Graphics and Visual Computing 4 (2021) 200020

Contents lists available at ScienceDirect

Graphics and Visual Computing

journal homepage: www.elsevier.com/locate/gvc

Special Section on CAD & Graphics 2021

Virtual reality game level layout design for real environment constraints

Huimin Liu, Zhiquan Wang, Angshuman Mazumdar, Christos Mousas*

Department of Computer Graphics Technology, Purdue University, West Lafayette, IN 47907, USA

ARTICLE INFO

Article history:
Received 27 March 2021
Accepted 13 April 2021
Available online 30 April 2021

Keywords:
Game level
Level layout design
Virtual reality
Real environment constraints

ABSTRACT

This paper presents an optimization-based approach for designing virtual reality game level layouts, based on the layout of a real environment. Our method starts by asking the user to define the shape of the real environment and the obstacles (e.g., furniture) located in it. Then, by representing a game level as an assembly of chunks and defining the game level layout design decisions in cost terms (mapping, fitting, variations, and accessibility) in a total cost function, our system automatically synthesizes a game level layout that fulfills the real environment layout and its constraints as well as the user-defined design decisions. To evaluate the proposed method, a user study was conducted. The results indicated that the proposed method: (1) enhanced the levels of presence; (2) enhanced the levels of involvement of participants in the virtual environment; and (3) reduced the fear of collision with the real environment and its constraints. Limitations and future research directions are also discussed.

1. Introduction

Virtual reality games are designed so that the player uses controllers to navigate in the virtual environment. However, navigation through locomotion is considered one of the universal tasks performed in real and virtual environments [1]. Moreover, sensorimotor actions are essential factors in providing a compelling experience for virtual reality users [2], just as the mismatching between the real and virtual environment is equally essential in impacting the movement behavior and arousal of virtual reality users [3]. If the user cannot naturally move around and engage with the virtual environment as they would in a real one, then the illusion of being in another place would diminish, making the whole play experience poor and less realistic.

Given the fact that walking is a simple and intuitive method of interaction in the environment, providing a game player with the ability to experience the gaming environment by walking in it would likely enhance the player's gaming experience (i.e., the player will be able to freely move and interact in the virtual game level environment). However, it is impossible for a game level designer to know in advance the layout and size of the real environment and the obstacles (e.g., furniture) involved. Therefore, it is difficult to create customized game levels for numerous real environment configurations. To overcome this challenge, this paper presents a novel computational approach that considers both the shape of the real environment and its obstacles, and automatically generates game levels that take into account the real environment layout and its constraints (see Fig. 1).

Our method first considers that a game level can be represented as an assembly of multiple game level chunks. Second, it asks the user to define the play area and the constraints/obstacles located in it by simply using a virtual reality controller (e.g., the Oculus Touch controller); thus, the real environment's layout is generated. Third, our method assigns four cost terms to a total cost function that encodes design decisions (mapping, fitting, variations, and accessibility) and provides a designer the ability to prioritize the cost terms in order to generate level layouts with different objectives. Finally, the game levels are synthesized using an optimization-based method, the Markov Chain Monte Carlo. We implemented our optimization framework as a plug-in for the Unity game engine and demonstrated how it could generate different types of games. We intend to release the plug-in for public use.

To understand the effectiveness of our method, a user study was conducted. The results indicated that the proposed method (1) enhanced the levels of presence, (2) enhanced the levels of involvement of participants in the virtual environment, and (3) reduced the fear of collision with the real environment. The significant contributions of our work include:

* Corresponding author.
E-mail addresses: liu2833@purdue.edu (H. Liu), wang4490@purdue.edu (Z. Wang), amazumda@purdue.edu (A. Mazumdar), cmousas@purdue.edu (C. Mousas).

https://doi.org/10.1016/j.gvc.2021.200020

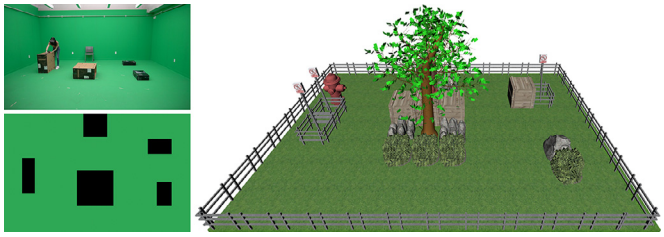


Fig. 1. Given a real environment and its constraints, the user can easily capture its layout using a virtual reality controller. Later, our proposed method synthesizes a game level layout that matches the real environment layout and its constraints.

- An optimization-based approach to design game level layouts reflecting the real environment and its constraints. Such a concept could be used in a variety of virtual reality games and therefore improving user experience while also allowing users to immerse themselves in the virtual world and the gaming scenario.
- The ability of our approach to generate game level designs for different layouts and constraints.
- The ability to customize the synthesized game level by prioritizing each cost term.

We think that both the game development and the virtual reality community will benefit from such a method. Our method provides any user the ability to walk and interact in the gaming environment more efficiently, even though in a constrained real environment (e.g., living room). Each space is unique in shape, size, and furniture configuration (there might be empty spaces or spaces with a lot of furniture). Therefore, a method that automatically synthesizes a game level for any real environment is essential to provide all game players with a compelling virtual reality gaming experience.

This paper is organized in the following sections. Section 2 covers related work. Section 3 presents preliminary information on the proposed method. Section 4 describes the way game level design problems are formulated and solved. Section 5 outlines the details for implementation. Section 6 presents the conducted user study and its results. Finally, Section 7 summarizes our conclusions, our method's limitations, and potential future work.

2. Related Work

The difficulty of moving through a highly constrained real environment while observing the virtual environment lies in the impact such an environment has on the sense of presence and immersion of virtual reality users [4,5]; the result could be an experience that is entirely less engaging to the user. Therefore, natural walking in virtual environments remains a challenge primarily because of the large space required to allow the user to experience the virtual world [6]. However, among others, redirected walking [7–9] and virtual-to-real environment mapping [10] partially solve the limited space problem of real environments by manipulating the user's real-world trajectories.

Several attempts have been made to overcome the limitation of experiencing a virtual environment while walking in a real one. Nescher et al. [11] proposed a method that analyzes the real environment in advance in order to identify walkable areas in the virtual environment. Later, this information is processed and used in order to provide ad hoc free walking in virtual environments when the user finds himself in a constrained virtual environment. Shapira et al. [12] developed a method that analyzes the user's real environment in order to identify flat surfaces, like a wall or a couch, which will determine candidate locations for placing virtual objects in the real environment. Nuernberger et al. [13] developed an augmented reality application in which the edges and planes of the real environment are detected so they can be aligned with virtual content placement. McGill et al. [14] proposed the Augmented Virtuality [14] which adds out-of-context information to the virtual environment only when considered necessary by the system. Other methods include the use of occupancy maps and glass walls [15,16], virtual environment layers [17,18], such as wireframes or other visual indicators that indicate the presence of real obstacles, and vibrotactile actuators in the head-mounted display that trigger an alert signal when users are approaching obstacles [18]. However, in most of the previously mentioned methods the sense of presence was not improved, indicating a significant limitation when experiencing a virtual environment.

Prior research has also focused on using 3D scanning technology to acquire a replica of the real environment. For example, Kanamori et al. [19] used a 3D scanner to scan a real environment and superimposed the 3D point cloud of the user's real environment onto the virtual environment through the head-mounted display. Sra et al. [20] used 3D scanning technology to acquire a 3D model of the real environment. Then, by detecting the walkable area, they could generate fences or water that would prevent the user from walking in specific locations. In addition, by using a small dataset of objects, Sra et al. [20] could substitute real environment objects with virtual ones. Although promising, this approach requires using a 3D scanner, equipment that only a limited number of virtual reality users can access in their homes. Moreover, the applicability of such a method in creating virtual environments for game purposes is unclear.

