architecture

MDPI

Article

Algorithmic Design in Virtual Reality

Renata Castelo-Branco *,† **and António Leitão** †

INESC-ID/Instituto Superior Técnico, University of Lisbon, 1000-029/1049-001 Lisbon, Portugal; antonio.menezes.leitao@tecnico.ulisboa.pt

* Correspondence: renata.castelo.branco@tecnico.ulisboa.pt

† These authors contributed equally to this work.

Abstract: Virtual reality has been shown to facilitate perception and navigation inside 3D models, while stimulating creativity and enhancing architect/client interaction. In this scenario, in order to better explore paths along the design space that are suggested from this interaction, it is important to support quick updates to the model while still immersed in it. Algorithmic design, an approach to architectural design that uses parametric algorithms to represent a design space, rather than a single design instance, provides such support. We present a novel architectural design process based on the integration of live coding with virtual reality, promoting an immersive approach to algorithmic design. The proposed workflow entails the use of an algorithmic design tool embedded in a virtual environment, where the architect not only creates the design but also interacts with said design, changing it by live coding its algorithmic representation from within virtual reality. In this paper, we explain the challenges faced and solutions devised for the implementation of the proposed workflow. Moreover, we discuss the applicability of algorithmic design in virtual reality to different stages of the architectural design process and the future developments that may arise from this proposal.

Keywords: virtual reality; algorithmic design; live coding

Citation: Castelo-Branco, R.; Leitão, A. Algorithmic Design in Virtual Reality. *Architecture* **2022**, 2, 31–52. https://doi.org/10.3390/architecture2010003

Academic Editors: Tigran Haas and Avi Friedman

Received: 27 December 2021
Accepted: 28 January 2022
Published: 3 February 2022

1. Introduction

Current digital representation tools for architectural design employ a visualization strategy that conveys the building's form through the use of plans, section, elevations, and perspective views. This forces architects and clients to not only combine separate views of the model to form a mental model of the entire scenario [1], but also to scale it to real size in their imagination. This process can be considerably improved through Virtual Reality (VR) technology, allowing users to inhabit the building and perceive it at its natural scale [2].

Despite the advantages VR brings to architectural design, it has an important limitation: the majority of the ideas suggested during a VR session cannot be implemented and experimented during that session. In fact, most approaches for changing the design, with or without VR, are limited to manual model manipulation and, thus, cannot have extensive effects in the design. Algorithmic Design (AD) addresses this problem through the algorithmic description of the model, allowing parametric changes to the design that preserve its internal logic, thus ensuring the consistency of the modified design. However, AD has not yet been used in combination with VR and, thus, designers cannot update the algorithmic descriptions of their models while immersed in a Virtual Environment (VE).

To address this problem, in this paper, we propose Algorithmic Design in Virtual Reality (ADVR), a workflow based on the use of algorithms to represent designs that supports extensive design changes while immersed in VR. Using ADVR, architects can quickly experiment design variations without leaving the VE. However, this workflow also entails considerable challenges, which we discuss in the following sections.

The article structure is organized as follows. Section 2 presents the literature review on the applications of VR in architecture and existing solutions for changing the design in VR. To that end, we also elaborate on the definition and state of the art of AD and live coding. Section 3 introduces the goals and methods of the proposed research. Section 4 describes

the building blocks of the proposed methodology. Sections 5 and 6 present, respectively, the evaluation of the methodology with three exploratory exercises, and the discussion of the obtained results. Finally, Section 7 produces the conclusion and future research paths.

2. Related Work

In 1965, Ivan Sutherland envisioned the ultimate display [3], a reality within which the computer controls the existence of matter itself. Although we are still far from achieving this radical vision, VR technology can already provide enough realism and immersiveness to make it an appealing tool for various professions, and architecture is no exception. AD uses algorithms to describe architectural designs, producing complex and parametric results that greatly benefit from immersive visualization. Having this in mind, several approaches have been developed that allow architects to access and change their AD programs from the VE. This section presents the current state of the art on the integration of parametric and algorithmic-design solutions with VR.

2.1. Virtual Reality in Architecture

The architectural industry is rapidly embracing VR since this technology helps users understand building designs, particularly when compared to the traditional alternative, which is based on flat projections that must be combined and scaled to form a mental model of the design [1]. Figure 1 illustrates this scenario for the case of the Astana National Library (ANL) project. This building, originally designed by Bjarke Ingels Group in 2008, has the shape of a 3D Moebius strip, a rather complex form for one to understand through projections only. Although architects are known for their 3D visualization capabilities, therefore being more than able to work within these boundaries, their clients, on the other hand, are not. For this reason, early adoption of VR technologies in this industry was mostly motivated by contractors and real estate owners [4], but nowadays more and more architectural studios use them to support design processes as well [2,5–8].

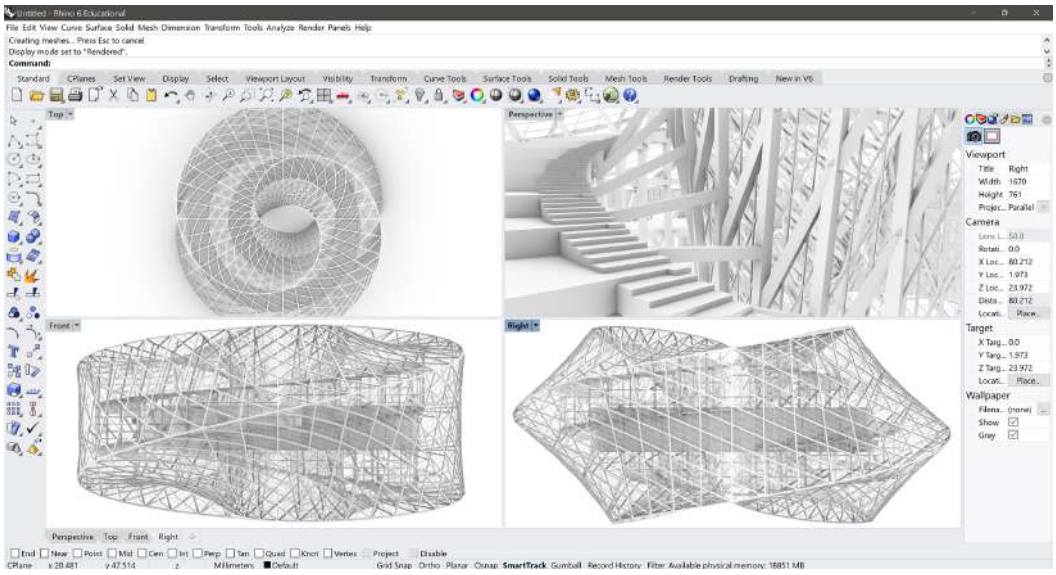

Figure 1. Rendered ANL model in Rhinoceros 3D shown in the 4 default viewports (top, perspective, front, and right view).

Despite the advantages of VR for facilitating the understanding of a building design, there is one significant limitation: most applications of VR in architecture focus on model visualization only [9–11], and the few existing approaches for changing the design while in the VE require virtual manual model manipulation [12,13] using VR controllers and associated transformation operations. This modeling workflow tends to be slower than the traditional (digital) one, which in turn is already quite slow when systematic changes are needed. There is one approach, however, that overcomes this problem: AD.

2.2. Algorithmic Design

AD defines the creation of architectural designs through algorithmic descriptions, implemented as computer programs [14,15]. AD allows architects (1) to model complex geometries that would take a considerable amount of time to produce otherwise, (2) automate repetitive and time-consuming tasks, and (3) quickly generate diverse design solutions without having to rework the model for every iteration [16,17]. Figure 2 presents several variations of the ANL model, obtained by manipulating the radius of the central plaza and the building's facade torsion.

Traditionally, AD entails the incremental development of the program that describes a building design and the visualization of the generated model to confirm the correctness of the program. In the case of VR, however, the design process needs to become a real-time and interactive one. Real-time interaction between the program and the model requires the use of a very performative AD tool, so that the designer can get immediate visual feedback on the impact of the changes applied to the program. Grasshopper [18], a visual programming environment, and LunaMoth [19], a textual programming one, are good examples of tools that possess this liveliness [20] aspect. Nevertheless, they were not intended to be used in VR and known integrations with this technology remain within the sphere of static visualization [21], failing to take advantage of AD's interactive potential.

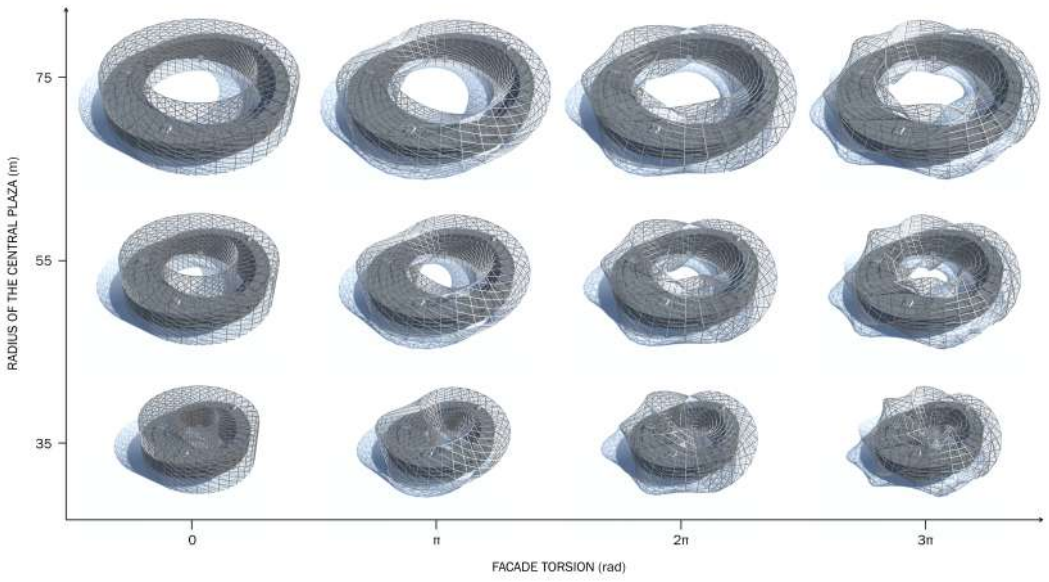

Figure 2. Variations of the ANL model produced by changing selected parameters in the program: radius of the central plaza and the building's facade torsion.

