




Article

Virtual Reality Tool for Exploration of Three-Dimensional Cellular Automata

Camilo Arevalo , Yuta Kariyado  and Julián Villegas * 

Computer Arts Lab., University of Aizu, Aizuwakamatsu 965-8580, Japan; d8231101@u-aizu.ac.jp (C.A.); m5251116@u-aizu.ac.jp (Y.K.)

* Correspondence: julian@u-aizu.ac.jp

Abstract: We present a Virtual Reality (VR) tool for exploration of three-dimensional cellular automata. In addition to the traditional visual representation offered by other implementations, this tool allows users to aurally render the active (alive) cells of an automaton in sequence along one axis or simultaneously create melodic and harmonic textures, while preserving in all cases the relative locations of these cells to the user. The audio spatialization method created for this research can render the maximum number of audio sources specified by the underlying software (255) without audio dropouts. The accuracy of the achieved spatialization is unrivaled since it is based on actual distance measurements as opposed to coarse distance approximations used by other spatialization methods. A subjective evaluation (effectively, self-reported measurements) of our system ($n = 30$) indicated no significant differences in user experience or intrinsic motivation between VR and traditional desktop versions (PC). However, participants in the PC group explored more of the universe than the VR group. This difference is likely to be caused by the familiarity of our cohort with PC-based games.

Keywords: cellular automata; auralization; sonification; VR



Citation: Arevalo, C.; Kariyado, Y.; Villegas, J. Virtual Reality Tool for Exploration of Three-Dimensional Cellular Automata. *Electronics* **2022**, *11*, 497. <https://doi.org/10.3390/electronics11030497>

Academic Editors: Osvaldo Gervasi and JungYoon Kim

Received: 29 December 2021

Accepted: 4 February 2022

Published: 8 February 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Cellular automata (CAs) are one of the earliest computer-based models that mimic some properties of biological systems [1]. In its digital form, an automaton is comprised of a lattice of multi-state cells that, once initialized, can change their state according to some fixed rules. These rules are usually local and simple but can create complex evolutive patterns in the automaton. Although a CA lattice can have arbitrary dimensions, one- or two-dimensional models have been traditionally preferred. In the latter case, square cells have been preferred over triangular or hexagonal ones to create a lattice; these three regular polygons are the only ones capable of tiling the plane without overlaps or gaps [2].

The complexity that CAs can achieve increases with the number of dimensions of their lattice. So does their inspection difficulty, which is usually performed using visual displays. In one-dimensional CAs, the state of a cell in the next iteration depends on its current state and those of the neighboring cells. Considering only adjacent cells, the state of three cells, two adjacent plus the cell itself, would be necessary and sufficient for computing the state of that cell (active or inactive, in the simplest case) in the next iteration. For these three cells, there are $2^3 = 8$ possible state combinations and $2^8 = 256$ rules. Registering the outcomes of each iteration in a second dimension results in patterns that are useful to classify the rules generating these patterns into those yielding homogeneous states (class 1), leading to stable or simply periodic structure (class 2), leading to non-periodic/chaotic-like behaviors (class 3), and those resulting in complex patterns with self-propagating properties (class 4) [3]. Although this classification is not exclusive to one-dimensional CAs, the classes are notoriously more difficult to determine in higher dimensions. In some cases, they have been observed only through two-dimensional slices of the full CAs [3].

In two-dimensional CAs, the number of adjacent cells can be up to eight (including those sharing vertices and edges), so the number of possible state combinations and rules

increase accordingly. Traditionally, outcomes of these rules are mapped to time instead of a third geometric dimension. This is partly because 3D graphics techniques are often required and such techniques evolved long after 2D ones. The latter techniques were already available for many researchers who may have considered further geometrical dimensions as an unnecessary nuisance.