In this paper, it is proposed the use of an optimization-based method, which has been extensively used in designing virtual environments quickly and affordably [21–25]. The optimization-based synthesis of game-related content is a critical technique used extensively in the modern game-development process [26]. Furthermore, optimization-based design techniques are beginning to enhance game replayability because they offer users the ability to play multiple variations of a single game. Examples of optimization-based methods include the work of Hartsook et al. [27] and Hullett and Mateas [28]. They employed optimization-based design methods for matters of adaptability or replayability. Such methods also provide the ability to design games that adapt to a variety of constraints and parameters; both during the initialization process and before the game starts [29–31]. Optimization-based methods can even alter a game dynamically in response to events in the game [32–35].

Our method synthesizes a game level layout taking into account the real environments and its constraints, while also allowing a game level designer to control the synthesized gaming environment by prioritizing the cost terms of the presented total cost function. Thus, our method facilitates designer control over generated content and gives players the ability to dictate the degree to which the synthesized game will focus. Our method requires a simple virtual reality controller that comes with a head-mounted display to capture the layout of the real environment. This feature is in contrast to the 3D scanner common in previous methods [19,20]. Although no precise information about the real environment could be captured using a virtual reality controller, our low-cost layout-capturing method provides enough data to sufficiently synthesize the game levels. We think our method could be useful for the automatic design of unique virtual reality game level layouts without the need for additional hardware.

3. Preliminaries

This section presents the preliminary steps required to develop our pipeline. The steps include: (1) the capture process for the real environment and its associated constraints; (2) the game level

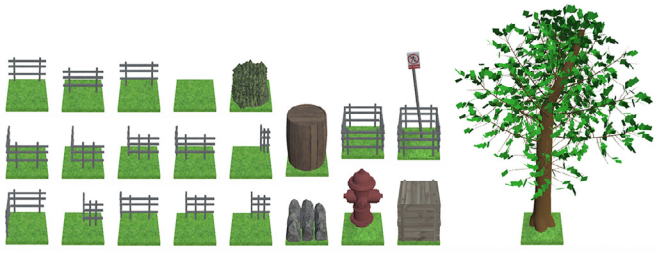


Fig. 2. Examples of game level chunks developed for the proposed project.

chunks that need to be designed for synthesizing the virtual reality game levels; and (3) generating and characterizing the virtual grid.

3.1. Capturing the real environment and Constraints

Our method begins by asking the user to capture the real environment and its constraints in which the virtual reality game will be created. This process generates a virtual layout according to the real environment in which the user is willing to play the game. The player is then asked to use the controller, which comes with the head-mounted display, to define the real environment's entire play area. To do so, the player simply clicks a button when the controller is located at the diagonal corners of the environment. In our case, we are using the Oculus Quest head-mounted display and the Oculus Touch controller (Oculus Quest does not allow passthrough access; therefore, a user should take slightly off the head-mounted display to observe the real environment during the capturing process). In larger and more complex shaped environments, in which the shape is more than a single rectangle, the user can capture multiple rectangular shapes. Later, these shapes are combined by our system to provide the actual shape of the environment. Note that only the area of the plane (x, z coordinates) is captured.

Next, the user is asked to define the constraints found in the real environment. Constraints are defined as any object that might prevent the user from moving within the physical environment (e.g., coffee tables, couches, television stand, chairs, etc.). For this process we use the virtual reality controller, with which the user must define a shape that encloses the objects. After the user finishes the capturing process for the real environment and its constraints, the system generates the environment's layout. Fig. 1 shows examples of the real environment and the associated layouts generated based on the process described above.

This paper does not present a commercial product but a proof of concept; that is, how to automatically design virtual reality game levels based on real environment constraints by considering a number of game level layout design decisions. To simplify the process of capturing the environment and its constraints, we use rectangular shapes. If a number of resources for game level chunks are available, a developer could easily extend our approach to capture the real environment more precisely by incorporating additional shapes or using a paint-based method to define environment's boundaries. However, most virtual reality systems use base stations which enclose the user in a square shaped area. Thus, our proposed method considered only rectangular-shaped game level chunks.

3.2. Game level chunks

To synthesize game levels based on a real environment and its constraints, we used game level chunks (see Fig. 2). Because this project is a proof of concept, we decided to use a relatively small number of chunks compared to multiple 3D game objects found in a commercial game. However, given the availability of various resources, a developer could easily extend our approach and incorporate more game level chunks. Our particular game level chunks are divided into three types:

- **Open Space Chunks:** Open space chunks are placed in free-from-obstacles grid cells and are used to define the virtual environment's walkable area.
- **Boundary Chunks:** Boundary chunks are placed in the virtual environment's boundaries in order to inform users of the boundaries and prevent them from colliding with the walls in the real environment. We developed two types of boundary chunks: (1) the corner boundary chunk and (2) the straight boundary chunk.
- **Obstacle Chunks:** Obstacle chunks are intended to substitute real environment obstacles in the virtual reality game. These chunks inform the user that particular areas in the virtual environment are occupied, thereby preventing the user from colliding with real environment obstacles.

Each chunk is represented with the label that characterizes it and a directional vector (up, down, left, right, right and up, right and down, left and up, and left and down). The directional vector is later used to correctly align the game level chunks with the generated grid map. Additional chunks might be needed for more complex games and real environments. However, based on our experiments during development, we found that the three types of game level chunks are sufficient enough to cover almost any area and allow developers to synthesize a virtual reality game level that can fit a real environment and its constraints.

3.3. Grid generation and characterization

Given the area captured by the user as input, our system generates a $M \times N$ virtual grid, which encloses the entire captured area. Later, a part of this grid is used to synthesize the game level. For this project, a cell in the grid has a dimension of 50×50 cm, which is equal to the size of each game level chunk; however, other dimensions could also be considered. Our system uses an inside-outside test [36] to identify which grid cells correspond to the captured area. Next, the grid cells that correspond to the captured area become the "boundary" or "inner." The remaining grid cells which are not in a captured area are then excluded.

Our system assigns a directional vector ($\mathcal{V}$) to the boundary cells: up, down, left, right, right and up, right and down, left and up, and left and down. The first four vectors are assigned to the straight boundary chunks, and the last four diagonal directional vectors are used for the corner chunks. For the inner cells, the system labels each cell as "walkable" for any cell that could be walked by a user, or "obstacle" for any cell occupied by a real environment obstacle. In characterizing each cell as "walkable" or "obstacle," we used the inside-outside test between each cell of the grid and each obstacle shape captured by a user. Both "walkable" and "obstacle" cells are assigned with an up vector. Fig. 3 shows an example of a captured layout and its representation in the virtual grid.