2.3. Live Coding

Live coding is a technique centered upon the writing of interactive programs on the fly [20]. In the case of VR, we envision the application of live coding in a scenario where the architect can code the algorithmic description of the model alongside the resulting geometry. The idea is similar to the workflow supported by some AD tools such as Dynamo [22] and Luna Moth [19], where the code (or visual components) is manipulated in the same environment where the model is concurrently being updated.

The concept of using VR for coding is also not a novelty. The Primitive tool [23], for instance, has proven the use of VR for collaborative software analysis visualization. Specific coding applications for VR have also been developed, such as NeosVR's LogiX [24]—a VR multiplayer visual programming system; Rumpus [25]—a live coding playground for room-scale VR; Fenek [26]—a live coding JavaScript framework that allows developers to modify the underlying render engine while immersed in the VE; RiftSketch [27] and CodeChisel3D [28]—both browser-based live coding environments for VR. However, most

of these tools were only tested with simple graphical models, and none were applied to the architectural context.

2.4. Parametric Design Solutions for Virtual Reality

Within the specific context of architectural design, by transporting the programming environment to the VE, we can have architects and clients manipulating the models in VR. Given that, in this industry, the production line involves multiple iterations of the solutions discussed by several stakeholders, having these discussions take place synchronously with the project's development can resolve the existent asymmetric collaboration [29]. This, in turn, considerably accelerates the ideation process, thus, saving time and resources.

Solutions that allow for the modification of programs in VR have also been developed. Parametric Modelling Within Immersive Environments [30], Shared Immersive Environments for Parametric Model Manipulation [31], and Immersive Algorithmic Design [32] present solutions to connect AD tools to a VE, where architects are immersed in their models, apply changes to the program, and visualize the corresponding impacts in real time.

All three solutions contemplate a visual programming outline, with the former two [30,31] only allowing parameter manipulation. This means users do not have access to the entirety of the code in VR, but only to a chosen set of parameters, which they can change via sliders. For this reason, these solutions are framed within the scope of parametric design only, not AD, according to the definitions proposed by [33]. The third approach [32] goes further, also supporting textual programming and full control over the program. Nevertheless, and despite having presented multiple code manipulation solutions in the paper, the implementation of these solutions within the proposed system is not discussed, nor is the proposal formally evaluated.

3. Goals and Methods

The present research proposes to augment the workflow proposed by Castelo-Branco et al. [32], namely by (1) focusing on the textual paradigm alone, (2) depicting the building blocks of the proposed system in order to provide readers with a reproducible methodology they can use, and (3) performing preliminary evaluation exercises to understand the main benefits and challenges of the proposed solution.

Although it is possible to use the interaction mechanisms available in visual programming to manipulate code in VR, the complexity of the models that most benefit from immersive visualization make visual programming an inviable solution. Despite being more difficult to learn, textual programming languages tend to offer more expressive power, scalability, flexibility, and efficiency than visual ones [34,35]. Hence, we propose to enrich the design experience by integrating a textual-based AD practice, which natively supports extensive design changes, with VR, an intrinsically interactive and immersive medium. The following sections present ADVR, a methodology for interacting with textual AD programs and models in VR. To achieve the aforementioned goals, this investigation involved the following steps:

1. Studying existing solutions for the integration of parametric solutions in VR.
2. Developing a methodology that integrates textual AD in VR.
3. Outlining the required building blocks to implement the methodology.
4. Evaluating the methodology with exploratory user testing.

The previous section reviewed relevant literature on the integration of parametric design solutions with VR. In the following section, we explain and illustrate the proposed methodology along with the required components for its implementation. The methodology is then evaluated through exploratory exercises that aim to assess the usability of the implemented system, and the time gains and interaction advantages of using VR in the context of algorithmically developed projects. We conclude by drawing some final considerations on the results of the evaluation and outlining future research paths.

4. Algorithmic Design in Virtual Reality

ADVR aims to aid the algorithmic design process by integrating live coding in VR. In this workflow, architects use a Head Mounted Display (HMD) and an AD tool integrated in a VE to code their designs while immersed in them. In the VE, the generated design is concurrently updated in accordance with the changes made to its algorithmic description. Seeing these updates in near real time allows designers to conduct an iterative process of concurrent programming and visualization of the generated model in VR, enhancing the project with each cycle.

Figure 3 presents a conceptual scheme of this loop: architects develop the algorithmic description of the design using an Interactive Development Environment (IDE) or a programming editor to input the coding instructions into the AD tool, which then generates the corresponding model. From the VE designers, then, evaluate the results and, possibly, modify the algorithmic description from there, thus repeating the loop. Figure 4 presents a mock-up of the corresponding VR experience.

In order to provide an efficient coding platform in the VE that will enable this workflow, the following components are required: (1) an interactive AD tool that allows for the generation of complex architectural models, along with (2) a VR tool that can be coupled to the AD framework. This tool must be a sufficiently performative visualizer, such as a game engine, to guarantee near real-time feedback; (3) a mechanism that allows designers to code while immersed, i.e., an IDE or a programming editor; (4) text input mechanisms, as well as (5) language and IDE considerations; and, finally, (6) the ability to smoothly update the model even when its complexity hinders performance. The implementations we chose to pursuit for each of the numbered items are described below. Figure 5 presents the implementation workflow, featuring the chosen tools.

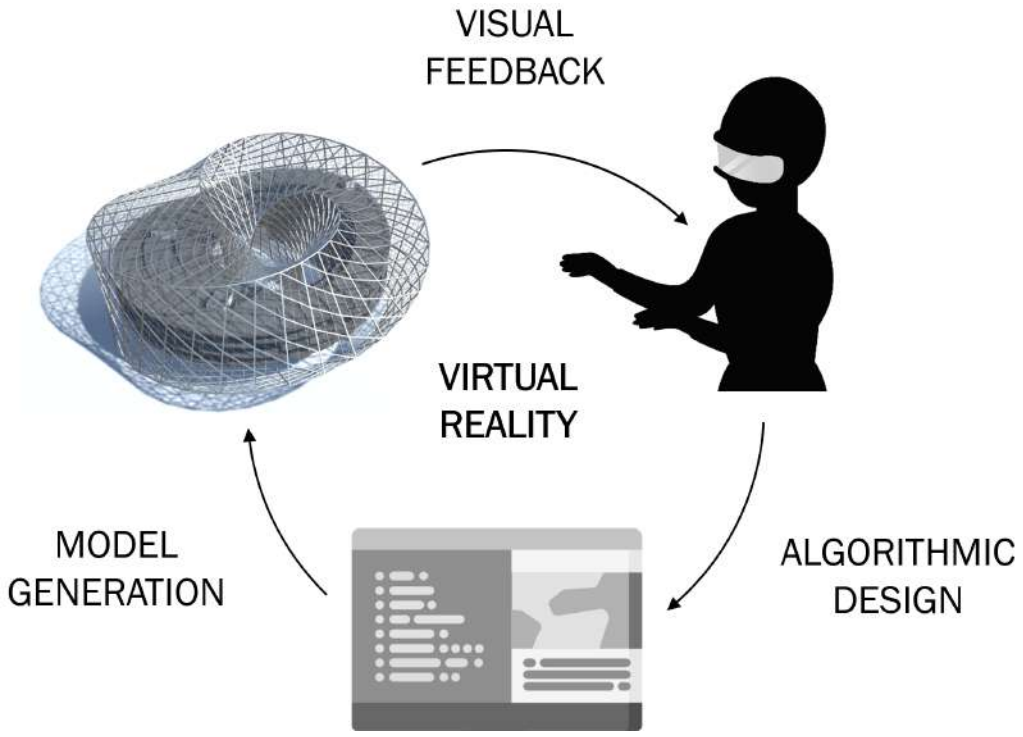

Figure 3. Conceptual scheme of the architect/program/model loop happening in VR: architects develop the algorithmic description of the design in VR, generating the corresponding model around them, whose visualization them motivates further changes to the description.

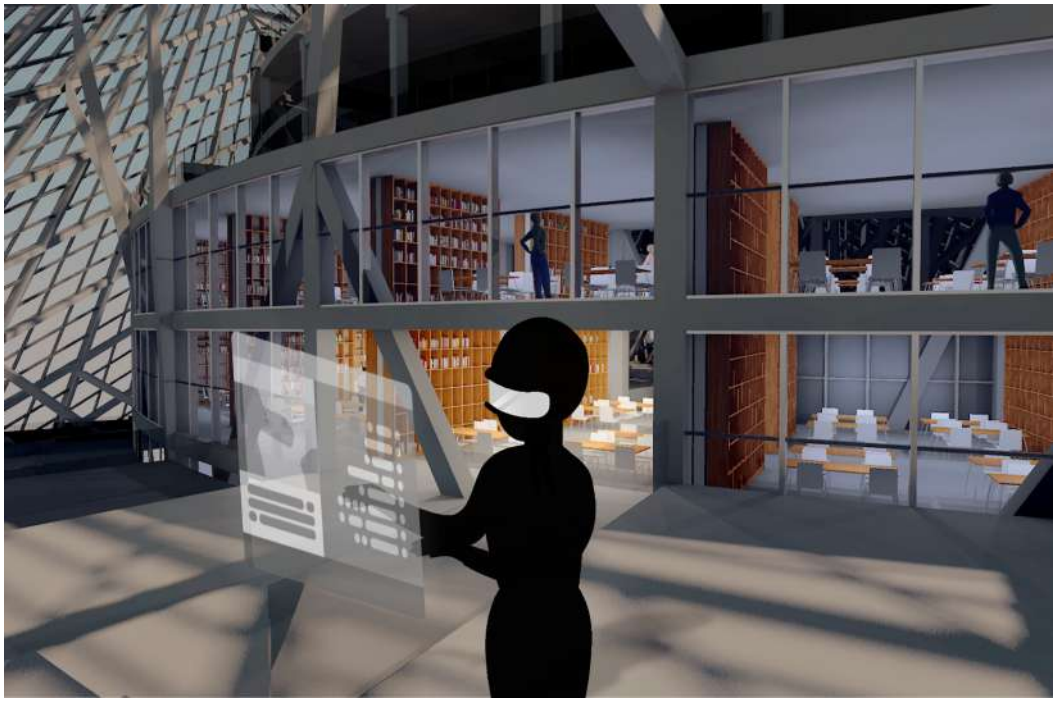

Figure 4. Mock-up of the ADVR experience: the architects live codes the algorithmic description of the design from within the VE.

