interactions for Procedural Content Generation in VR

B. Carvalho[1], D. Mendes[1,2], A. Coelho[1,2] and R. Rodrigues[1,2]

[1]Faculdade de Engenharia, Universidade do Porto, Portugal
[2]INESC TEC, Porto, Portugal

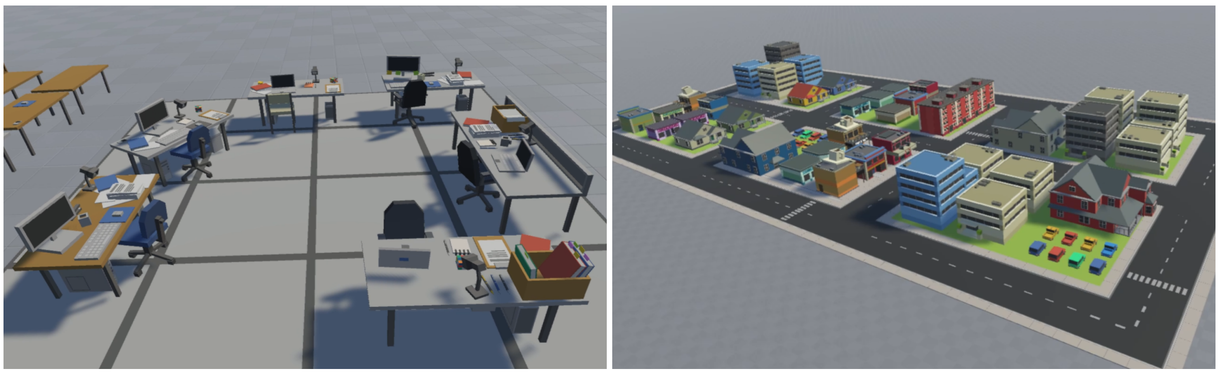

Figure 1: *Two scenes created using ProGenVR: (left) a computer room, and (right) a town center. In both cases, a main model (a desk and a city block, respectively) was created from an asset library using a set of procedural rules, and cloned multiple times to achieve the shown results. While all clones follow the same set of rules, each one is unique due to randomization features.*

Abstract

3D content creation for virtual worlds is a difficult task, requiring specialized tools based typically on a WIMP interface for modelling, composition and animation. But these interfaces pose several limitations, namely regarding the 2D-3D mapping required both for input and output. To overcome such limitations, VR modelling approaches have been proposed. However, translating relevant tools for creating large 3D scenes to VR settings is not trivial. Procedural content generation (PCG) is one such tool that allows content to be automatically generated following a set of parameterized rules. In this work, we propose a novel approach for immersive 3D modelling based on a set of procedural rules for content generation and natural interactions to bridge the gap between immersive content creation and PCG. We developed a prototype implementing our approach and conducted a user evaluation to assess its applicability. Results suggest that the cost of time and mental effort associated with the rules' definition can be compensated by the saved time and physical effort when creating complex scenes.

CCS Concepts

• ***Human-centered computing*** $\rightarrow$ ***Graphical user interfaces; Virtual reality;***

1. Introduction

3D content creation is a complex and time-consuming task usually carried out on a desktop environment, in editors like Maya or Blender with WIMP-style interfaces (windows, icons, menus, pointers). It is used to create digital content like virtual worlds for television and cinema, human and animal models for biology and medicine, machine models for engineering and the manufacturing industry, digital characters and scenarios for video games, etc. Each domain has a different purpose for 3D content creation and therefore requires a different set of tools or approaches. Editors based on WIMP and 2D screens have a few limitations. The first is that the input space is limited to the two-dimensional movement allowed by the mouse, making it difficult and awkward to perform certain kinds of 3D operations. The 2D display output space where 3D scenes and models are visualized is also limited, and the user's per-

DOI: 10.2312/egve.20221277

ception of shape, depth and size would be improved if stereoscopic visualization was employed instead.

To overcome the issues above, 3D immersive editors with spatial input have been researched. However, despite the more natural interactions they offer, translating certain tasks from traditional WIMP/2D tasks to 6-DOF/3D is not trivial. In particular, a set of tools which have become highly relevant in the creation of large 3D scenes is the one of procedural content generation (PCG) tools. These tools enable the generation of multiple entities automatically following a set of rules, which can be parameterized. These rules can specify, for example, the scale and position of multiple generated objects, transformations applied to them, and the number of repetitions to e.g. populate a forest with trees. To create variations and thus avoid repetition patterns, randomness is often used.

If these pose some challenges to users (in particular non-technical ones, as many artists are) in a WIMP/2D setting, their usage in an immersive setting is even more daunting. To address this challenge, we propose a novel approach for creating 3D scenes in an immersive environment using natural interaction and procedural generation techniques. We resort to a set of simple rules that users can apply to existing models, and use as building blocks to hierarchically create complex scenes, as exemplified in Figure 1. We developed a prototype implementing our approach, and validated it through a user evaluation. Therefore, the main contributions of our work are: (1) a set of procedural content generation rules for 3D modelling in immersive environments; (2) an interaction design that allow users to apply such rules using natural metaphors; (3) a working and usable implementation of that proposal; and (4) a user evaluation highlighting the pros and cons of our approach.