4. Problem formulation and optimization

The goal of the proposed approach is to synthesize virtual reality game levels optimized for the real environment and its constraints and other design criteria, which are encoded by cost terms. Let $L = [c_{1,1}, \ldots, c_{M,N}]$ denote the current configuration of the synthesized game level layout composed of several chunks $c_{i,j}$. Although the game levels chunks are represented in a 2D grid for simplicity, any chunk that belongs to the game level L will be represented as c_i. To synthesize a game level, we developed a total

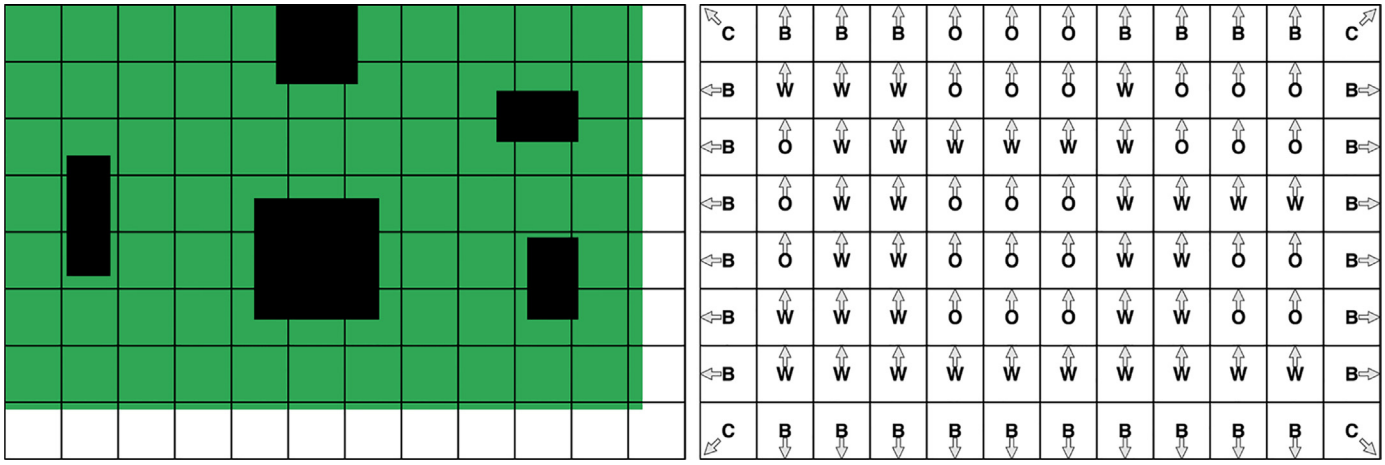


Fig. 3. A layout of the real environment (left) and its representation based in the virtual grid (right). **W** stands for walkable, **O** stands for obstacle, and **B** stands for boundary grid cells.

cost function $C_{Total}(L)$ that evaluates the quality of a level L based on a number of costs (game level layout design decision):

$$C_{Total}(L) = w_M C_M(L) + w_F C_F(L) + w_V C_V(L) + w_A C_A(L) \tag{1}$$

C_M is the mapping term that attempts to map the game level chunks according to the input information represented as virtual grid array after the grid characterization process. C_F ensure a one-to-one fitting between the captured area and the synthesized game level. The C_V denotes the variation that could be introduced to the composed game level layout. The C_A represents the accessibility term that evaluates whether the user will be able to access any open space chunk in the synthesized level layout. Finally, the w_M, w_F, w_V, and w_A are weights that correspond to the cost terms and prioritize each cost term differently during the optimization process depending on their weighted value, given that $\{w_M, w_F, w_V, w_A\} \in [0, 1]$.

Various cost terms could be implemented to handle the layout synthesis of a game level. However, in this implementation phase, we limited the level layout cost terms to those most important for this project. The cost terms and the optimization process are presented in the subsections below.

4.1. Mapping cost

We implemented a level layout mapping cost that tries to map the chunks composing the game level based on the labeled grid of the real environment layout. To do so, we defined the following cost term:

$$C_M(L) = \frac{1}{|L|} \sum_{(c_i, g_i)} \Gamma_{(c_i, g_i)} \tag{2}$$

where c_i denotes a chunk of the game level, and g_i denotes a grid layout cell. $\Gamma_{(c_i, g_i)}$ is then computed based on the following condition:

$$\Gamma_{(c_i, g_i)} = \begin{cases} 0 & \text{if } \mathcal{L}(c_i) = \mathcal{L}(g_i) \&\& \mathcal{V}(c_i) = \mathcal{V}(g_i) \\ 1 & \text{otherwise} \end{cases}$$

where $\mathcal{L}(\cdot)$ returns the label information of the chunk c_i and grid cell g_i, respectively. $\mathcal{V}(\cdot)$ returns the vector information of the chunks c_i and grid cell g_i, also respectively. This ensures that the $C_M(L)$ cost term returns a high value when there is an inconsistency between the layout of the synthesized game level and the target grid map information. Conversely, the cost term returns a low value when the synthesized game level is mapped correctly into the grid map information.

4.2. Fitting cost

To synthesize a game level layout that matches the layout of the real environment as close as possible, we introduced the use of the fitting cost function that attempts to minimize the difference between the area that the game level layout occupies and the area that is captured by a user using the virtual reality controller; this is our input area. This step is achieved by tweaking parts of the game level chunks (e.g., moving the fence of the boundary game level chunk closer to the real environment's captured boundary or selecting another boundary fence game object from our dataset that better fits the target grid). The fitting cost term is then represented as:

$$C_F(L) = \left(\frac{1}{\mathcal{N}_L} \mathcal{A}_B(L) - \underbrace{\frac{1}{\mathcal{N}_R} \mathcal{A}(R)}_{\text{target}} \right)^2 \tag{3}$$

where $\mathcal{A}_B(L)$ denotes the areas of the level layout L inside the boundaries and $\mathcal{A}(R)$ denotes the captured area of the real environment. Finally, $\mathcal{N}_\mathcal{L}$ and $\mathcal{N}_\mathcal{R}$ are normalization constants.

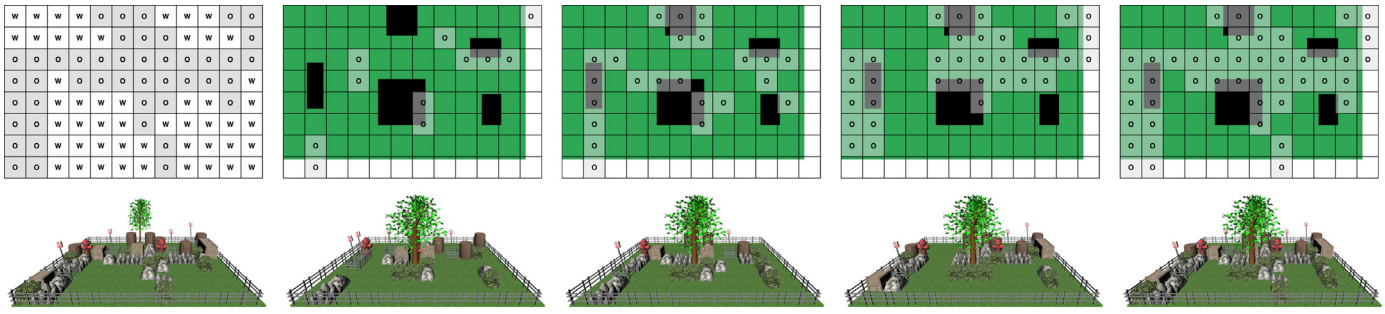
4.3. Variations cost

We realize that, as long as the information provided by the same real environment, the grid array remains the same (i.e., the topology and size of the real environment and the obstacles located in it do not change at all). Thus, the synthesized game levels would have minimal to no difference with one another; and therefore, the synthesized game level could be considered monotonic, and the game players might become bored quickly. In order to synthesize game levels that differ from one another and keep the players engaged as the game levels progress, a variation cost term was introduced to our total cost function. This cost term ensures that each generated game level will not look the same.

For the level variation cost term, we use as input the grid array generated according to the real environment. We apply a Perlin noise [37] function to synthesize an intermediate game level layout I. Then, the variations cost is defined as:

$$C_V(L) = \left(\frac{1}{|L|} \sum_{c_i} |L(c_i) - I(c_i)| - \underbrace{\sigma_V}_{\text{target}} \right)^2 \tag{4}$$

where $\sigma_V \in [0, 1]$ is the target difference between the level layout L and the intermediate level layout I composed of c_i game level chunks. For example, $\sigma_V = 0.50$ means that the intermediate level



(a) (b) $\sigma_V = .25$ (c) $\sigma_V = .50$ (d) $\sigma_V = .75$ (e) $\sigma_V = 1.00$

Fig. 4. The layout of an intermediate game level synthesized using the Perlin noise (a), and the layout of the final synthesized game level layout in which the Perlin noise is included with different σ_V targets (4(b)–(e)). For all examples, the variation cost is given high priority, $w_V = 1.00$, and the accessibility cost is given low priority, $w_A = .05$.

layout I and the synthesized level layout L are 50% similar. Fig. 4 illustrates different intermediate level layouts placed on top of the level layout based on different variation targets (σ_V). It should be noted that with the proposed method the variation can be be controller by the user; therefore, the user can choose the amount of variation that will appear in the synthesized game level.