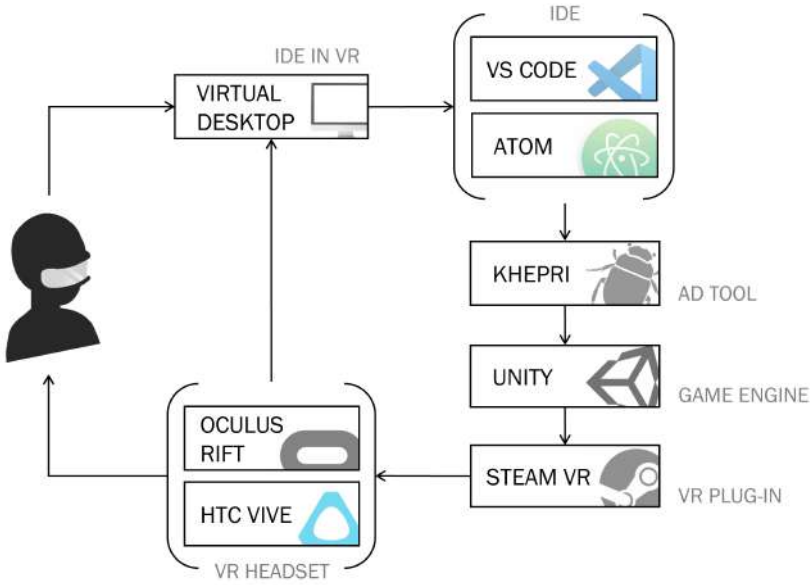

Figure 5. ADVR implementation: while immersed in the VE, the user accesses the IDE through the headset's virtual desktop application. The AD tool is responsible for translating the given instructions into operations recognized by the game engine, which is connected to the HMD through the VR plug-in.

4.1. AD Tool

Regarding the AD tool, we opted for Khepri [36], a portable AD tool that integrates multiple backends for different design purposes, namely Computer-Aided Design (CAD), Building Information Modeling (BIM), game engines, rendering, analysis, and optimization tools. The use of multiple tools along the development process is motivated by their different benefits: while CAD tools outperform the rest in free-form modelling [37]; BIM tools are essential for dealing with construction information [38]; game engines present a good alternative for fast visualization and navigation [39]; rendering tools offer realistic but

time-consuming representations of models for presentation; and, finally, analysis and optimization tools inform and guide the design process based on the model's performance [40].

4.2. VR Tool

Regarding the game engine, our choice fell upon Unity [41], since it provides good visual quality, including lighting, shadows, and physics; good visualization performance for average-scale architectural projects; platform independence; and availability of assets; as well as VR integration. Through the Steam VR plug-in, Unity communicates with the two tested headsets: Oculus Rift and HTC Vive. It must also be noted that, despite the fast response guaranteed by the game engine, the capacity for real-time feedback will always be conditioned by the model's complexity.

4.3. IDE Projection

For architects to be able to program from inside the VE, they need to access their preferred IDE while immersed in VR. To this end, we use the virtual desktop application provided by most HMDs (including the ones used for this implementation), which mirrors the user's desktop in VR, allowing the use of any application and, more specifically, of any IDE.

Figure 6 presents the workflow with the ANL model being live coded in VR. In this case, a change in the facade torsion parameter can be observed. Looking at the pictures in the first row, one may also observe that the mirrored desktop represents a partial visual blocker to the scene. However, it should be noted that this two-dimensional representation fails to convey the full 360° experience the user has in the VE. Moreover, the screen can be moved, scaled, or hidden at the designer's will, as shown in the second row of images in Figure 6.

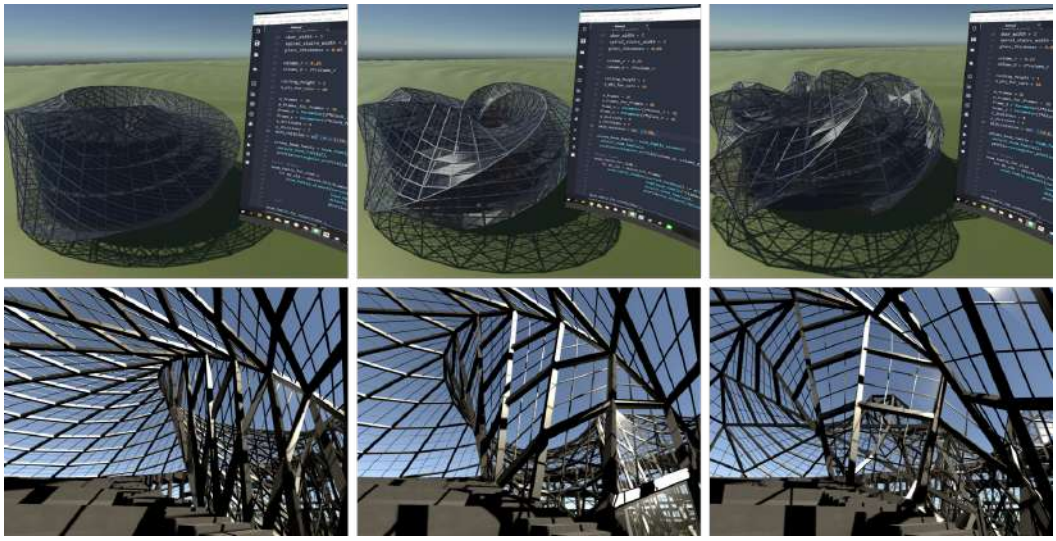

Figure 6. ADVR of the ANL model: manipulation of the facade torsion parameter (π, 2π, and 4π).

4.4. Text Input

Regarding textual input, there are several solutions currently available for the use of virtual and physical keyboards. Considering the result obtained in previous studies on the matter [42,43], we opted for the latter solution in this implementation. Figure 7 presents a use-case of the assembled workflow, showcasing both the IDE display along with the generated model of ANL in VR, and the interaction mechanisms in action, inside and outside the VE.

Figure 7. ADVR workflow: on the left, the VE where the model, the IDE and the responsive virtual keyboard are visible; on the right, the architect typing in the physical keyboard.

We stress that, despite the performance of the chosen solution in comparison to the remaining ones, it does not yet reach that of typing on a normal keyboard outside the VE, particularly for users who cannot touch type and, thus, heavily rely on both visual and haptic feedback. This question is particularly relevant in the present context, as the majority of programming architects are, in fact, non-experienced typists. Hence, other solutions for the problem must be sought and we believe the industry will soon provide them, as some interesting new concepts are already starting to emerge [44].

4.5. Language and Editor

In order to guarantee fast typing results, we opted for a dynamically-typed programming language, as these tend to be more concise than statically-typed ones. Although the latter offer more performance in run time and can detect static semantic errors, they force the user to provide type information, which is a lot more verbose. The chosen IDE can also help the user in the typing task, particularly, by providing automatic completion for names and for entire syntactical structures, such as function definitions. For this implementation, we used the Julia language running in the Atom editor with the help of the Juno plugin, a combination which considerably augments the user's typing speed. As shown in Figure 5, the Visual Studio Code editor was also tested, although the lack of a user friendly menu with shortcut buttons forced users to type more in order to run commands.

4.6. Model Update

ADVR takes advantage of game engines' ability to efficiently process geometry. As a result, we can generate large-scale models, as is the case of the ANL, in a matter of seconds. When applying changes to the model in VR, these seconds are, nevertheless, troublesome, since the AD tool deletes and (re)generates the entire model in each iteration, regardless of the number of changes applied. Consequently, as the model grows, a small lag becomes noticeable and the sudden reshaping of the entire VE is disorienting and makes it difficult to understand the effects of the applied changes.

To solve these problems, we implemented a multi-buffering approach that keeps the user in an outdated but consistent model, while a new model is invisibly being generated. When finished, the new model replaces the old one, allowing the user to immediately visualize the impact of the changes. It is also possible to have several models available simultaneously, in different buffers, for the user to switch back and forth between them, facilitating comparisons and improving the decision-making process. Figure 6 illustrates this by showing two different views of the variations created.

5. Evaluation

In order to evaluate the framework, we began by assessing the usefulness of VR in the process of architectural creation, both from the architect's and the client's point of view; thus, removing any possible coding difficulty from the equation in this phase. For that purpose, we developed an exploratory exercise to evaluate the *time gains* allowed by the introduction of VR in the design process, as well as an experimental *interaction* exercise with algorithmically-designed models in VR. The third exercise, named *usability*, effectively joined all pieces together for the use of the ADVR workflow.

The first exercise—time gains—was conducted by a single subject, an architect comfortable with the programming language, the chosen AD tool, and the ANL project. The results were summarized in a time graph. The second exercise—interaction—involved 21 participants from two different backgrounds, architecture and computer science (see Figure 8). From this group, only a portion was selected for the third exercise—usability—since it required specialized knowledge in both the programming language and the AD tool used for this implementation. The results of the latter two exercises are based on post-activity user surveys and will be presented through a statistical analysis of the responses.

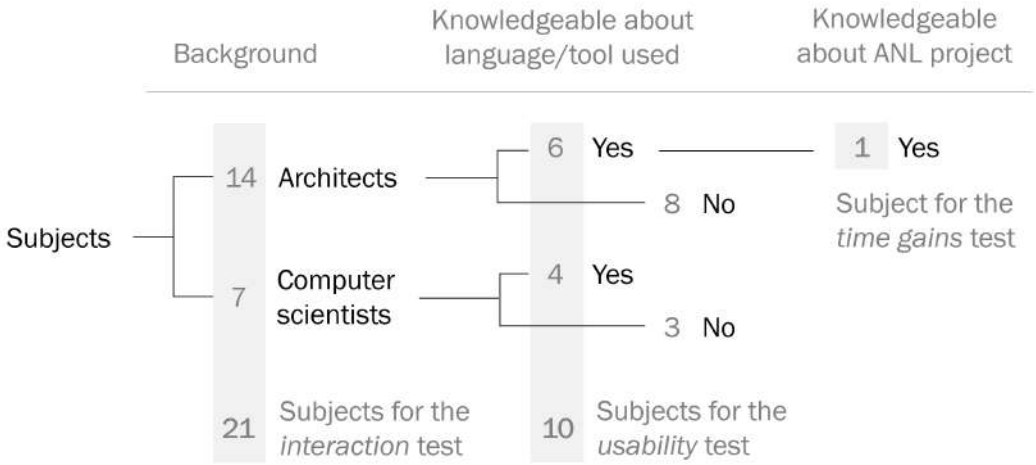

Figure 8. Selection of test subjects for the evaluation exercises.