2. Related Work

We address the challenges in immersive content editing with procedural generation in three key aspects: content creation in immersive virtual environments (IVE), the particular case of procedural content generation, and current VR interaction models.

2.1. Content creation in IVE

Approaches to content creation can be grouped in the four categories of *sketching*, *instantiation*, *CSG* and *PCG*. Most works are sketch and drawing based, where the user must draw the contours of objects they wish to create, or draw on already existing ones to cut or extrude them [JMY*13, RD19, FLJZ98, DCJH13, IMT06, CRSM13, Hal13]. The instantiation approach is the simplest one – the user selects an asset from a collection and places it in the scene, performs basic manipulation operations – translations, rotations and scaling - and adjusts other properties such as textures and lighting [JMY*13, WLL13, FLJZ98, MMS*17, Bol80]. In the *CSG* approach the user creates new objects using constructive solid geometry operations like intersection and union of two or more objects [RD19, DCJH13, MMS*17]. Finally with the *PCG* approach the user does not directly create every object instance but instead specifies rules, patterns and properties that are used to generate the object(s). This is further explored in the following section.

2.2. Procedural Content Generation (PCG)

In procedural content generation, new content is created through the application of *functions* instead of modelling it interactively or describing it explicitly. The generated content can be adapted from other content, for example by applying random variations, solving constraint problems so to satisfy specific properties, or just exercising a set of transformation rules defined by the creator in a procedural generation language. This practice can be applied to create various different types of data, from textures and random meshes to terrain, trees, road networks, cities and buildings [STBB14].

Two common techniques used are *L-systems* and split grammars [CBSF07]. A basic *L-system* is composed of a formal grammar and a collection of *production rules*; starting with an *initiator* string, each *atom* in the string may be expanded into a string of symbols according to a *generator* rule, recursively. They are usually accompanied with a mechanism for generating a figure or geometric object. A good example of their application is the seminal work of Parish and Muller [PM01] - *CityEngine*. It takes as input various maps with information such as terrain elevation, water bodies and population density, and uses extended L-systems to derive the road network of the city and the distributions of buildings; another L-system is used to generate the buildings themselves after extrusion of each allotment. Split grammars incorporate the notion of geometrical shape into the production rules, making them a better fit for modelling of architecture [CBSF07]. The alphabet's symbols are shapes that can be *split* according to production rules and annotated with additional information. In the work of Wonka et al. [WWSR03], where they were introduced, the symbols of the grammar were augmented with parametric attributes representing physical dimensions of shapes, depth and texture data. A control grammar was further used to help refine the attributes of the shapes created by the split grammar.

2.3. VR interaction models

Interaction models are the ways in which the user interacts with the modelling system and are closely coupled with the input/tracking devices used. Some of the approaches revolve around using hand gestures or 3D-tracked controllers either in a fully immersed virtual reality environment [JMY*13, WLL13, MMS*17, CRSM13] or in an augmented environment [RD19, DCJH13, FLJZ98, Bol80, Hal13]. Others rely on stroking and drawing in either two or three dimensions with a pen or just the user's fingers, and applying operations on them, such as extrusions [RD19, NGDA*16, FLJZ98, DCJH13, IMT06, CRSM13]. Voice commands can be a primary or secondary input mechanism depending on the modelling approach, with uses ranging from tool selection to object manipulation and instantiation [FLJZ98, Bol80]. Other works rely on more traditional WIMP-based input from mouse and keyboard [NGDA*16] or tablet [WLL13], particularly those works of procedural modelling of urban buildings which do not focus on the interaction model [WWSR03, PM01].

Many of the works studied combine 3D input and perceived spaces [JMY*13, RD19, WLL13, DCJH13, MMS*17] while combining two different content creation approaches: sketching [JMY*13, RD19, DCJH13], instantiation [JMY*13, WLL13, DCJH13, MMS*17] and CSG [RD19, DCJH13, MMS*17]. However, we have not found a modelling approach that combines 3D input and perceived space in a fully-immersive virtual environment with procedural generation techniques and a natural interaction model, and therefore proposed to explore that combination with this work.

3. ProGenVR

The proposed approach allows users to assemble models in a virtual reality environment. Our approach considers a model as a composition of various static assets, with hierarchical relationships between these and procedural rules to change their visibility status, placement, and orientation. Users have core modeling operations available to build the model, such as cloning and moving objects, entities such as groups and uniformly spaced tilings, and an assortment of randomization operations and objects. The randomization allows users to clone models, generating a different outcome following the specified rules. This tool can be helpful for quickly spawning several city blocks when creating a city or generating different rooms at runtime that a player can explore.

3.1. Primitive Objects

Our modelling approach starts with loading primitive objects into the scene. We call the initial assets primitive objects because they do not have any procedural rules and can only be moved around and copied. They cannot be edited or partitioned into smaller blocks, unlike procedural objects.

3.2. Procedural Objects

Procedural objects can be either composite or randomized. When creating these procedural objects, all types of objects might be used, both primitive as well as other composite and randomized objects.

3.2.1. Composite Objects

Composite objects are those which contain sets of other objects organized either in a manually defined layout - Groups - or a regular rule-based layout - Tilings - as described next.

Group. The first composite object is the Group, which is the union of one or more objects with a fixed relative layout. A Group is created by selecting one or more objects in the scene, and then joining them into one aggregate object. The incorporated objects are called the elements of the group. Once formed, the Group can be moved and rotated as a single object and its elements keep their relative distances and orientations.