4.4. Accessibility cost

Because of the variations cost, we understand that a synthesized game level might become over-occupied with obstacle chunks, blocking walkable areas that would otherwise be accessible to a user (see Fig. 4(d) and (e)). To overcome this limitation, we included in our total cost function the accessibility cost that penalizes a synthesized game level when there are unoccupied grid cells that are not accessible. Our accessibility cost term is represented as:

$$C_A(L) = \frac{1}{|L|} \sum_{(c_i,c_j)} \Pi_{(c_i,c_j)} \tag{5}$$

which detects whether all open space area chunks c_i are accessible from any other open space chunk c_j. $\Pi_{(c_i,c_j)}$ is computed based on the following condition:

$$\Pi_{(c_i,c_j)} = \begin{cases} 0 & \text{if } \pi : c_i \rightarrow c_j \\ 1 & \text{otherwise} \end{cases}$$

In the condition above, π denotes the path between c_i and c_j. To compute whether a path exists between c_i and c_j, we used a simple pathfinding algorithm, the Depth-First Search [38,39]. Thus, if the pathfinding algorithm cannot connect two walkable area chunks, it returns 1 and forces the optimizer to continue searching for a game level layout by generating new intermediate-level layouts $I(c_i)$. Otherwise, if a path between c_i and c_j exists, the cost for that cell becomes 0; therefore, the optimizer can achieve its goals. It should be noted that there are cases in which blocked areas might appear because of the capture process of the real environment and its constraints. Because this blocked area results from the real environment's initial capture process and not due to the variation cost, our system does not consider that area as blocked because of the intermediate game level layout $I(c_i)$. Therefore, it does not penalize the $C_A(L)$ cost term.

4.5. Optimization

The game level layout L is optimized for the defined total cost function C_{Total}. A Markov Chain Monte Carlo optimization technique, known as simulated annealing [40] with a Metropolis-Hastings state searching step [41], was applied to solve this optimization problem. For this, we first define a Boltzmann-like [42] objective function:

$$f(L) = \exp\left(-\frac{1}{t} C_{Total}(L) \right) \tag{6}$$

where t denotes the temperature parameter of the simulated annealing. In each iteration of the optimization process, a move is applied to the current game level layout configuration L to propose a new configuration of the level layout L'. In the current implementation, the following moves were implemented:

- **Replace a chunk:** from the current game level layout configuration, a chunk c_i is randomly selected and replaced with another randomly chosen chunk from the chunks dataset.
- **Swap chunks:** from the current game level layout configuration, two chunks are randomly selected; the two chunks then swap positions.
- **Rotate a chunk:** our system randomly selects a c_i chunk and rotates it either to -90 or $+90$ degrees. This move helps the optimization to align the boundary chunks in the layout.
- **Edit a chunk:** our system randomly selects a boundary or obstacle chunk c_i and edits it by moving its child object (e.g., a fence) in either a positive or negative direction of the assigned vector.

The probability of selecting the move to be applied during the optimization process at each iteration was pre-defined by the authors. Unless otherwise specified, for the "replace a chunk" move, we set the selection probability to $p_{replace} = 0.30$. For the "swap chunks" move, we set the selection probability to $p_{swap} = 0.20$. For the "rotate a chunk" move, we set the selection probability to $p_{rotate} = 0.20$, and for the "edit a chunk" move, we set the selection probability to $p_{edit} = 0.30$.

The output of each move is the proposition of a new configuration of the game level layout L'. To decide whether the developed method should accept the proposed level design L', the proposed total cost value $C_{Total}(L')$ is computed and compared to the cost of the current game level layout configuration $C_{Total}(L)$. To maintain the detailed balance in the optimization, our approach accepts the proposed level L' with the acceptance probability $P(L'|L)$ based on the Metropolis criterion for each move as follows:

$$P(L'|L) = \min\left(1, \frac{f(L')}{f(L)}\right) \tag{7}$$

To efficiently explore the solution space, simulated annealing [40] was applied. Simulated annealing is controlled by a temperature parameter t that, at the beginning of the optimization, is assigned a high temperature t value to allow the optimizer to explore the synthesis of the game level solution space aggressively. During the optimization process, the temperature values t is lowered gradually. In the current scheme, we set $t = 1.00$ at the beginning

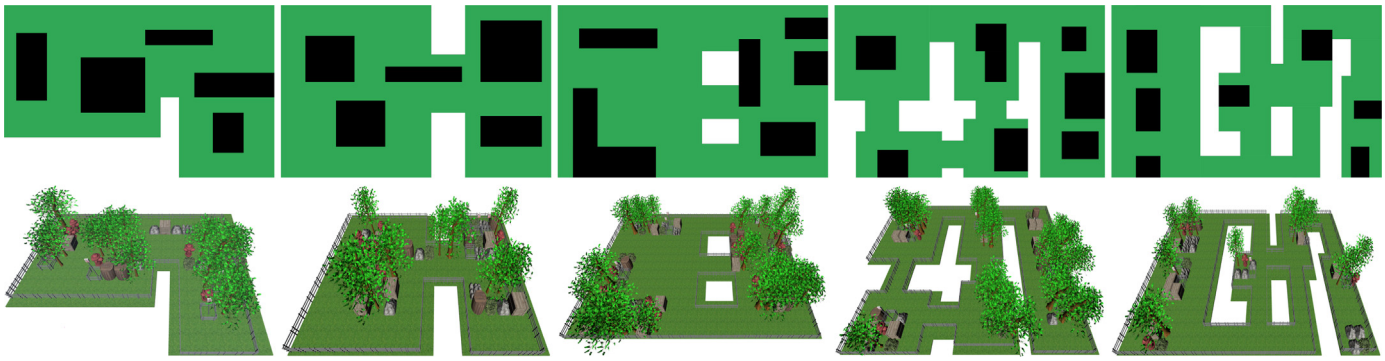


Fig. 5. Example of synthesized level layouts based on different input layouts and weights assigned to each cost term of the total cost function. For all examples, the weight of the variation cost is set to $w_V = .00$.

of the optimization and decided to decrease it by 0.10 at every 1000 iterations until it reaches zero. Essentially, the optimizer becomes increasingly greedy in seeking to refine the solution while it is set to terminate if the total cost change is less than 5% over the previous 100 iterations. Based on our experimentation, we set the weights of each component of the total cost function in our optimization as $w_M = 1.00$, $w_F = 1.00$, $w_V = 0.50$, and $w_A = 1.00$, unless otherwise specified.

By changing the weight of each cost term, the game level designer can always control the synthesis of the layout to emphasize specific design goals. We think that providing the game level designer the freedom to interactively explore possible game level layout designs can be a helpful feature. For example, the designer might want to prioritize the synthesized game level in a particular way by simply adjusting the weights on the total cost function (e.g., if a designer wants to assign a lower priority to the accessibility cost term, the weight of the accessibility cost term should be set be $w_A < 1.00$). The optimizer is responsible for generating the level layout and proposing a game level design for the designer-specified priorities. Fig. 5 shows examples of game levels (the two games presented in Section 5) synthesized for different real environment layouts and different weights assigned to each component of the total cost function.

5. Implementation details and example games

This section provides details about the implementation of our proposal and the two games that were developed.