Figure 9 illustrates the variables considered in these evaluations. For the first exercise, the only independent variable is the use of VR for showcasing, and we intend to measure its impact in time gains. The second exercise has two independent variables: the use of VR and the scale of the project. Here, our evaluation focuses on important concepts to architectural visualization, i.e., characteristics that an architectural representation method must offer to be useful to the practice, namely, spatial and scale perception, navigability (in the case of 3D models), and realism. Still in this exercise, we also question the participants regarding the usefulness of VR in either scale, as a showcasing or design tool, and how likely would they use it themselves. The third exercise has the use of ADVR as independent variable, and intends to assess its effect on the design process in terms of fun, speed, productiveness, and advantages, as well as its usefulness (similarly to VR), and how likely would subjects use it in their design workflows. The following subsections describe the three exercises in detail.

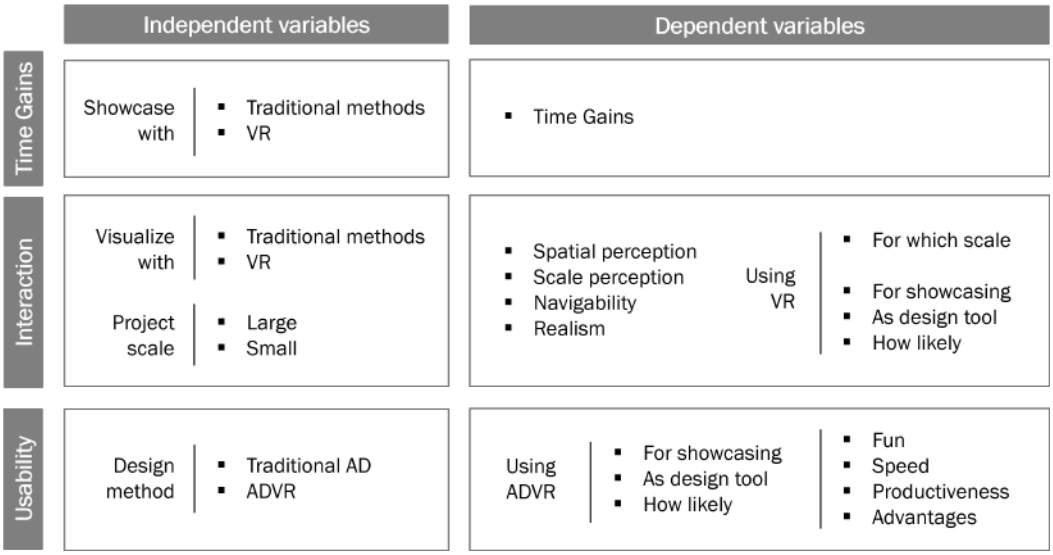

Figure 9. Independent and dependent variables of the study.

5.1. Time Gains

The typical client/architect communication cycle involves (1) showcasing ideas, receiving feedback on them; (2) applying changes to the design; and (3) producing new elements for another iteration of this cycle.

Comparing ADVR to a more traditional AD process that does not resort to VR technologies, we consider that stage 1 takes about the same amount of time and/or effort in either case, although the latter does not benefit from this alternative showcasing method. The same happens with stage 2, since it also relies on algorithmic means to apply changes to the design. In both cases, the time required will vary in accordance with the changes to be applied and/or to showcase.

However, in stage 3, the production of presentation elements, the required times may differ significantly. While the usual AD approach would rely on traditional representation methods, such as physical models, technical drawings such as plans and sections, and rendered images, which take longer to produce; in ADVR we only need to regenerate the model after applying the changes and the client can immediately visualize the 3D model with the added advantage of immersion and real-time rendering. If the changes proposed are small enough, the entire cycle can even occur live, with the architect altering the building instantaneously while the client is still in VR, seeing the project change around them on request. This workflow considerably improves the communication between the client and the architect because it promotes the quick showcase of new design ideas with immediate feedback on the client's part. The typical communication cycle therefore becomes orders of magnitude faster.

To put this thesis in practice, we asked a group of architects to suggest plausible changes to the ANL model. Figure 10 presents the changes they suggested and the time it took a single architect to implement them. The subject, in this case, was an architect who was comfortable with the programming language, the chosen AD tool, and the ANL project. The proposed changes are divided in two groups: predefined and non-predefined parameters. The former comprises existing parameters in the program, i.e., the changes anticipated by the architect when developing the algorithm. The latter includes any other modifications that, not having been anticipated, take more time to implement. The time for each of the proposed changes is divided in 6 phases:

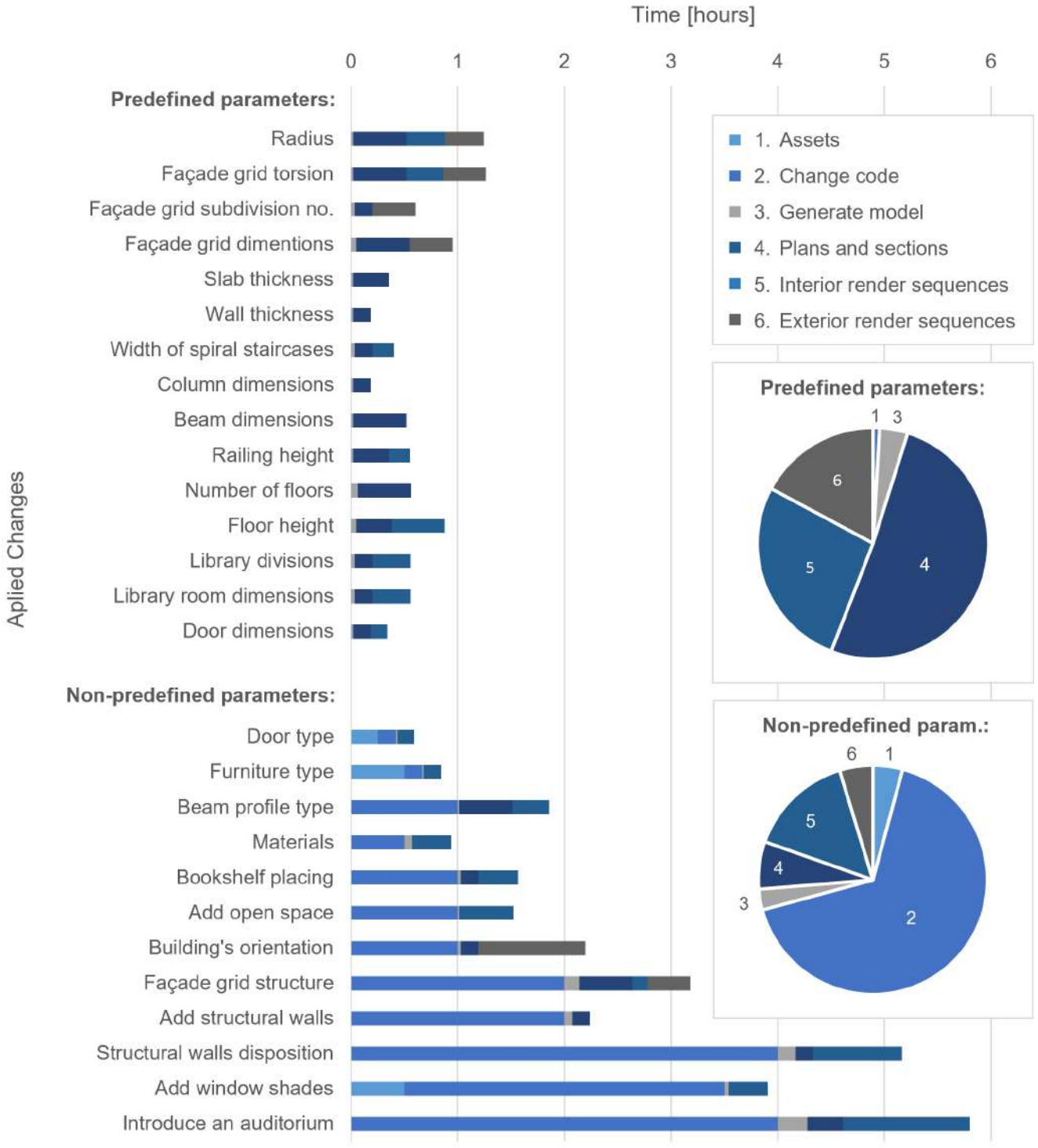

Figure 10. Time graph for the changes applied to the ANL mode: time spent on each of the 6 tasks for the case of predefined (on **top**) and non-predefined parameters (on the **bottom**).

- (1) Assets: time spent collecting and arranging assets for Unity's library, which was required only for changes that involved new objects.
- (2) Change code: time dedicated to the application of changes to the code. Changes applied to predefined parameters are virtually instantaneous.

- (3) Generate model: the average generation time for a complete ANL model is 50 s in Unity. When iterating over specific parts of the code, this time can be reduced by generating only the parts of the model involved. The major discrepancies visible in this category are due to the number of times the model was generated during more complex change processes, such as the introduction of an auditorium, for instance, which require the programmer to iteratively (re)generate the model in order to evaluate the applied changes.
- (4) Production of static elements for presentation (plans and sections): Since we are working within a multi-backend algorithmic workflow, technical drawings are automatically extracted from the BIM version of the model. However, and despite the considerable gains this automation provides, the time spent generating model in phases 4 to 6 is still relevant. The ANL model must be generated (at least) twice: once in the game engine to produce rendering sequences, and another in the BIM backends to produce drawings. Moreover, most plans and sections also required some additional manual work.
- (5,6) Production of static elements for presentation (interior and exterior render sequences): render sequences or films are automated in the code and generated in the game engine. To reduce the time spent in this stage we made sure to generate only the sequences that were impacted by the changes applied.