Tiling. Another composite object is the Tiling. A Tiling wraps a single object, the tile or child, and lays out a fixed number of clones of this tile along a straight line with uniform spacing (Figure 2). The number of tile clones, the spacing between tiles and the orientation of the tiles can be adjusted, and the tiling direction can be changed by rotating the Tiling object itself. To create matrices of objects, a Tiling object can be used as the child of another Tiling object.

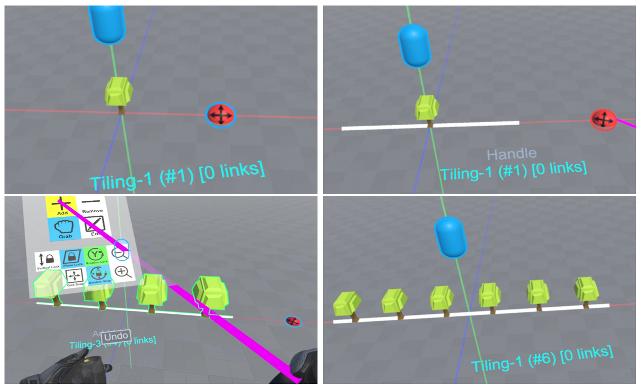

Figure 2: *Editing a Tiling procedural object. The red handle is used to set the tiling range and the blue one can be used to change the tiles' orientation. The number of tiles can be increased and decreasing using menu buttons.*

3.2.2. Randomized Objects

Randomized objects are objects that can assume different forms or properties, either by selecting a random object from a given set, or applying randomized transformations to an object.

Random. The first procedural object is the Random, which can be constructed in a similar way to the Group, but instead of showing the union of all incorporated objects with a fixed layout, it shows exactly one of these objects, chosen uniformly at random. To have different probabilities for each object being shown, multiple copies of the objects can be used in the same Random. The incorporated objects are called the variants of the Random, and exactly one of these variants is visible or active at any moment. Upon creation of the Random, all variants snap to the position of the first selected at creation time, but retain their world-space orientation. All the variants appear, by default, in the same place, though this can be adjusted later.

Mover. The Mover adjusts an object's location by applying a random offset chosen uniformly at random from within a defined three-dimensional offset range (Figure 3). This can be used to create variations of an object that does not appear in the same position. For instance, it can be used within a Tiling object to create a row of objects that are not perfectly aligned.

Rotator. The Rotator is similar to the Mover, focusing instead on object orientation. It adjusts an object's rotation around one of the primary axis, by a random angle chosen uniformly at random from within a defined angle range (Figure 4).

3.2.3. Editing Objects

All procedural objects may be edited after creation, for example to adjust children positions and orientations, add and delete elements or variants, or adjust parameters like spacing. When editing an object, it is called the edit subject, and only its children can be modified or moved. The operations available while editing are only those that pertain to adjusting the subject. When editing a Group, the user can modify the relative positions of the elements, delete elements, and add new elements, either by cloning elements within the group or from outside into the group. Similarly, when editing a Random the user can change the relative positions of the variants and also add and remove variants. In the case of the Random, this introduces the ability to have the variants appear with different relative positions. For reference, a set of axes are shown in the origin of the local space of the object being edited.

Tilings, Movers and Rotators are edited in the same way their parameters are set on creation. For the Tiling, the number of tiles and the spacing between the tiles can be adjusted. For the Mover, the permissible offset box appears while editing and it can be resized. For the Rotator the permissible angle range can be adjusted around one of the three rotation axes. Editing may be performed recursively in the object tree: while editing a Group, the editing process can go to one of its children; while editing a Random the active variant may be edited instead; and while editing a Tiling, Mover or Rotator the child entity can be edited.

3.3. Empty Object

The last object type of our approach is the Empty object, which behaves identically to a Primitive object but is actually invisible. This object can be used to represent the concept of nothing among the variants of a Random. The visibility of Empty objects can be toggled on and off, so they have a visual representation the user can interact with.

3.4. Object Trees

The composition of objects form object trees. The children of a group node are the Group's elements, the children of a random node are the Random's variants (including those not shown), and the other three object types have only one child. The leaves of an object tree are all Primitive or Empty entities. Figure 5 show three examples of object trees. Once the models are built, the root objects G, M and T_2 are the only objects that the user can interact with.

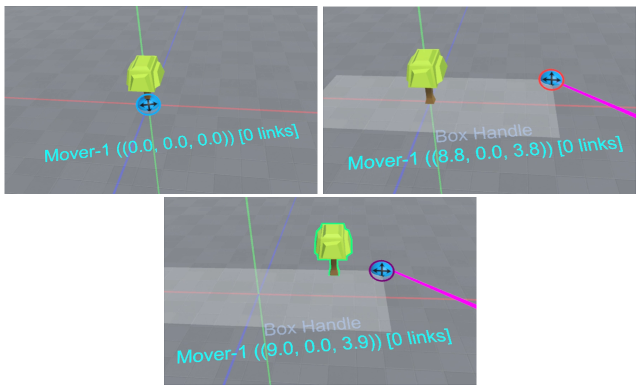
Mover-1 ((0.0, 0.0, 0.0)) [0 links]
Box Handle
Mover-1 ((8.8, 0.0, 3.8)) [0 links]
Box Handle
Mover-1 ((9.0, 0.0, 3.9)) [0 links]