5.1. Implementation

Our virtual reality game level design framework was implemented on a Dell Alienware with Intel a Core i7 CPU and 32 GB of memory. Our framework was developed in the Unity game engine using C#. Our scripts are easily adaptable to different game level chunks. The user simply needs to attach the necessary chunks to the Inspector window editor that has been implemented. After providing input about the layout of the real environment, our system automatically synthesizes the game level layout. The games presented in the subsections below were also implemented in the Unity game engine using the Oculus Integration. Our application was implemented and exported to Oculus Quest. Depending on the size of a real environment, synthesizing a game level consisting of 100 chunks (e.g., 10×10 grid) requires less than 5000 iterations. Based on our current implementation running on the Oculus Quest head-mounted display, this process can be finished in less than one minute. Finally, both games run on 65 fps in Oculus Quest. Moreover, we tested the number of iterations needed to synthesize game levels with different grid resolutions (5×5, 10×10, 15×15, and 20×20 grids). In this evaluation for each grid resolution we developed input layouts that are occupied with 0%, 25%, 50%, 75% and 100% obstacles and we ran the optimization process five times. The results are shown in Fig. 6. As we can observe, the iterations needed are not related to the obstacle percentage but to the size of the grid.

A comparison of the virtual reality game level layout optimization between the MCMC and Greedy algorithms is shown in Fig. 7. It should be noted that, compared to MCMC, the greedy algorithm only accepts a proposed total cost $C_{Total}(L')$ that provides a better configuration than the current total cost $C_{Total}(L)$. The MCMC algorithm obtains a solution with a lower total cost value (0.08) compared to the Greedy approach (0.47). The total cost value of the greedy algorithm experiment didn't change from about iteration 550 to about iteration 650. Thus, the greedy optimization stopped at about iteration 650. Since the MCMC algorithm can accept a solution with a cost higher than that of the current solution with a certain acceptance probability, the sampling is capable of jumping out from a locally optimal solution. This prevents the sampling from being performed locally, and eventually locating a more optimal solution with a lower total cost value. Thus, the MCMC optimization stopped at about iteration 910.

5.2. Example games

We developed two virtual reality games to demonstrate how our game level layout method can be used in gaming scenarios. The first game is called *Backyard Fortune*, a puzzle game. The second game is called *The Rebooter*, a shooting game. The *Backyard Fortune* game was the basis of the user study presented in Section 6. Below we provide more details about the games. Screenshots of the two games from the player's point of view are shown in Fig. 8. Moreover, Fig. 9 shows a user playing the *Backyard Fortune* game in our lab space.

*Backyard fortune*The primary concept behind *Backyard Fortune* is for the player to collect puzzle pieces, find the key to a treasure box, and unlock it. The player is free to move in the walkable area of the game level and collect puzzle pieces using the Oculus Touch controller. Two panels on the virtual controllers are positioned where the knuckles would usually be placed. The first panel is designed like a clipboard, which provides instructions to the player on how to play the game. The second panel is an inventory that keeps track of which pieces have been collected in relation to the total number of puzzle pieces in the level. These panels can be toggled. The user can hide them and bring them back if he/she wants to keep track of how many more pieces need to be collected. The puzzle pieces are randomly placed in the walkable area once the level layout has been generated. The primary objective is to navigate the environment and use the controllers to

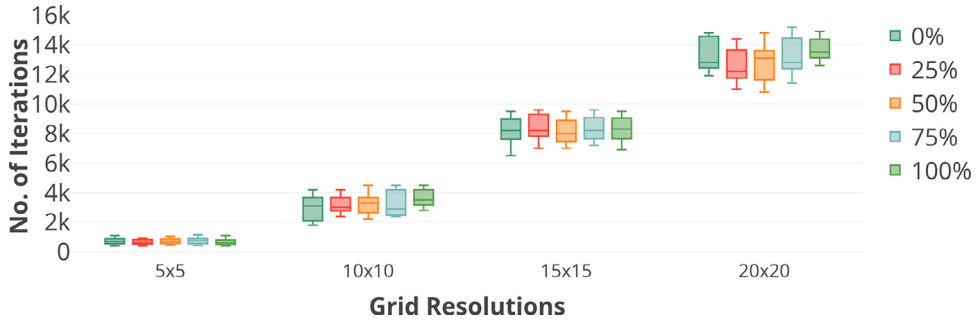


Fig. 6. Number of iterations needed by our system to synthesize game level layouts with different grid resolutions (5×5, 10×10, 15×15, and 20×20 grids) and different percentages (0%, 25%, 50%, 75% and 100%) of obstacles that occupy the grids.

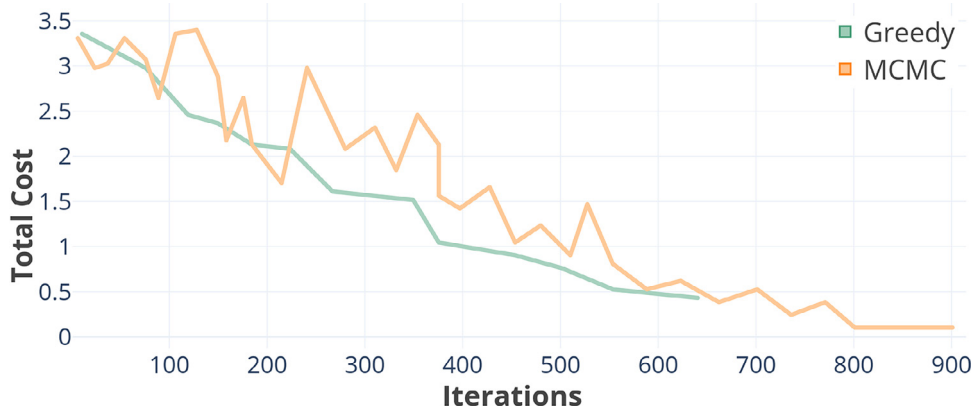


Fig. 7. A comparison between optimizing the game level layout using the MCMC and the greedy algorithm in a 5×5 grid. The MCMC algorithm achieves lower minima compared to the greedy algorithm.

Fig. 8. Screenshots of the *Backyard Fortune* (left) and *The Rebooter* (right) games.

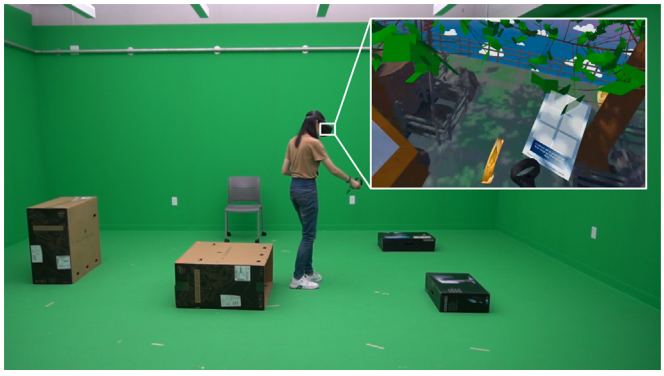


Fig. 9. A user playing our *Backyard Fortune* game in our lab space.

collect the puzzle pieces and put them in their inventory. Once all the pieces have been collected, a key that unlocks the treasure box appears. The player can then pick up the key, insert it in the front of the treasure box, and unlock the box to enjoy his/her fortune.

The rebooter The Rebooter game presents more challenging conditions to the player. Specifically, the player tries to shoot the enemies that chase him/her. The enemies have been designed so that they have a set patrol path around the obstacle piece. The set patrol path is included to satisfy one of the core gameplay mechanics of video games—anticipating and reacting to non-playable characters' patterns. In this game, the player holds a virtual gun in his/her right hand that shoots lasers. The laser gun is used to fulfill the instruction to destroy any enemy. The game ends once the player destroys all enemies.