There is no straightforward way to visualize or inspect three- or higher-dimensional CAs. Displays such as Virtual Reality (VR) headsets (featuring stereoscopic vision and spatial audio) could help in these tasks, but they would inherit natural limitations of human perception: vision and audition (the superior sensory modalities in humans) are more acute in front of a person and gradually deteriorate away from that direction [4]. However, audition trumps vision in allowing the perception of objects in all directions, even those which are occluded by closer ones. Hence, presenting spatially distributed sounds allows for larger data representations [5], and could ease the exploration of CAs more complex than one-dimensional ones.

The main purpose of this research is to use VR techniques, including spatial sound, to explore three-dimensional CAs. By providing users with VR simulations, we intend to ease the inspection of such systems and gain higher introspection of their capabilities. We also address whether VR offers benefits in terms of user experience, intrinsic motivation, and subjective performance relative to non-immersive presentations of the same visualization and auralization. Additionally, we explore the use of these automata as a source of generative music: by associating each cell to a different timbre depending on its location, the 3D-automaton may be aurally rendered sequentially or in parallel to create melodic and musical textures. To that end, we have created a VR application where users donning Head-Mounted Displays (HMDs) and headphones are able, among other affordances, to navigate, modify rules, store, and retrieve CAs at will. We have focused on CAs similar to the Game of Life (Life) [6] in the definition of rules and lattice structure for our proof-of-concept application.

The next section presents a literature review on CAs, and their applications in music. This is followed by an explanation on how our VR application and the spatialization engine were created and evaluated, segueing into a general discussion, and concluding remarks. Previous results of this research were presented in [7]. The current manuscript builds upon those results and comments of the reviewers, specifically, elaborating on the results of using HMD.

2. Background

2.1. Life and Life-Like CA

The cellular automaton known as “Life” can be considered a simplification of the processes involved in the creation, evolution, and destruction of organic life. The state of a cell in the next iteration, “alive” or “dead”, is determined by the following rules [8]:

- Birth: A dead cell adjacent to three alive ones becomes alive.
- Survival: Cells adjacent to two or three alive cells remain alive.
- Death: Alive cells adjacent to four or more alive cells, or those adjacent to less than two alive cells die.

Originally confined to a 2D-lattice (as shown in Figure 1), three-dimensional extensions were soon explored. In this case, the only regular solid that can fill the space is the cube. Each cubic cell has up to 26 adjacent cells considering those that share at least a vertex. Consequently, rules governing a cell state in the next iteration were generalized and defined in terms of four integer arguments (r_1, \dots, r_4) in the range of 0 to 26, and the number of neighbors n [9]:

- Birth: A dead cell becomes alive if $r_1 \leq n \leq r_2$.
- Survival: A cell remains alive if $n \geq r_4$ and $n \leq r_3$.
- Death: A cell dies if $n > r_3$ or $n < r_4$.

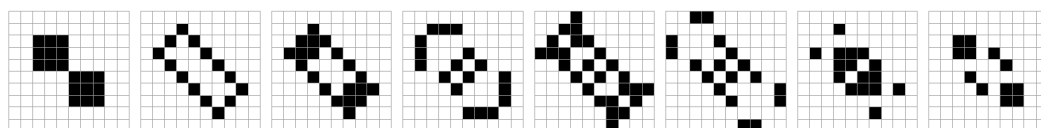


Figure 1. Recreation of “Big beacon,” a period-8 oscillator pattern in Life described in [10].

One of the earliest reports on three-dimensional versions of Life was presented by Bays [11]. He formally defined rules governing this CA in terms of environment and fertility and formulated a set of theorems to construct these rules. Later on, several software implementations of three-dimensional CAs were proposed, for example, “Kaleidoscope of 3D life” [9] and CA3D [12]. A currently abandoned project, Kaleidoscope allowed users to define rules as well as the size of the lattice. Additionally, users could select various known presets and store their own patterns. CA3D is an HTML5 and Javascript program where the initial alive cells are set at random. Intersecting lines in the display are used to mark the center of cells, and users can explore this 3D grid by means of arrow keys or GUI buttons. One of the most impressive software implementations of cellular automata is Golly [13]. This open source, cross-platform application includes features such as bounded and unbounded universes, fast generating algorithms, etc. It also allows creation of 3D models wherein users can define initial conditions or set them at random. As in previously mentioned implementations, Golly inherits the limitations imposed by 2D-displays (e.g., controlling the user perspective by means of buttons), making this kind of CA exploration somewhat cumbersome.