Figure 3: *Editing a Mover procedural object. The handle starts in the origin of the local space (top left), the user drags the handle creating a box centered in the origin which defines the offset range (top right), and a new random offset is applied after releasing the handle (bottom).*

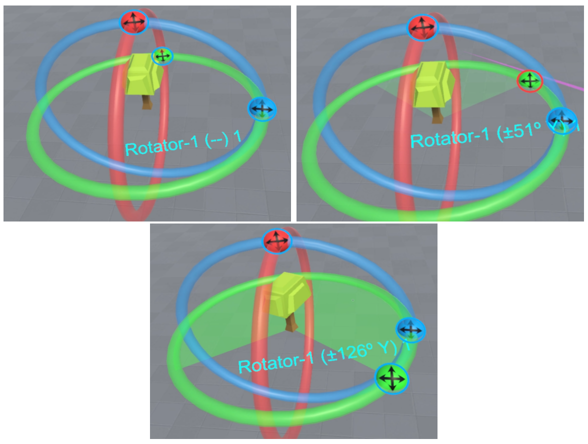

Figure 4: *Editing a Rotator procedural object. The three handles illustrate each of the available rotation axis (top left), the user drags one of the handles defining the angle range (top right), and a new random rotation is applied after releasing the handle (bottom).*

3.5. Operations

Besides moving entities, and creating and editing procedural objects, the following operations are available to users.

3.5.1. Clone + Linking

A Clone operation can be applied to any object, and it creates another instance of the same object, with its children cloned recursively. The object tree of the clone is identical to the original, therefore following the same rules. Cloning a Primitive object creates

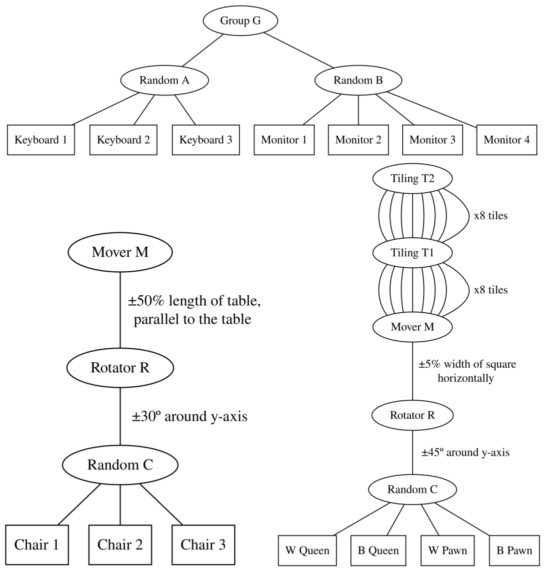

Figure 5: *Object trees for three examples: a computer prop (top), a chair prop (bottom left), and a chessboard (bottom right).*

another Primitive object with the same asset. Procedural objects are cloned by reference. When a Group G_1 is cloned into another group G_2, the two groups become linked. If another clone G_3 is made of either G_1 or G_2, then the three groups will be linked to each other. The same link-on-clone logic applies to the other composite and procedural objects. We call two objects that are linked to each other siblings.

Linking is recursive. If G_1 has another linked object A_1 as its element, then the corresponding clones A_2 and A_3, in G_2 and G_3 respectively, are also linked to each other and to A_1. Also, linking propagates modifications. When a linked object is modified, the modification automatically propagates to all of its siblings. For example, if the designer edits group G_1 by cloning an object X and adding it as a new element X_1, then two clones X_2 and X_3 of X are automatically created and added to G_2 and G_3, respectively, in the same position and orientation. If the user then moves X_1 inside G_1, the same movement is applied to X_2 and X_3 in their own local space, and so on for the remaining operations available to edit G_1. The user has live feedback of these mirror operations on the sibling objects, so he can visualize the effect of his changes simultaneously on all siblings that fit in his field of view. Lastly, linking is reciprocal. Modifications made to G_2 also propagate to G_1 and G_3; there is no master object. However, linked objects are not necessarily identical. For instance, if two Randoms R_1 and R_2 are siblings they have the same object tree structure and also the same set of variants, but they do not necessarily show the same variant. Similarly, two Mover objects M_1 and M_2 have identically sized offset ranges, but their random offsets are different. Two siblings still make different and independent random choices.

3.5.2. Unlink

Using the Unlink operation on a linked object, the user can break the link that object has to all of its siblings. In the previous example, applying the Unlink operation to G_3 allows the user to modify G_3 without propagating the changes to G_1 or G_2 and vice-versa. Edits to G_1 will still propagate to G_2 and edits to G_2 will still propagate to G_1. If G_3 is then cloned to G_4 then G_3 and G_4 will be siblings, but still separate from G_1 and G_2.

3.5.3. Reroll

A Reroll operation can be applied to any object, and it requests the object to make a different set of random choices, recursively down the object tree. When applied to a Primitive object the operation does nothing. When applied to a Random object it requests that another variant be shown, and that the shown variant itself be rerolled. When applied to a Group the elements are themselves rerolled, and similarly for a Tiling. When applied to a Mover the random offset is rerolled, and when applied to a Rotator the random angle is rerolled, and then the child object itself is rerolled as well.