The pie charts in Figure 10 show the total amount of time spent in each phase for the two collections of changes applied. We can verify that, for the case of predefined parameters, most of the time is spent on the creation of elements for presentation (phases 4, 5, and 6), whereas the time required to change the program is minimal. The same cannot be verified with non-predefined changes. Cosmetic changes, such as modifying furniture elements (Figure 11), also require small changes to the program, but most of the time is spent collecting and modifying assets, and redoing renders. More complex and/or unanticipated changes take a lot more time to implement in the program, although this is highly dependent on the architect's working speed. They also generally require the remake of presentation elements, depending on the amount of geometry affected by the change. The most dramatic case from our collection of changes was the introduction of an auditorium in the library volume (Figure 12), which required breaking the regularity of the wall distribution in that volume.

Figure 11. Furniture material and layout change in the ANL model.

In either case, using VR for showcasing instead of the more traditional representation methods, phases 4, 5, and 6 could be skipped altogether, which represents a considerable time gain. In our analysis, the accumulated time of these three tasks had a minimum value of 10 min (in the furniture type change) and a maximum of one hour and 41 min (with the introduction of an auditorium). Furthermore, while long-lasting VR immersion periods would be required for the application of the second group of changes, in the case of predefined parameters, architects can rely not just on VR but also on ADVR, since fast changes can be performed live within seconds.

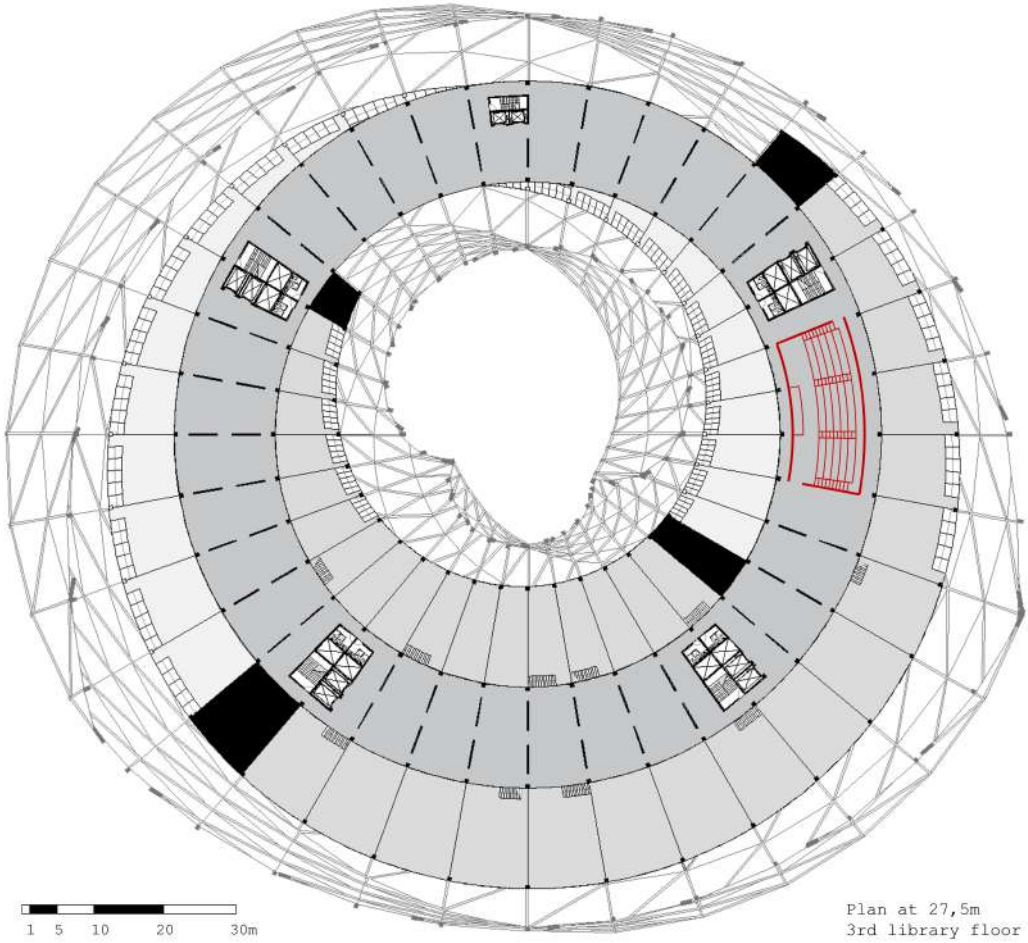

Figure 12. Introduction of an auditorium in ANL's library volume.

5.2. Interaction

The previous exercise assessed the time gains granted by the use of VR in AD. The following one placed the emphasis on the interaction process between user and model, within the VE. We asked a group of participants to analyze plans, sections, and render images of both a large-scale and a small-scale project. They were encouraged to try and understand the projects' spatiality only. After this, they were placed in the VE, where they could navigate both models and do the same analysis.

The large-scale project chosen was, again, the ANL. The small-scale project was an acoustic intervention to the ceiling of a classroom. The intervention consisted of a grid of wooden panels hanging from the ceiling. The grid had a ripple effect pattern influenced by a set of attractor points spread around the room. The VR exercise included the possibly to change the project's layout on demand by adding or removing attractors. Subjects did not interact with the AD description directly, however. They simply placed themselves on a chosen location in the model and verbalized their request, much like a client might do in a showcasing session. The process is explained in Figure 13: subjects inside the model in VR would ask for the addition or removal of attractor points above them; a change that took no more than a few seconds (the time required to regenerate the model in the game engine). The total amount of time spent on this exercise was entirely up to the subject. In general, navigation and interaction with the two models did not exceed 15 min.

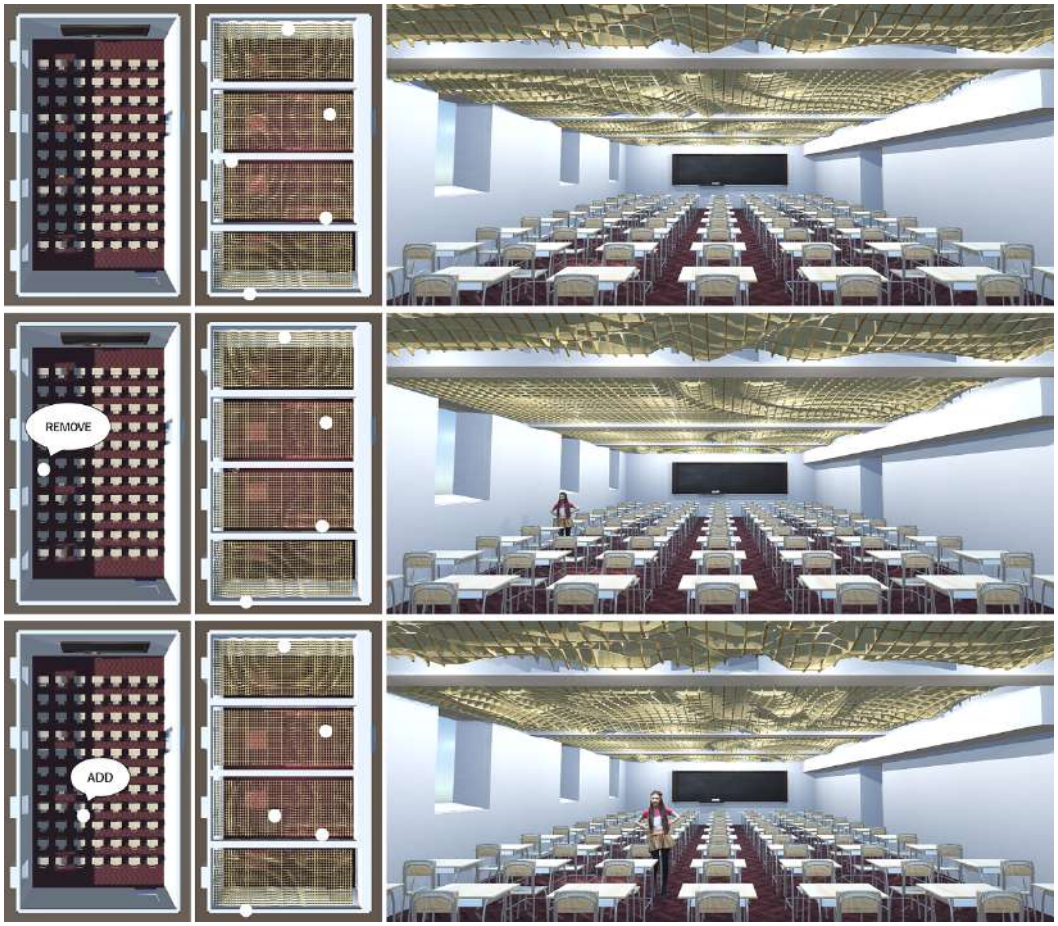

Figure 13. From left to right: top view of the classroom (1) with the user's location and (2) showing the intervention and the active attractors' location; and (3) render of the resulting solution. From top to bottom: (1) default solution; (2) removal of an attractor; and (3) addition of an attractor.

The exercise ended with an inquiry of twelve questions using the Likert scale rating system (1 to 5) to measure agreement, quality, importance, usefulness, or likelihood, depending on the variable, and one qualitative question where subjects could comment freely on the experience. Although Likert-type answers are typically considered ordinal data, in this analysis, we will treat them at the interval level, assuming an even distancing between categories. Given the large number of variables under analysis, for better comprehension, we choose to present the central tendency through the average value for the survey responses, complemented with the standard deviation to account for response variability.

The statistical analysis of the answers to the inquiry is presented in Figures 14 and 15: the graphs show the average result and standard deviation of the responses for each variable. The test group included 21 subjects from different backgrounds: young architects, architecture students, as well as computer science engineering students. The graphs separate user responses in two groups, considering their backgrounds. We use the letter A to represent the 14 subjects with a background in architecture and the letter O for all others. The results of the analysis will be presented with this letter, followed by the pair average/standard deviation.