6. User study

A user study was conducted to evaluate the proposed optimization-based game level design method. The following sections explain the details of our study.

6.1. Participants

The recruitment of participants was based on emails that were sent to all students in our department. In total, 25 students volunteered for the study (18 males and 7 females). Participants ranged in age from 19 to 27, with the mean age $M = 21.56$ ($SD = 2.78$). All participants had prior experience with virtual reality games. There was no compensation for participating in the study. The study followed a within-group design; therefore, all participants experienced all three conditions presented in the section below.

6.2. Experimental conditions

Three experimental conditions were developed to evaluate participants' experiences: (1) when interacting with our method in which the virtual environment is optimized based on the real environment and its constraints; (2) when interacting with a real environment that mismatches the constraints of the virtual environ-

ment; and (3) when interacting with a free-from-obstacles real environment. For this study, the Oculus Quest head-mounted display was used. Note that the synthesized game level was the same for all three conditions for all participants, which means that the game level layout that was optimized in the first condition was also used for the rest of the conditions. The only difference was in the layout of the real environment and the manner in which the users were made aware of the real environment obstacles. Details of the three conditions are provided below.

- **Optimization:** For this condition, the layout of the real environment and the obstacles located in it were captured using our method. Then, the game level's layout was automatically generated based on the proposed method. In doing so, the boundary game level chunks were placed on the boundaries (walls) of the real environment. Virtual obstacles were placed as substitutes in the exact position of real environment obstacles.
- **No Optimization:** This condition was used to determine whether mismatching the real and virtual environments in terms of obstacles placed in the real environment would affect participants' responses. For this condition, we are using the real environment and its constraints that were initially captured using our method. Then, the real environment obstacles (carton boxes) were moved to a different position. Finally, the real environment is captured using the calibration tool in Oculus Quest; therefore, a mismatching between the real and the virtual environment was achieved. During the gameplay, the user is informed if he/she is close to a real environment obstacle by the guardian functionality of Oculus Quest.
- **No Obstacles:** This is considered our baseline condition and was used to investigate how participants interact in the real environment while knowing in advance that there are no obstacles (the carton boxes were removed from the play area); this kind of real environment could be considered as safe. As in the other two conditions, we are using the game level layout that was generated by the optimization condition. Boundary game level chunks were placed in the virtual environment to inform the participant about the actual boundaries of the real environment and obstacle chunks were placed at the initial positions of real environment obstacle.

6.3. Measurements

In this study, a computer-based questionnaire was provided to all participants. The purpose was to explore their presence and fear of collision (emotional state) with the virtual environment. Specifically, the sense of presence was measured using the Igroup Presence Questionnaire (IPQ) [43,44], which consisted of 14 items and was divided into four parts: (1) one item reflected the initial definition of **presence**, according to Slater and Usoh [45]; (2) five items reflected **spatial presence**, denoting the sense of being "physically there" in the virtual environment; (3) four items reflected **involvement** focused on attention during the interaction, as well as the perceived involvement of the participants; and (4) four items reflected the **experienced realism**, which evaluates the realism of the virtual environment. The four-item scale on emotion, drawn from Tcha–Tokey et al. [46], was also used to investigate participants' **fear** (emotional state) while in the virtual environment. Finally, a section of the questionnaire asks participants for additional input about their experience when interacting with the three experimental conditions outlined in Section 6.2.

6.4. Procedure

When participants entered the lab, the research team asked them to sign a consent form approved by the Institutional Review Board of our university if they agree to participate in the study. The, the participants were then asked to complete a demographics questionnaire.

As mentioned, Oculus Quest does not allow passthrough access; therefore, a member of the research team that was experienced with the capturing process used the Oculus Quest to capturing the real environment and its constraints, and during this process the researcher took slightly off the head-mounted display to observe the real environment. The participants were not given any information about the conditions they would experience. They would first see the virtual environment once they put on the head-mounted display, and then game would start. The research team helped the participants with the virtual reality equipment (Oculus Quest) before the game started. Once the virtual reality gaming application started, the participants were asked to play the game. When the game was over, a visual indication on the screen would notify the participant. The research team then helped the participants by setting up the next experimental condition.

To control potential carry-over effects, the sequence in which each participant would experience the three experimental conditions was randomized using Graeco-Latin squares. Between the conditions, the participants were asked to complete a questionnaire distributed in a paper-based format. This time period was also used to provide participants a short break. The participants were informed that the virtual environment's boundaries corresponded to the boundaries of the real environment. However, participants were not told whether there was a match or mismatch between the real and the virtual environments. They were able to observe it once they put on the head-mounted display. It should be noted that none of the participants made contact with any of the walls during the study. Each participant spent no more than 45 min in completing the study. All participants were aware that they were free to quit the study at any time.

6.5. Results

In analyzing our data, we used a one-way repeated measures analysis of variance (ANOVA) to determine the differences across the three experimental conditions. The internal validity of the scales of the questionnaire was measured using Cronbach's alpha coefficient. With sufficient scores ($0.73 < \alpha < 0.94$), we used a cumulative score for each item that belonged to each questionnaire component. The normality assumption of the objective measurements and subjective ratings were evaluated with Shapiro–Wilk tests at the 5% level and with the residuals' graphic Q-Q plots. Post hoc comparisons were conducted using Bonferroni corrected estimates. A $p < 0.05$ value was deemed statistically significant. Boxplots from the obtained results are shown in Fig. 10.

Statistically significant results were found for **presence** across the examined experimental conditions [$\Lambda = 0.603$, $F(2, 23) = 7.560$, $p < 0.005$, $\eta_p^2 = 0.397$]. The post hoc comparison showed that the mean score of the no optimization condition ($M = 3.24$, $SD = 1.56$) was lower than that of the no obstacle condition ($M = 4.88$, $SD = 1.42$) at the $p < 0.005$ level and the optimization condition ($M = 4.64$, $SD = 1.60$) at the $p < 0.05$ level

The **spatial presence** of participants was statistically significant across the examined experimental conditions [$\Lambda = 0.504$, $F(2, 23) = 11.220$, $p < 0.001$, $\eta_p^2 = 0.494$]. The post hoc comparison showed that the mean score of the no optimization condition ($M = 3.52$, $SD = 1.61$) was lower than that of the no obstacle condition ($M = 5.56$, $SD = 1.38$) at the $p < 0.005$ level and the optimization condition ($M = 4.92$, $SD = 1.52$) at the $p < 0.001$ level.

We also identified a statistically significant effect on the participants' **involvement** across the examined experimental conditions [$\Lambda = 0.613$, $F(2, 23) = 7.273$, $p < 0.005$, $\eta_p^2 = 0.387$]. The post hoc comparison showed that the mean score of the no optimization

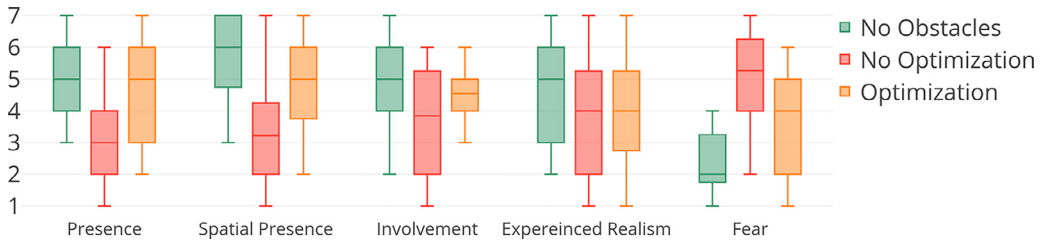


Fig. 10. Boxplots from self-reported ratings for each examined concept across the three experimental conditions.

condition ($M = 3.36$, $SD = 1.80$) was lower than that for the no obstacle condition ($M = 4.96$, $SD = 1.59$) at the $p < 0.01$ level and the optimization condition ($M = 4.56$, $SD = 1.44$) at the $p < 0.05$ level.