3.5.4. Disband

A Disband operation can be applied to any object and it serves to break a composite object into its constituents, by deleting the root node of the object tree but not its children. When applied to a Primitive object this operation does nothing. When applied to a Group or Tiling with n elements or tiles, the Group/Tiling node disappears while the n entities remain as object roots. When applied to a Random a similar thing happens, but only one root remains, the visible variant, with the hidden variants being deleted. When applied to a Mover or Rotator, the position/orientation offset is bound to the child object, as if the user had specified it, and the modifier node is removed, leaving the child element as a new object root.

3.5.5. Delete

Lastly, the Delete operation deletes the whole object tree rooted at the object it is applied to. This is also available while editing Groups and Random to remove elements and variants, respectively.

4. Prototype

The prototype was developed in Unity with the SteamVR tool. For development, we used an Oculus Rift virtual reality headset. The user holds two controllers, one on each hand. The dominant hand's controller shoots a laser outwards that collides with the first object or interface entity in its path. The laser is also used to interact with an interface menu floating over the user's non-dominant hand (Figure 6). Besides options to create procedural objects and execute operations, the menu also includes options common in 3D modelling applications, such as movement and rotation lock, snapping and scene scaling. Locomotion is possible by moving around in the real-world playing area, and it can be aided by standard arc teleportation.

A scene initially contains only Primitive objects. All objects in the scene are static – they do not respond to gravity and do not collide with each other or with the user. Our prototype supports undo and redo actions, by maintaining a stack of modifications applied to the objects in the scene. A simple text log is shown fixed and over other objects just below the center of the user's field of view. It shows information of the object currently being interacted with, such as: asset name, object type, number of elements and variants in a Group or Random, offset or angle range in a Mover or Rotator, number of tiles in a Tiling, as well as the number of siblings.

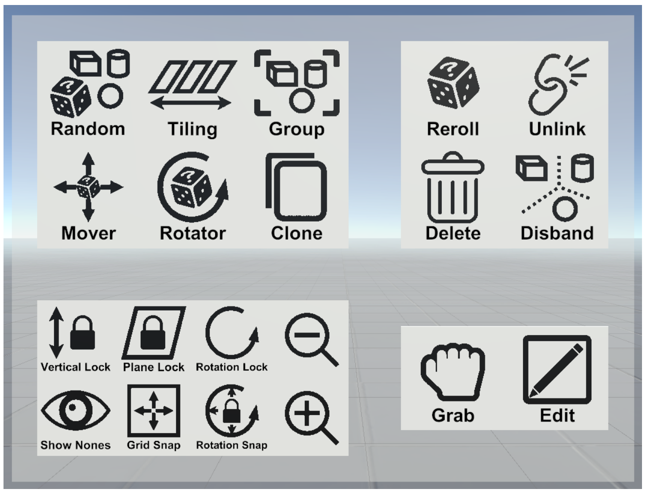

Figure 6: *Floating menu over the user's non-dominant hand.*

5. User Evaluation

We conducted an evaluation to both assess the prototype's viability and usability and compare our modelling approach with a baseline consisting of the core subset of objects and operations. We present the evaluation methodology used, the results obtained and the conclusions that follow from them.

5.1. Method

Our primary goal in this study is to identify whether the procedural approach grants a significant speedup in the development of models with moderate complexity, over the baseline. Hence, we prepared a test protocol, and reached out openly for volunteers in our university to participate in the tests. No prior experience with virtual reality devices and applications, modelling applications or programming was required. The details on the tests themselves are presented next.

5.1.1. Testing environment

All volunteers used an Oculus Rift headset during testing. Sessions were expected to last between 60 and 70 minutes. Participants without any prior virtual reality experience were given an informal and gentle introduction in the Google Earth VR demo for about 15 minutes before the session proper. This included an introduction to laser mechanics, the VR controllers, and gave inexperienced volunteers an opportunity to adjust and settle in this virtual environment. All participants agreed to continue with the session after this introduction. Participants were given a standard consent form, informing them they could interrupt or abort the session at any moment, and that their execution of the tasks within the modeling sandbox would be recorded anonymously for later analysis.

5.1.2. Tutorial

The users were introduced to the system with a guided tutorial in a very simple sandbox with five different trees and a square grass tile (Figure 7). The tutorial has two parts. The first part is a very quick introduction to the basic elements of the system, where the user is guided to perform a set of well-defined steps regarding locomotion, teleportation, grabbing objects, the undo system, and the baseline menu's operations. The second part of the tutorial introduces the procedural operations and objects. The Random, Mover and Rotator objects are introduced outside of the virtual reality environment with real-world examples and analogies, alongside the Reroll operation. The user then goes back to the tutorial sandbox and is guided to perform a new set of steps using these new tools.