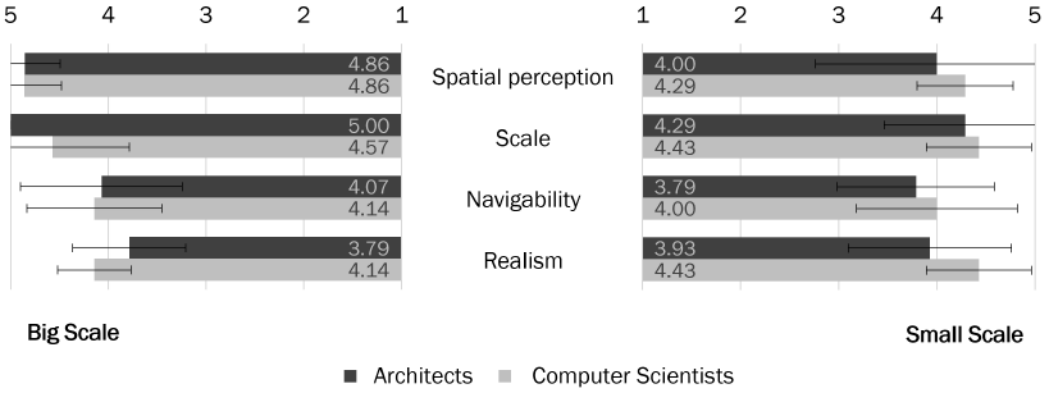

Figure 14. User study results on VR interaction with a small and a large-scale architectural project: scale, spatial perception, navigability, and realism.

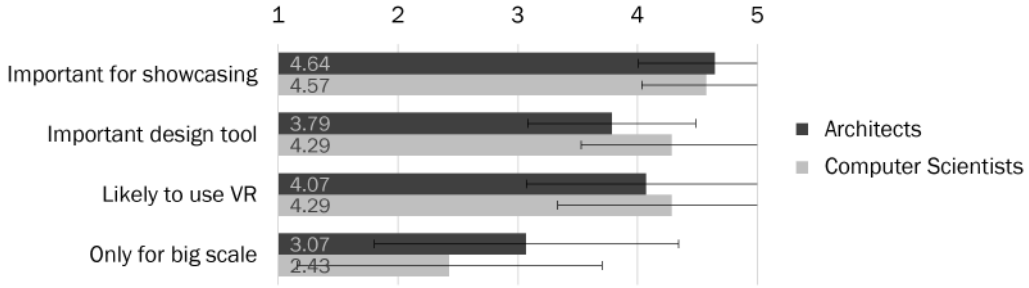

Figure 15. User study results on VR interaction with a small and a large-scale architectural project: showcase, design, and use.

5.2.1. Scale, Spatial Perception, Navigability, and Realism

Subjects tended to agree that both the notion of scale and spatial perception increased in VR, particularly for the large-scale case. Big scale was evaluated with A: 4.86/0.36; O: 4.86/0.38, regarding spatial perception, and with A: 5/0; O: 4.57/0.79, regarding scale, while small scale was evaluated with A: 4/1.24; O: 4.29/0.49 and A: 4.29/0.83; O: 4.43/0.53, respectively. The realism of the models had slightly poorer results, with subjects voting, on average, on better navigability in ANL (A: 4.07/0.83; O: 4.14/0.69 for big scale, vs. A: 3.79/0.80; O: 4/0.82 for small scale), and more realism in the classroom (A: 3.79/0.58; O: 4.14/0.38 for big scale, vs. A: 3.93/0.83; O: 4.43/0.53 for small scale).

Regarding navigability, this outcome was expected: since both models were to be evaluated in the same take, we decided to optimize the VR controllers for large-scale displacement, advising subjects to be gentle on the motions inside the small room. Regarding realism, we can imagine two possible explanations: (1) due to its large dimension, in the ANL, subjects were tempted to use the free-fly mode to explore the project, instead of the walk mode, which may have contributed to the feeling of non-realism; and (2) while the area of the ANL project is considerably vaster, only the library area was furnished, which resulted in empty (ghosted) rooms/areas.

5.2.2. Showcase, Design, Use

We asked our subjects if they felt VR was, hence, an important tool for showcasing and designing, if they were likely to use it themselves, and if they felt it only compensated for big-scale projects. The responses tended towards agreement; exception made for the use of VR in big-scale projects. All subjects voted for more usefulness in showcasing designs (A: 4.64/0.63; O: 4.57/0.53 for showcasing, vs. A: 3.79/0.7; O: 4.29/0.76 for design, and A: 4.07/1; O: 4.29/0.95 for probability of future use). Predictably, non-architects voted towards the use of VR in both small- and large-scale (O: 2.43/1.27), while architects were mostly neutral (A: 3.07/1.27). These results also suggest people from other backgrounds

may prefer to visualize projects in VR in any case, while professional architects may find they can very well infer a project's spatiality from traditional representation means alone. In either case, standard deviation is higher than in other responses, which means subjects tended to disagree more on this topic.

From the qualitative question, we acquired important feedback, namely suggestions on how to (1) handle controller sensitivity for navigation and (2) render more realistic scenarios, (3) positive comments on the usefulness of VR, and (4) an interesting pattern of motion sickness complaints, with 58% of our female subjects reporting dizziness, nausea or headaches.

5.3. Usability

The previous exercises purposely isolated the benefits of VR from the burdens of the coding task. In this last exploratory exercise, we evaluated the perceived usability of the ADVR methodology to designers. The test was conducted with a more reduced number of participants, as the exercise required specialized knowledge in both the programming language and the AD tool used for this implementation. In a total of ten subjects, 6 were architectural practitioners trained in AD, and 4 were computer science MSc students.

Figure 16 shows (on the left) some of the images contained in the exercise's briefing, and (on the right) the view our users had if they completed the exercise. The goal was to create, from scratch, an algorithmic description of a semi-random pagoda city using the ADVR workflow. The exercise was intentionally simple, in order to last no longer than 20 min, although some users ended up remaining in VR for over an hour, navigating the model and experimenting variations to their algorithms. The task was subdivided in intermediate goal functions (i.e., creating one story, one pagoda, one random-pagoda, and a city), but it was not mandatory to complete the entire briefing.

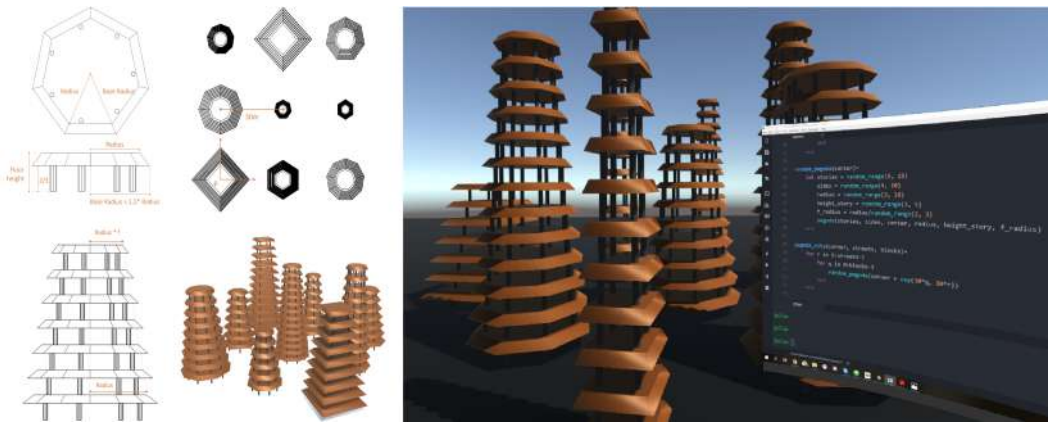

Figure 16. Random pagoda city exercise using the ADVR workflow: on the **left**, images from the briefing provided and, on the **right**, the goal of the exercise.

This experiment also ended with a survey, whose answers are summed up in Figure 17. We anticipated architects, as non-experienced typists, would suffer from serious decrease in typing performance; hence the decision of including computer scientists in the experiment, as they typically are more experienced typists.

As expected, the voting average for the architecture subjects was a lot more pessimistic than their counterparts. Users were asked if the ADVR experience was more fun, slower, more productive, and more advantageous than traditional AD approaches. Architects mostly agreed on speed (A: 4.17/1.6), while computer scientists mostly disagreed (O: 2.25/1.26). The latter also voted for more fun (O: 4.25/0.96), while architects remained fairly close to neutral on average (A: 3.33/1.03), on account of the typing difficulty, according to the comments left on the open question. Architects also voted ADVR as slightly less productive (A: 2/0.63), but equally advantageous (A: 3/0.63). They agreed on its usefulness for showcasing (A: 4.83/0.41), and also for design, although less enthusiastically

(A: 3.67/1.03). Only a few stated they would likely use ADVR in their own design processes (A: 3.17/1.6). In this exercise, we also verified a general increase in the standard deviation, meaning the topic tended to divide opinions a bit more.

On the open question, besides the typing difficulty, comments were also issued on the added difficulty in reading program text in VR for poor-sighted users.

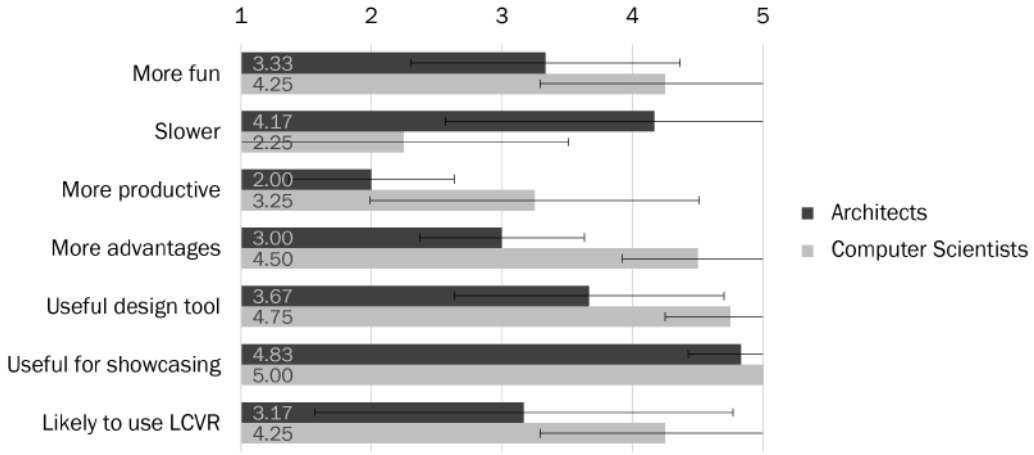

Figure 17. User study results on the usability of the ADVR workflow.