Notably, no statistically significant results were found for the **experienced realism** measurement across the examined experimental conditions [$\Lambda = 0.930$, $F(2, 23) = .864$, $p = 0.435$, $\eta_p^2 = 0.070$]. However, a statistically significant effect on the participants' **fear** (emotion) was found across the examined experimental conditions [$\Lambda = 0.328$, $F(2, 23) = 23.593$, $p < 0.001$, $\eta_p^2 = 0.672$]. The post hoc comparison showed that the mean score of the no obstacle condition ($M = 2.32$, $SD = 1.10$) was lower than that for the no optimization condition ($M = 4.88$, $SD = 1.53$) at the $p < 0.001$ level and the optimization condition ($M = 3.48$, $SD = 1.53$) at the $p < 0.05$ level. Moreover, we found that the mean fear rating of the optimization condition was lower than the no optimization condition at the $p < 0.05$ level.

6.6. Discussion

The analyses of presence, spatial presence, and involvement revealed that our method could in fact synthesize the virtual reality game level based on the real environment and its constraints and further synthesize game levels that keep the user engaged. Specifically, our optimization-based method (optimization condition) was able to outperform the no optimization condition. Please note that the no optimization condition describes the way that people experience virtual reality games from their living room; in other words, from a real environment full of obstacles that do not match the virtual environment in terms of constrains and appearance. Additionally, it appeared that, during the gaming experience, the presence of our participants were not interrupted (break-in-presence effect [5]) by the guardian functionality of Oculus. As a result, our participants were more able to focus on and enjoy the game during the optimization and no obstacle conditions.

Moreover, the statistical analyses showed that the proposed method could indeed provide results close to the no obstacle condition. This encouraging finding means that the participant level of presence, even in a constrained environment, was close to the level of presence in a free-from-obstacles environment. This finding indicates that when participants are placed in a virtual environment that matches the constraints of the real environment, their presence level is close to a condition during which they know in advance that they will be able to move and interact in a free-from-obstacles environment; even if there is no appearance matching between the two environments. Finally, for experienced realism, participants experienced the same virtual environment across the three conditions. As a result, they provided similar ratings since the experienced realism was related more to the appearance of the virtual environment and less on the structure of the real environment [47,48].

In addition to the positive results regarding presence, we also observed interesting findings relating to the participants' emotional state when examining their fear during their interaction with the three different experimental conditions. Specifically, participants indicated that when playing the game during the no obstacle condition, their level of fear was lower than when playing in the no optimization and the proposed optimization conditions. This result indicates that when participants were placed in a free-from-obstacles real environment, they felt safer walking in it because simply there were no obstacles to anticipate or avoid. However, once obstacles were placed, the participants began to feel less safe since they needed to move more carefully in order to avoid any potential sudden encounters like hitting an obstacle. Moreover, our results showed that participants' fear was rated lower during the optimization condition than in the no optimization condition. The finding related to fear is significant because it demonstrates that, to the extent participants become aware that there is a match between the real and the virtual environment in terms of layout and constraints, this spatial awareness reduced the fear of colliding with the obstacles. We consider this to be the most important finding of the study that highlights the advantage of the proposed approach.

The participants also submitted several comments about the different conditions they experienced. All of the comments we received for the optimization-based approach were positive, suggesting that optimizing virtual reality game levels for real environments and their constraints should be taken into consideration by virtual reality game developers. For the optimization condition, some of the participants reported that once they became familiarized with the virtual environment, they realized there was a match between the position of the real environment obstacles and the virtual objects they were viewing. A few more said that the game objects in the virtual environment made the location of the walkable area clear and therefore easy for them to navigate freely in the virtual environment. Regarding the no optimization condition, some participants reported they disliked the interruption of the Oculus' guardian functionality. Moreover, others noted that such mismatching between the real and the virtual environment made them more apprehensive before performing their next step.

In conclusion, we realized it would have been useful to collect additional data to further understand how participants interacted in the virtual environment during the three conditions. In particular, added feedback on data such as participants' movements, proximity to boundaries, proximity to real and virtual obstacles, and the number of collisions with real obstacles would have provided additional and, potentially, useful information on the way they perceived and interacted between the real and virtual environments. However, considering the findings from the self-reported ratings and the comments participants submitted, we concluded that the proposed optimization method could be a promising solution for synthesizing virtual reality game levels for real environment constraints while also helping keep the user engaged with the game.

7. Conclusions, limitations, and future work

This paper introduced an optimization-based method of designing game levels based on real environment layouts and the constraints that vary in shape and size. We think that our method

could be effectively used in a variety of real environments, including living rooms, bedrooms, or office spaces. The proposed method synthesizes game level layouts in a fast and scalable manner with minimum effort by the user and without the need for additional hardware (e.g., a 3D scanner) for providing input information. Additionally, our method provides game level designers with the necessary control of the synthesized game level in order to prioritize the objectives of the design process by simply changing the weight that controls the cost terms.

To understand the effectiveness of the proposed method, we conducted a user study. Our study provided a number of interesting insights into the participants' experience in the virtual environment. The results of the user study indicated that the proposed method was admittedly able to enhance the participants' presence and involvement while reducing the fear of collision with the real environment and its obstacles.

There are several limitations that should be addressed in the future. Tackling these limitations would allow modifications to the proposed method by making it applicable to a broader range of users and by enhancing its efficiency to accommodate more complex gaming scenarios. The proposed method examines only a small number of chunk types and chunks with simple shapes. However, there may be games that require a variety of complex chunks in terms of shape and size. Further, the proposed method looks at only the (x, z) coordinates of the real environment and its constraints, thereby excluding the height of the environment and the obstacles contained in it. We think that incorporating the third dimension, or even a more precise representation of the real environment using a 3D scanner, would allow us to synthesize game levels that match the real environment more precisely in terms of shape and constraints.

In addition to the above mentioned limitations, several alternative directions could also be explored in the near future. Specifically, the current method is highly dependent on the actual size of the real environment. We think that the implementation of a layer-based method similar to the Flexible Spaces [49] and the Impossible Spaces [50], in addition to our optimization-based approach, would allow game developers to design longer and more complex game levels (e.g., a dungeon-related game). Moreover, instead of using a small number of game level chunks, we think that experimentation with a large 3D dataset might be useful to generate game levels with enhanced appearance alternatives within the synthesized layout. We hope that more optimization-based approaches that synthesize virtual reality game levels for real environment constraints will be proposed in the near future.

Declaration of Competing Interest

Authors declare that they have no conflict of interest.