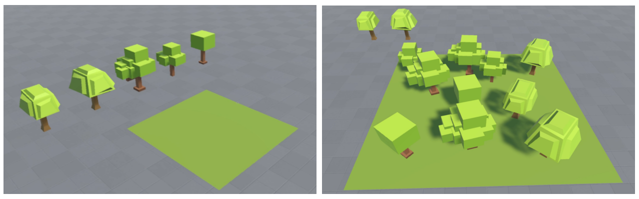

Figure 7: *Tutorial scenario set. On the right a possible output after the tutorial's first part.*

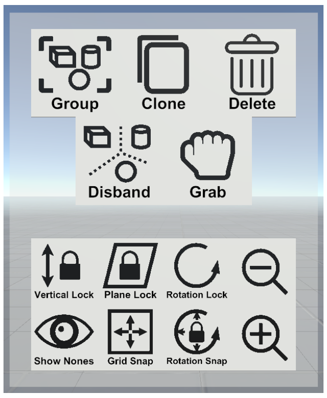

Figure 8: *Restricted baseline menu.*

5.1.3. Approaches

We evaluated two modelling approaches: our procedural approach, ProGenVR, with all modeling operations and objects available, and a baseline approach that restricts the set of objects and operations available to only grouping, cloning, deleting and disbanding objects (Figure 8). In this approach, there are no randomization tools available. It is still possible to use the undo system and there is still feedback for all operations performed.

5.1.4. Tasks

Each session was split into two segments. In one segment, participants performed two tasks using the baseline approach, and in the other segment used the full procedural approach. Approximately half of the participants began the session with the baseline approach and the others with ProGenVR to prevent bias in either direction. In each segment, participants had to perform two tasks, consisting of assembling different scenarios, for a total of four tasks for the whole session. In the first task, participants were asked to assemble a computer room, and in the second task a town center. Participants were informed of both tasks, their specifications, the structure of the entire session, and the nature of both approaches upfront. Before each task, participants went through the specifications again.

In the first task, participants were given a library of assets (primitive objects) consisting of objects that can often be found in an office or classroom, including tables, chairs, computer monitors, keyboards and a laptop, mousepads, notebooks, pencils, paper stashes, desk lamps and a few more. The user must assemble the room over a designated area right next to the library (Figure 9). The room must meet the following specifications:

- There must be exactly six desks in the room, with no particular layout required.
- Each desk must have one chair in front of it, one computer set and at least three other assets on top of it.
- The computer set must be either a laptop with no keyboard, or a monitor with a keyboard.
- Objects should not overlap each other.
- The desks should have different sets of items and the layout of these items should vary. This requires varying the chair, tabletop and computer set assets per desk.

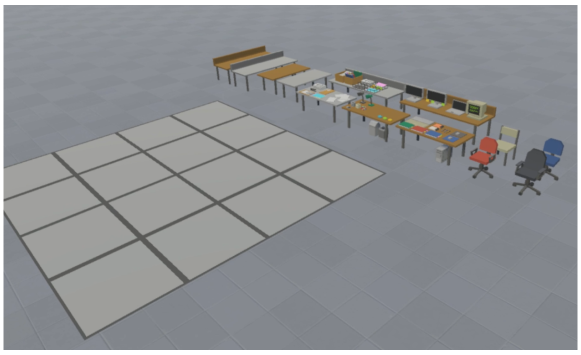
Figure 9: *Computer room task: the initial scene and asset library.*

In the second task, participants were given a similar setup with assets to create a town center including grass, pavement, parks and buildings of various type, shapes and sizes. Participants must assemble the town inside the road skeleton provided right next to the library (Figure 10). The skeleton had six empty blocks, and several buildings can fit in each block. The town meet the following specifications:

- All six blocks must be filled with buildings, no particular design enforced.
- All types and shapes of buildings must be used across all blocks.
- Buildings and parks must not overlap each other.
- The blocks should have different sets of buildings on them, but the layout of the buildings does not have to vary.

With the baseline approach, participants could assemble the scenarios however they wished, and they were informed that the goal was to produce the design as quickly as possible. With ProGenVR, participants were suggested to assemble the unit model first. In the first scenario this was a desk group that meets the desk specifications, and in the second scenario it was a block group that met the block specification, both using procedural objects. The whole scene should then be assembled only by cloning this unit model six times to meet the specifications for the whole scenario. A possible outcome of both tasks is shown in Figure 1. This requires participants to work backwards, to figure out what structure the unit model should have to ensure that the clones would meet the global variation constraint. Since our primary goal was assessing the difference between task completion times, we deemed it proper to inform the users whenever they broke the specification.

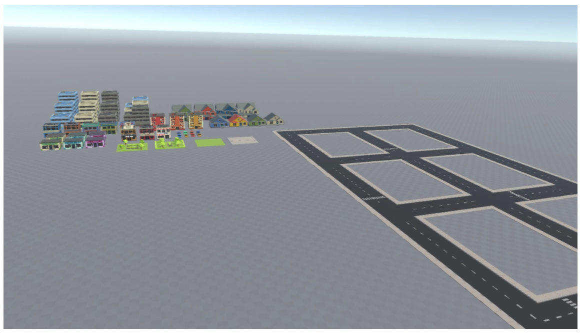
Figure 10: *Town center task: the initial scene and asset library.*

5.1.5. Questionnaire

Before beginning the session participants were asked to fill in a form with their profile information, including age, education and experience with virtual reality or modelling applications. After concluding the first two tasks participants were asked to answer a short questionnaire about their experience in those two tasks, and then again after the last two tasks when the session was finished.