6. Discussion

In this paper, we proposed a methodology for live coding AD in VR: ADVR. We used a game engine to allow models to be generated and updated in near real time, as architects modify the algorithmic descriptions of their designs. The coupling with VR allowed us to take AD one step further, towards an immersive live coding experience: architects can program while inside the VE, virtually inhabiting the building they are designing, which is changing around them to reflect their design decisions.

The evaluation of the methodology already identified some frailties in the workflow. The *time gains* and *interaction* analyses yielded the expected results, with very positive feedback from users. The *usability* exercise, however, had low ratings on average. By making use of traditional text input mechanisms to code in the VE, the ADVR methodology was severely hampered. Furthermore, it suffers from current VR technology's limitations regarding motion sickness and eye fatigue. These issues are currently being addressed by the VR industry and, thus, we expect to see them solved in the near future.

A bias disclaimer must be made regarding the exploratory exercises presented, as they relied on a small and non-probability sample, which results in a weaker inference about a more general population of potential users. It must also be noted that, although the Likert scale rating system is widely used to acquire fine-grained insights into subjects' opinion, its inherent subjectiveness may also compromise the validity of the conclusions. Withal, other studies corroborate our findings [45].

Considering the presented experimental analysis, we can confirm that the introduction of VR in the AD paradigm can provide a more tangible interaction with the generated model. In this sense, ADVR has the capacity to improve the design experience and the communication of architectural projects. Live coding offers immediate feedback to changes in algorithmic descriptions and, combined with VR, motivates designers to engage in more exploratory design actions. Communication with clients or other stakeholders is also improved, as their understanding of the project is closer to the built reality itself. Clients can not only visualize projects to scale, but also see the project change around them in real time, as they ideate back and forth with the architect.

Given the results of our experiments, we believe ADVR, as it is currently implemented, might not yet be suitable for initial design stages, since it still has drawbacks in comparison to the traditional programming workflow, such as the reduced typing speed for users unable to touch type. This means designers should not necessarily engage in live coding in the VE until they have a fairly complete version of their model, as the imposed trade-offs might not compensate for early-stage or small-scale models. On the other hand, with relatively complex models, the spectacle added by immersiveness provides a great source of motivation and inspiration for experimentation. Thus, we believe the ADVR approach here presented provides a good contribution for subsequent experimentation phases.

We do, however, foresee future developments to adapt ADVR to initial stages of the design process as well:

1. Code input mechanisms: Taking into account the issued comments on the difficulty users have in typing using an occluded keyboard, other text input solutions may be considered. We can already find several tools that aid the coding task, such as dynamic code completion mechanisms and voice input technology. Handwriting recognition will likely improve in the years to come, considerably reducing the current level of discomfort in using these technologies. In the near future, we will also likely see the results of currently undergoing brain-computer interface research [46], which can dramatically speed up the communication process. Until then, simple typing training may also render considerable performance gains. Image resolution, motion sickness, and eye fatigue belong in another category of issues soon to be addressed by the VR industry.
2. Code Visualization options: To enable the coding task while in VR, the presented implementation considered mirroring the user's desktop. However, more interesting solutions may be thought of, such as having only selected parts of the program projected onto the VE along with the generated geometry. Such an approach would require the program to recognize the parcel of code responsible for generating the objects selected by the user in the VE and projecting that parcel onto the VE as well. This entails substantial investment in finding intelligent ways of managing program traceability and showcasing the control flow of the program, but the solution would likely improve the exploration of the 3D space and it would also be more synthetic, which might help less experienced programmers orient themselves in the program.
3. Automatic Algorithmic Design: Another promising path, which we intend to explore in the near future, is to manually model geometry in VR, which the AD tool automatically converts to algorithmic descriptions [47,48]. There is plenty of work done on manual modeling in VR, although most of the solutions rely on cuboid or voxel modeling. The challenge lies in the conversion of the modeled geometry into code, namely in: (a) the automatic recognition of the 3D geometry; (b) metaprogramming, that is, having the program generate another program on its own, describing the 3D geometry; and (c) refactoring the generated code into an understandable program. If successfully implemented, this path has the potential to integrate in the architect's design workflow, not only ADVR and traditional AD, but also the traditional modeling approach as a quasi-physical experience in VR. This might prove a very interesting design tool for architects to experiment with, as well as for clients immersed in the VE to express their ideas in a way that is comfortable and intuitive for them.

With the presented issues overcome, the ADVR methodology can not only benefit early design stages but also learning scenarios. Live coding alone has been shown to facilitate the comprehension of programs [20]: the ability to make changes in the program and immediately visualize their impact on the generated geometry allows programmers to easily relate program parts to the geometry each one generates. With the added impressiveness of the VR medium, this ability can have a very positive effect on AD learning. An ADVR methodology adapted to more conceptual design stages will, therefore, appeal to less experienced programmers as well.

7. Conclusions

In this paper, we proposed the Algorithmic Design in Virtual Reality (ADVR) methodology, which implies live coding Algorithmic Design (AD) descriptions in Virtual Reality (VR). In an immersive live coding experience, architects can program from the virtual environment while virtually inhabiting the building that is changing around them.

The evaluation of the methodology identified some frailties in the workflow. Despite the very positive feedback regarding time gains and interaction, usability had lower ratings, in large part caused by the limitations of (1) traditional text input mechanisms to code in VR and (2) current VR technology's related motion sickness and eye fatigue. We believe that, soon, these problems will be addressed by the VR industry.

Nevertheless, we can confirm that the integration of VR with AD provides a more tangible interaction with the generated model. However, to provide the immediate feedback required by VR, AD needs live coding. This combination motivates designers to engage in more exploratory design actions and makes ADVR a promising endeavor. However, in its current form, it might not be suitable for early design stages or small-scale models, as the imposed trade-offs might not compensate. On the other hand, with relatively complex models, the spectacle added by immersiveness provides a great source of motivation and inspiration for experimentation.

We plan to further develop the ADVR methodology, making it easier to use and capable of supporting initial stages of the design process as well. To this end, we plan to adopt text input solutions that aid the coding task, such as dynamic code completion mechanisms and voice input technology or, in a not-so-distant future, by exploring brain-computer interfaces. To speed-up code production, we plan to explore code-generation techniques that elaborate algorithmic descriptions from hand-made models. Finally, considering that the presented evaluation of the workflow was conducted informally and on a small number of participants, we also plan on conducting a more extensive user-experience evaluation, as well as more objectively validating the proposed methodology by comparing it to similar solutions.

Author Contributions: Conceptualization, R.C.-B. and A.L.; methodology, R.C.-B.; software, R.C.-B. and A.L.; validation, R.C.-B. and A.L.; formal analysis, R.C.-B.; investigation, R.C.-B. and A.L.; resources, A.L.; data curation, R.C.-B.; writing—original draft preparation, R.C.-B.; writing—review and editing, R.C.-B. and A.L.; visualization, R.C.-B.; supervision, A.L.; project administration, A.L.; funding acquisition, R.C.-B. and A.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by national funds of *Fundação para a Ciência e a Tecnologia* (FCT) (references UIDB/50021/2020, PTDC/ART-DAQ/31061/2017) and a PhD grant under contract of FCT (DFA/BD/4682/2020).

Institutional Review Board Statement: Ethical review and approval were waived for this study, since the user study here presented began as an informal evaluation of the workflow while it was still being developed. The research members in the authors' lab began experimenting with the workflow and, somewhat organically, the testing extended to more participants who volunteered. The experiments were not recorded and the study is anonymous, hence no privacy or data protection issues were raised; it was entirely voluntary; it did not include any vulnerable category of individuals or persons unable to give conscious consent; and it involved no threats to the health or safety of the participants. When the question of ethical approval was raised, reasonable deadlines to submit a validation request to the institution's ethics commission were past. Nevertheless, the authors consider that the experiments were done according to the ethics standards available at their institution.

Informed Consent Statement: Participants were given a form with detailed information on the aims, methods and implications of the study, and, although not formally registered, informed consent was collected in advance from all subjects.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to thank Guilherme Santos for the assistance provided in conducting the informal user studies.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AD	Algorithmic Design
ADVR	Algorithmic Design in Virtual Reality
ANL	Astana National Library
BIM	Building Information Modeling
CAD	Computer-Aided Design
HMD	Head Mounted Display
IDE	Interactive Development Environment
VE	Virtual Environment
VR	Virtual Reality