References

[1] LaViola JJ Jr, Kruijff E, McMahan RP, Bowman D, Poupyrev IP. 3D user interfaces: theory and practice. Addison-Wesley Professional; 2017.
[2] Slater M. Place illusion and plausibility can lead to realistic behaviour in immersive virtual environments. Philos Trans R Soc B 2009;364(1535):3549–57.
[3] Mousas C, Kao D, Koilias A, Rekabdar B. Real and virtual environment mismatching induces arousal and alters movement behavior. In: IEEE conference on virtual reality and 3Duser interfaces. IEEE; 2020. p. 626–35.
[4] Slater M. Presence and the sixth sense. Presence 2002;11(4):435–9.
[5] Slater M, Brogni A, Steed A. Physiological responses to breaks in presence: apilot study. In: Annual international workshop on presence, 157. Citeseer; 2003. p. 1–4.
[6] Usoh M, Arthur K, Whitton MC, Bastos R, Steed A, Slater M, et al. Walking> walking-in-place> flying, in virtual environments. In: Annual conference on computer graphics and interactive techniques; 1999. p. 359–64.
[7] Razzaque S, Kohn Z, Whitton MC. Redirected walking. In: Eurographics; 2001. p. 1–6.
[8] Suma EA, Bruder G, Steinicke F, Krum DM, Bolas M. A taxonomy for deploying redirection techniques in immersive virtual environments. In: IEEE virtual reality workshops. IEEE; 2012a. p. 43–6.
[9] Sun Q, Patney A, Wei L-Y, Shapira O, Lu J, Asente P, et al. Towards virtual reality infinite walking: dynamic saccadic redirection. ACM Trans Graph 2018;37(4):1–13.
[10] Sun Q, Wei L-Y, Kaufman A. Mapping virtual and physical reality. ACM Trans Graph 2016;35(4):1–12.
[11] Nescher T, Zank M, Kunz A. Simultaneous mapping and redirected walking for ad hoc free walking in virtual environments. In: IEEE virtual reality; 2016. p. 239–40.
[12] Gal R, Shapira L, Ofek E, Kohli P. Flare: fast layout for augmented reality applications. In: IEEE international symposium on mixed and augmented reality. IEEE; 2014. p. 207–12.
[13] Nuernberger B, Ofek E, Benko H, Wilson AD. Snaptoreality: aligning augmented reality to the real world. In: ACM CHI conference on human factors in computing systems; 2016. p. 1233–44.
[14] McGill M, Boland D, Murray-Smith R, Brewster S. A dose of reality: overcoming usability challenges in VR head-mounted displays. In: ACM conference on human factors in computing systems; 2015. p. 2143–52.
[15] Keller M, Exposito F. Game room map integration in virtual environments for free walking. In: IEEE conference on virtual reality and 3Duser interfaces. IEEE; 2018. p. 763–4.
[16] Keller M, Tchilinguirian T. Obstacles awareness methods from occupancy map for free walking in VR. In: IEEE conference on virtual reality and 3Duser interfaces. IEEE; 2019. p. 1012–13.
[17] Sousa M, Mendes D, Jorge J. Safe walking in VR. In: ACM SIGGRAPH international conference on virtual-reality continuum and its applications in industry; 2019. p. 1–2.
[18] Wu F, Rosenberg ES. Combining dynamic field of view modification with physical obstacle avoidance.. In: IEEE conference on virtual reality and 3Duser interfaces; 2019. p. 1882–3.
[19] Kanamori K, Sakata N, Tominaga T, Hijikata Y, Harada K, Kiyokawa K. Obstacle avoidance method in real space for virtual reality immersion. In: IEEE International symposium on mixed and augmented reality. IEEE; 2018. p. 80–9.
[20] Sra M, Garrido-Jurado S, Schmandt C, Maes P. Procedurally generated virtual reality from 3Dreconstructed physical space. In: ACM conference on virtual reality software and technology; 2016. p. 191–200.
[21] Müller P, Wonka P, Haegler S, Ulmer A, Van Gool L. Procedural modeling of buildings. ACM Trans Graph 2006;25(3):614–23.
[22] Parish YI, Müller P. Procedural modeling of cities. In: Annual conference on computer graphics and interactive techniques; 2001. p. 301–8.
[23] Smelik RM, Tutenel T, Bidarra R, Benes B. A survey on procedural modelling for virtual worlds. Comput Graph Forum 2014;33(6):31–50.
[24] Peng C-H, Yang Y-L, Wonka P. Computing layouts with deformable templates. ACM Trans Graph 2014;33(4):1–11.
[25] Jiang H, Yan D-M, Dong W, Wu F, Nan L, Zhang X. Symmetrization of facade layouts. Graph Models 2016;85:11–21.
[26] Ma C, Vining N, Lefebvre S, Sheffer A. Game level layout from design specification. Comput Graph Forum 2014;33(2):95–104.
[27] Hartsook K, Zook A, Das S, Riedl MO. Toward supporting stories with procedurally generated game worlds. In: IEEE conference on computational intelligence and games. IEEE; 2011. p. 297–304.
[28] Hullett K, Mateas M. Scenario generation for emergency rescue training games. In: International conference on foundations of digital games; 2009. p. 99–106.
[29] Xie B, Zhang Y, Huang H, Ogawa E, You T, Yu L-F. Exercise intensity-driven level design. IEEE Trans Vis Comput Graph 2018;24(4):1661–70.
[30] Zhang Y, Xie B, Huang H, Ogawa E, You T, Yu L-F. Pose-guided level design. In: ACM CHI conference on human factors in computing systems; 2019. p. 1–12.
[31] Liu H, Wang Z, Mousas C, Kao D. Virtual reality racket sports: virtual drills for exercise and training. In: IEEE international symposium on mixed and augmented reality; 2020. p. 566–76.
[32] Harrison BE, Roberts D. Analytics-driven dynamic game adaption for player retention in a 2-dimensional adventure game. In: Tenth artificial intelligence and interactive digital entertainment conference; 2014. p. 23–9.
[33] Luo L, Yin H, Cai W, Lees M, Zhou S. Interactive scenario generation for mission-based virtual training. Comput Anim Virtual Worlds 2013;24(3-4):345–54.
[34] Rowe JP, Mott BW, Lester JC. Optimizing player experience in interactive narrative planning: a modular reinforcement learning approach.. In: AAAI conference on artificial intelligence and interactive digital entertainment; 2014. p. 160–6.
[35] Smith AM, Andersen E, Mateas M, Popović Z. A case study of expressively constrainable level design automation tools for a puzzle game. In: International conference on the foundations of digital games; 2012. p. 156–63.
[36] Adams B, Dutré P. Interactive boolean operations on surfel-bounded solids. ACM Trans Graph 2003;22(3):651–6.
[37] Perlin K. An image synthesizer. ACM Siggraph Comput Graph 1985;19(3):287–96.
[38] Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to algorithms. MIT Press; 2009.
[39] Tarjan R. Depth-first search and linear graph algorithms. SIAM J Comput 1972;1(2):146–60.
[40] Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by simulated annealing. Science 1983;220(4598):671–80.
[41] Chib S, Greenberg E. Understanding the metropolis-hastings algorithm. Am Stat 1995;49(4):327–35.
[42] Cercignani C. The Boltzmann equation. In: The Boltzmann equation and its applications. Springer; 1988. p. 40–103.

[43] Regenbrecht H, Schubert T. Real and illusory interactions enhance presence in virtual environments. Presence 2002;11(4):425–34.
[44] Schubert TW. The sense of presence in virtual environments: athree-component scale measuring spatial presence, involvement, and realness.. Z Medienpsychologie 2003;15(2):69–71.
[45] Slater M, Usoh M, Steed A. Depth of presence in virtual environments. Presence 1994;3(2):130–44.
[46] Tcha-Tokey K, Christmann O, Loup-Escande E, Richir S. Proposition and validation of a questionnaire to measure the user experience in immersive virtual environments. Int J Virtual Real 2016;16(1):33–48.
[47] Arthur E, Hancock P, Chrysler S. The perception of spatial layout in real and virtual worlds. Ergonomics 1997;40(1):69–77.
[48] Kuliga SF, Thrash T, Dalton RC, Hölscher C. Virtual reality as an empirical research tool—exploring user experience in a real building and a corresponding virtual model. Comput Environ Urban Syst 2015;54:363–75.
[49] Vasylevska K, Kaufmann H, Bolas M, Suma EA. Flexible spaces: dynamic layout generation for infinite walking in virtual environments. In: IEEE Symposium on 3D user interfaces. IEEE; 2013. p. 39–42.
[50] Suma EA, Lipps Z, Finkelstein S, Krum DM, Bolas M. Impossible spaces: maximizing natural walking in virtual environments with self-overlapping architecture. IEEE Trans Vis Comput Graph 2012b;18(4):555–64.