5.2. Participants

We gathered a total of 15 participants (1 female). Their age ranged from 16 to 30 years old. Regarding modelling application experience, 5 (33,3%) had never used them, 7 (46,7%) used them once a month or less, and 3 (20%) used them between once a day and once a week. In terms of VR applications, 6 (40%) had not tried them yet, 6 (40%) used them once a month or less, and 3 (20%) used them between once a day and once a week.

5.3. Results

The primary goal of the study was to determine, for each of the tasks, whether the procedural approach offered an advantage in terms of modelling time and complexity versus the baseline and, if so, to which degree. We were also interested in evaluating the task load associated with both approaches.

5.3.1. Task performance

For all tasks we tracked the total time to completion. For the procedural approach's tasks we also tracked the time to the first model, i.e. how long the user took to form the group model of the desk or square block. (Figure 11). The distributions of task completion

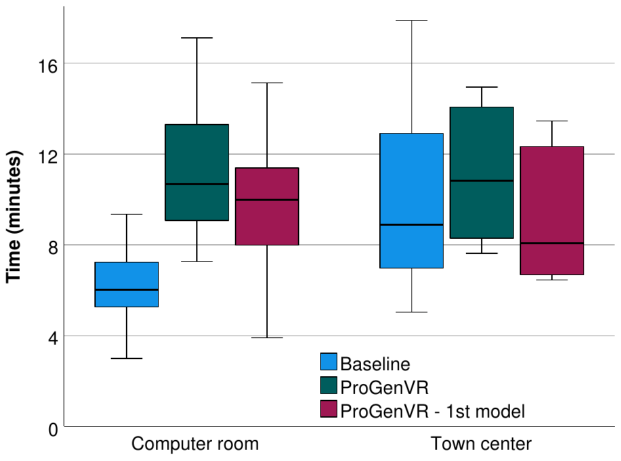

Figure 11: *Completion times for both tasks using the two approaches, and time to finish the first model with ProGenVR.*

	Computer room		Town center	
	Baseline	ProGenVR	Baseline	ProGenVR
I found the task mentally demanding.	3 (3)	2 (1)	3 (2)	3 (1)
I found the task physically demanding.	3 (4)	4 (4)	3 (4) *	4 (3) *
I feel the pace of the task was hurried or rushed.	3 (4)	3 (3)	3 (3)	4 (4)
I think I was successful in accomplishing what I was asked to do.	2 (4)	2 (3)	2 (4)	2 (3)
I had to work hard to accomplish my level of performance.	3 (2)	2 (1)	3 (1)	2 (2)
I felt insecure, discouraged, stressed or annoyed.	4 (4)	4 (4)	4 (4)	3 (4)

Table 1: *Questionnaire results (1-Strongly Agree, 5-Strongly disagree): means and interquartile ranges. * indicate statistical significance.*

time are approximately normally distributed (Shapiro-Wilk significance of 0.581 and 0.111 for baseline, and 0.754 and 0.030 for procedural approach). Therefore, we used the paired-samples t-test to assess statistically significant differences. In the room scenario, participants consistently took longer with the procedural approach to complete the task ($t(14) = 5.33, p < .001$). In fact every participant took strictly more time, on average 70% longer for the whole task. A significant contributor was the monitor/laptop challenge we further report. In the town scenario there is once again a significant difference for completion time ($t(10) = 2.717, p = .022$). One contributor to this fact was that aligning the large block model clones into the road skeleton was quite difficult if the user did not zoom out in this task, and very few actually did. Another contributor is the fact that participants could create a very simple block model and still meet all the required specifications.

Considering the times to complete the first model of each task, it is noticeable that this makes for the most part of the task execution. We calculated that participants took on average 95 seconds to conclude the computer room task after finishing the model. Increasing the number of required desks clearly favours the procedural approach. Extrapolating the mean time to model of all participants suggests the break even point is at approximately 12.3 desks, assuming the time to complete the baseline version of this task is proportional to the number of models created. For the town scenario the break even point was 9.8 blocks.

5.3.2. Usability and Task Load

After each session segment we inquired about task load, using a questionnaire with a 5-point Likert scale based on the NASA-TLX (Table 1). This assessment did not vary significantly between the two approaches, according to the Wilcoxon signed-rank test. The only exception is the reported physical load in the town center task ($Z = -2.070, p = .038$). This might be related to the extensive repetitive gestures in a larger environment required by the baseline.

5.3.3. Observations

As the final models were not required to look aesthetically pleasing, most design decisions were left open to the participants. In practice, almost all participants designed the computer room in roughly the same way in both runs, placing the computer in the center of the table, the chair facing the table, and the remaining items either left or right of the computer. In the city task, however, many participants specialized the square blocks with the baseline approach, such as making two residential blocks, two commercial blocks, and two office blocks. This was explicitly allowed, but with the procedural approach (before or after) this specialization required far more effort and complexity, so they instead created a block with a fixed layout and practically no specialization. We also noted that there is a peculiar difficulty in the room task using the procedural approach. It was easy to create the computer set model incorrectly by aggregating the monitors and the laptop together into one Random entity, which could make the laptop to appear with a keyboard in front.