References

1. Gobbetti, E.; Scateni, R. Virtual reality: Past, present and future. *Stud. Health Technol. Inform.* **1998**, *58*, 3–20. [PubMed]
2. Portman, M.; Natapov, A.; Fisher-Gewirtzman, D. To go where no man has gone before: Virtual reality in architecture, landscape architecture and environmental planning. *Comput. Environ. Urban Syst.* **2015**, *54*, 376–384. [CrossRef]
3. Sutherland, I.E. The Ultimate Display. *Proc. Inf. Process. (IFIP) Congr.* **1965**, *2*, 506–508.
4. Whyte, J. Industrial applications of virtual reality in architecture and construction. *J. Inf. Technol. Constr. (ITcon)* **2003**, *8*, 43–50.
5. Dorta, T.; Pérez, E. Immersive Drafted Virtual Reality: A new approach for ideation within virtual reality. In Proceedings of the 25th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA), Louisville, Kentucky, USA, 12–15 October 2006; pp. 304–316.
6. Gu, N.; Kim, M.J.; Maher, M.L. Technological advancements in synchronous collaboration: The effect of 3D virtual worlds and tangible user interfaces on architectural design. *Autom. Constr.* **2011**, *20*, 270–278. [CrossRef]
7. Schnabel, M.A. The immersive virtual environment design studio. In *Collaborative Design in Virtual Environments*; Wang, X., Tsai, J.J.H., Eds.; Intelligent Systems, Control and Automation (ISCA): Science and Engineering; Springer: Dordrecht, The Netherlands, 2011; Volume 48, pp. 177–191.
8. Wang, X. Mutually augmented virtual environments for architectural design and collaboration. In Proceedings of the 12th International Conference on Computer Aided Architectural Design Futures (CAAD Futures), Sydney, Australia, 11–13 July 2007; Springer: Dordrecht, The Netherlands, 2007; pp. 17–29.
9. Wang, P.; Wu, P.; Wang, J.; Hung-Lin, C.; Wang, X. A Critical Review of the Use of Virtual Reality in Construction Engineering Education and Training. *Int. J. Environ. Res. Public Health* **2018**, *15*, 1204. [CrossRef]
10. Davila Delgado, J.M.; Oyedele, L.; Demian, P.; Beach, T. A research agenda for augmented and virtual reality in architecture, engineering and construction. *Adv. Eng. Inform.* **2020**, *45*, 101122. [CrossRef]
11. Alizadehsalehi, S.; Hadavi, A.; Huang, J.C. From BIM to extended reality in AEC industry. *Autom. Constr.* **2020**, *116*, 103254. [CrossRef]
12. Anderson, L.; Esser, J.; Interrante, V. A virtual environment for conceptual design in architecture. In Proceedings of the Eurographics Workshop on Virtual Environments (EGVE), Zurich, Switzerland, 22–23 May 2003; Association for Computing Machinery: New York, NY, USA, 2003; pp. 57–63.
13. de Klerk, R.; Duarte, A.; Medeiros, D.; Pinto Duarte, J.; Jorge, J.; Lopes, D. Usability studies on building early stage architectural models in virtual reality. *Autom. Constr.* **2019**, *103*, 104–116. [CrossRef]
14. Burry, M. *Scripting Cultures: Architectural Design and Programming*; Architectural Design Primer; John Wiley & Sons, Inc.: West Sussex, UK, 2011.
15. Terzidis, K. *Algorithmic Architecture*; Architectural Press: New York, NY, USA, 2006.
16. Lee, J.; Gu, N.; Jupp, J.; Sherratt, S. Evaluating Creativity in Parametric Design Processes and Products: A Pilot Study. In *Design Computing and Cognition (DCC'12)*; Gero, J.S., Ed.; Springer: Dordrecht, The Netherlands, 2014; pp. 165–183.
17. Woodbury, R. *Elements of Parametric Design*; Routledge: London, UK, 2010.
18. Davidson, S. Grasshopper: Algorithmic Modeling for Rhino. 2021. Available online: https://www.grasshopper3d.com (accessed on 27 December 2021).
19. Alfaiate, P.; Caetano, I.; Leitão, A. Luna Moth: Supporting Creativity in the Cloud. In Proceedings of the 37th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA), Cambridge, MA, USA, 2–4 November 2017; pp. 72–81.
20. Rein, P.; Ramson, S.; Lincke, J.; Hirschfeld, R.; Pape, T. Exploratory and Live, Programming and Coding: A Literature Study Comparing Perspectives on Liveness. *Program. J.* **2018**, *3*, 1:1–1:33. [CrossRef] [PubMed]
21. Panya, D.S.; Kim, T.; Choo, S. A Methodology of Interactive Motion Facades Design through Parametric Strategies. *Appl. Sci.* **2020**, *10*, 1218. [CrossRef]
22. Autodesk, Inc. Dynamo: Open Source Graphical Programming for Design. 2020. Available online: https://dynamobim.org/ (accessed on 27 December 2021).

23. Prime Software Systems. PRIMITIVE: VR Collaboration for Distributed Teams. 2019. Available online: https://primitive.io/ (accessed on 27 December 2021).
24. James, P. Watch: 'LogiX' is an Impressive Multi-User Visual Programming Interface for VR. 2017. Available online: https://www.roadtovr.com/watch-logix-impressive-multi-user-visual-programming-interface-vr/ (accessed on 27 December 2021).
25. Iannini, L. Rumpus. 2016. Available online: https://store.steampowered.com/app/458200/Rumpus/ (accessed on 27 December 2021).
26. Schutz, M.; Wimmer, M. Live Coding of a VR Render Engine in VR. In Proceedings of the 26th Conference on Virtual Reality and 3D User Interfaces, Osaka, Japan, 23–27 March 2019; pp. 1150–1151.
27. Elliott, A.; Peiris, B.; Parnin, C. Virtual reality in software engineering: Affordances, applications, and challenges. In Proceedings of the 37th International Conference on Software Engineering (ICSE), Florence, Italy, 16–24 May 2015; Volume 2, pp. 547–550.
28. Krahn, R. CodeChisel3D: Live Programming with three.js and webVR. 2015. Available online: https://robert.kra.hn/past-projects/live-programming-with-three-and-webvr.html (accessed on 27 December 2021).
29. Clergeaud, D.; Roo, J.S.; Hachet, M.; Guitton, P. Towards Seamless Interaction Between Physical and Virtual Locations for Asymmetric Collaboration. In Proceedings of the 23rd Symposium on Virtual Reality Software and Technology (VRST), Gothenburg, Sweden, 8–10 November 2017; pp. 17:1–17:4.
30. Coppens, A.; Mens, T.; Gallas, M.A. Parametric Modelling Within Immersive Environments: Building a Bridge Between Existing Tools and Virtual Reality Headsets. In Proceedings of the 36th Education and Research in Computer Aided Architectural Design in Europe (eCAADe) Conference, Lodz, Poland, 17–21 September 2018; pp. 711–716.
31. Hawton, D.; Cooper-Wooley, B.; Odolphi, J.; Doherty, B.; Fabbri, A.; Gardner, N.; Haeusler, M.H. Shared Immersive Environments for Parametric Model Manipulation—Evaluating a Workflow for Parametric Model Manipulation from Within Immersive Virtual Environments. In Proceedings of the 23rd International Conference on Computer-Aided Architectural Design Research in Asia (CAADRIA), Beijing, China, 17–19 May 2018; pp. 483–492.
32. Castelo-Branco, R.; Leitão, A.; Santos, G. Immersive Algorithmic Design: Live Coding in Virtual Reality. In Proceedings of the 37th Education and research in Computer Aided Architectural Design in Europe (eCAADe) Conference, Porto, Portugal, 11–13 September 2019; Volume 2, pp. 455–464.
33. Caetano, I.; Santos, L.; Leitão, A. Computational design in architecture: Defining parametric, generative, and algorithmic design. *Front. Archit. Res.* **2020**, *9*, 287–300. [CrossRef]
34. Janssen, P. Visual Dataflow Modelling—Some thoughts on complexity. In Proceedings of the 32nd Education and research in Computer Aided Architectural Design in Europe (eCAADe) Conference, Newcastle, UK, 10–12 September 2014; Volume 2, pp. 547–556.
35. Leitão, A.; Lopes, J.; Santos, L. Programming Languages for Generative Design: A Comparative Study. *Int. J. Archit. Comput.* **2012**, *10*, 139–162. [CrossRef]
36. Sammer, M.J.; Leitão, A.; Caetano, I. From Visual Input to Visual Output in Textual Programming. In Proceedings of the 24th International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA), Wellington, New Zealand, 15–18 April 2019; Volume 1, pp. 645–654.
37. Zboinska, M.A. Hybrid CAD/E platform supporting exploratory architectural design. *Comput.-Aided Des.* **2015**, *59*, 64–84. [CrossRef]
38. Eastman, C.; Teicholz, P.; Sacks, R.; Liston, K. *BIM Handbook: A Guide to Building Information Modeling for Owners, Designers, Engineers, Contractors, and Facility Managers*; Wiley: Hoboken, NJ, USA, 2008.
39. Indraprastha, A.; Shinozaki, M. The Investigation on Using Unity3D Game Engine in Urban Design Study. *J. ICT Res. Appl.* **2009**, *3*, 1–18. [CrossRef]
40. Nguyen, A.T.; Reiter, S.; Rigo, P. A review on simulation-based optimization methods applied to building performance analysis. *Appl. Energy* **2014**, *113*, 1043–1058. [CrossRef]
41. Hocking, J. *Unity in Action: Multiplatform Game Development in C# with Unity 5*; Manning Publications: New York, NY, USA, 2015.
42. Grubert, J.; Witzani, L.; Ofek, E.; Pahud, M.; Kranz, M.; Kristensson, P.O. Text Entry in Immersive Head-Mounted Display-based Virtual Reality using Standard Keyboards. In Proceedings of the 25th Conference on Virtual Reality and 3D User Interfaces, Tuebingen/Reutlingen, Germany, 18–22 March 2018; pp. 159–166.
43. Walker, J.; Li, B.; Vertanen, K.; Kuhl, S. Efficient Typing on a Visually Occluded Physical Keyboard. In Proceedings of the CHI Conference on Human Factors in Computing Systems, Denver, CO, USA, 6–11 May 2017; Association for Computing Machinery: New York, NY, USA, 2017; pp. 5457–5461.
44. Vive Team. Introducing the Logitech BRIDGE SDKs. 2017. Available online: https://blog.vive.com/us/2017/11/02/introducing-the-logitech-bridge-sdk/ (accessed on 27 December 2021).
45. Coppens, A.; Mens, T.; Gallas, M.A. Integrating Virtual Reality During the Architectural Design Process: A Survey to Identify Practitioner Needs. In Proceedings of the 38th International Conference of International Council for Research and Innovation in Building and Construction (CIB) W78, Luxembourg, 15–11 October 2021; Volume 2021, pp. 539–548.
46. Merrill, N.; Chuang, J. From Scanning Brains to Reading Minds: Talking to Engineers About Brain-Computer Interface. In Proceedings of the CHI Conference on Human Factors in Computing Systems, Montreal QC, Canada, 21–26 April 2018; Association for Computing Machinery: New York, NY, USA, 2018; pp. 323:1–323:11.

47. Wu, F.; Yan, D.M.; Dong, W.; Zhang, X.; Wonka, P. Inverse Procedural Modeling of Facade Layouts. *ACM Trans. Graph.* **2014**, *33*, 1–10. [CrossRef]
48. Leitão, A.; Garcia, S. Reverse Algorithmic Design. In *Design Computing and Cognition (DCC'20)*; Gero, J., Ed.; Springer: Atlanta, GA, USA, 2021; pp. 323–334.