5.3.4. Discussion

Despite the small sample size of our study, some important aspects can be identified. In terms of performance, there is an initial cost to the procedural approach (compared to the baseline) that is compensated when generating a significant number of elements afterwards. Regarding task load, a similar trade-off is observed: a possible larger mental load in creating the procedural rules in the beginning is compensated later by reduced physical load in generation later, compared to the baseline approach. This suggests that the procedural approach can be beneficial for generating more complex scenes, both in terms of time and mental and physical load.

6. Conclusions and Future Work

We presented an overview of the difficulties associated with content creation, particularly in virtual reality settings. Then we detailed our proposed approach for modelling in virtual reality with procedural content generation tools, centered around a system with primitive, procedural and composite objects supporting basic operations such as replication, relative and randomized placement, orientation and spacing along different axes, and random selection. We designed and prototyped a natural interaction model for our approach. We conducted a user evaluation to compare our approach against a baseline without procedural tools. Results suggest that our approach can be better for complex scenes, in terms of time and mental and physical loads.

As future work, we would like to further explore PCG features and how to use them within an immersive environment. Some kind of constraint solving that would prevent content resulting from randomization to overlap/collide with each other would also benefit our approach. Also, our prototype works with assets that were previously prepared and imported to the scene. We would like to make this more flexible by allowing on-the-fly loading of any kind of 3D model. Lastly, we intend to explore ways to make this approach for 3D content creation collaborative, allowing multiple users to contribute for the specification of the rules of a scene.

References

[Bol80] BOLT R. A.: "Put-That-There" : Voice and Gesture At the Graphics Interface. *Comput Graphics (ACM) 14*, 3 (1980), 262–270. doi:10.1145/965105.807503. 2

[CBSF07] COELHO A., BESSA M., SOUSA A. A., FERREIRA F. N.: Expeditious modelling of virtual urban environments with geospatial l-systems. In *Computer Graphics Forum* (2007), vol. 26, Wiley Online Library, pp. 769–782. 2

[CRSM13] COCHARD D., RAHIER P., SAIGO S., MALTOUF M. F.: Building Worlds with Strokes. *IEEE Symposium on 3D User Interface 2013, 3DUI 2013 - Proceedings* (2013), 203–204. 2

[DCJH13] DE ARAÚJO B. R., CASIEZ G., JORGE J. A., HACHET M.: Mockup Builder: 3D modeling on and above the surface. *Computers and Graphics (Pergamon) 37*, 3 (2013), 165–178. doi:10.1016/j.cag.2012.12.005. 2, 3

[FLJZ98] FORSBERG A. S., LAVIOLA JR. J. J., ZELEZNIK R. C.: ErgoDesk: A framework for two-and three-dimensional interaction at the ActiveDesk. *Proceedings of the Second International Immersive Projection Technology Workshop* (1998), 11–12. 2

[Hal13] HALD K.: Low-cost 3DUI using hand tracking and controllers. *IEEE Symposium on 3D User Interface 2013, 3DUI 2013 - Proceedings* (2013), 205–206. doi:10.1109/3DUI.2013.6550250. 2

[IMT06] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: A sketching interface for 3D freeform design. *SIGGRAPH 2006 - ACM SIGGRAPH 2006 Courses* (2006). doi:10.1145/1185657.1185772. 2

[JMY*13] JERALD J., MLYNIEC P., YOGANANDAN A., RUBIN A., PAULLUS D., SOLOTKO S.: Makevr: A 3d world-building interface. In *2013 IEEE Symposium on 3D User Interfaces (3DUI)* (2013), IEEE, pp. 197–198. 2, 3

[MMS*17] MENDES D., MEDEIROS D., SOUSA M., FERREIRA R., RAPOSO A., FERREIRA A., JORGE J.: Mid-air modeling with Boolean operations in VR. *2017 IEEE Symposium on 3D User Interfaces, 3DUI 2017 - Proceedings* (2017), 154–157. doi:10.1109/3DUI.2017.7893332. 2, 3

[NGDA*16] NISHIDA G., GARCIA-DORADO I., ALIAGA D. G., BENES B., BOUSSEAU A.: Interactive sketching of urban procedural models. *ACM Transactions on Graphics 35*, 4 (2016), 1–11. doi:10.1145/2897824.2925951. 2

[PM01] PARISH Y. I., MÜLLER P.: Procedural modeling of cities. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), pp. 301–308. 2

[RD19] REIPSCHLÄGER P., DACHSELT R.: DesignAR: Immersive 3D-modeling combining augmented reality with interactive displays. *ISS 2019 - Proceedings of the 2019 ACM International Conference on Interactive Surfaces and Spaces* (2019), 29–41. doi:10.1145/3343055.3359718. 2, 3

[STBB14] SMELIK R., TUTENEL T., BIDARRA R., BENES B.: A survey on procedural modelling for virtual worlds. *Computer Graphics Forum 33*, 6 (2014), 31–50. doi:10.1111/cgf.12276. 2

[WLL13] WANG J., LEACH O., LINDEMAN R. W.: DIY World Builder: An immersive level-editing system. *IEEE Symposium on 3D User Interface 2013, 3DUI 2013 - Proceedings* (2013), 195–196. doi:10.1109/3DUI.2013.6550245. 2, 3

[WWSR03] WONKA P., WIMMER M., SILLION F., RIBARSKY W.: Instant architecture. *ACM SIGGRAPH 2003 Papers, SIGGRAPH '03* (2003), 669–677. doi:10.1145/1201775.882324. 2