2.2. Generative Music

The paths of music and algorithm-assisted composition crossed centuries before the conception of modern computers [14]. A complete review of these interactions is beyond the scope of this article. Instead, we limit the discussion to some applications of Cellular Automata in music.

WolframTones [3] is arguably the most successful application of CAs to music. Users are presented with high level parameters that drive the generation of music. These parameters target relevant aspects in the generation process: By choosing a CA rule, initial conditions, vicinity range, among others, users can generate musical patterns according to different styles, orchestration, tuning system, tempo, etc. Admittedly, the resulting musical pattern does not always agree with the selected style, but it could be used as initial step towards a more personalized creation.

Further, CAs have been used in granular synthesis of sounds [15]; to map cells directly to music notes, as in the case of CAMUS (Cellular Automata Music) [16]; or to sequence chord progressions (with limited success) [17]. Recently, Haron and colleagues [18] introduced an interactive sequencer where the movement of a user was employed to determine the configuration of Life in a 20×20 space. This automaton mapped cells to musical pitches, depending only on a single dimension (x -axis). The three-dimensional space was scanned sequentially (using different strategies) to create musical patterns.

Using massive parallel processing hardware (Field-Programmable Gate Arrays—FPGAs), Nedjah et al. [19] used CAs for generation of melodic intervals complying with the Musical Instrument Digital Interface (MIDI) standard [20]. More recently, an algorithm capable of learning CA rules from MIDI sequences has been proposed in [21]. In this research, CA rules evolve with time and provide a way to produce new MIDI sequences based on them.

2.3. Auralization

Associating sounds directly with cell locations can be considered a case of data sonification [4]. In addition to cell–tone associations, the relative location of each cell with respect to a listener can also be beneficial to data exploration via sound spatialization, in what is known as auralization [22]. Whereas the mapping of cells onto sounds is arbitrary and offer limited difficulty from the software development point of view, correct spatialization of

audio sources is difficult, especially in the near-field, i.e., when “The acoustic field is so close to an extended source that the effects of the source size are manifest in measurements” [23]. One reason for this is that the filters used to create binaural signals from monophonic ones are captured in anechoic conditions under the assumptions that (1) the sound source is small compared to the head, and (2) it is far enough that the wave front is planar. These assumptions are often violated in the near-field and when not, the recording process is painstaking, so with the exception of a few (e.g., [24–26]), most filters are recorded at a fixed distance outside the near-field. As a consequence, whereas horizontal and vertical angles (azimuth and elevation, respectively) are often accurately simulated in VR scenes, distance (the third dimension) is mostly approximated by means of different attenuation curves [27,28]. We have devised a better spatialization engine that correctly expresses the three dimensions, as explained in the following section.

3. Method

We decided to take full advantage of VR technologies (stereoscopic vision, spatial audio, limb- and head-tracking, etc.) to facilitate exploration of three-dimensional CAs. We also provide a desktop version (inheriting the interaction limitations pointed before) since VR technologies are not mainstream yet among consumers. The desktop version uses the same audio spatialization technique, and it was amply discussed in [7]. Although we focus on the VR version in this manuscript, pointers to the desktop version are added in the discussion when necessary.

Our proof-of-concept application was programmed in Unity [28] since this platform allows deployment of applications to several platforms, including VR headset- and desktop-based ones [29], and the integration of alternative audio spatializers. Source code, executable versions, and video demonstrations of this project are freely available from <https://github.com/YKariyado/LG> (accessed on 1 February 2022). In what follows, we explain the main components of our solution.

3.1. Apparatus

Users of our system donned an HMD comprising 2880×1600 pixel stereoscopic display (HTC VIVE headset) and a pair of closed-circumaural headphones (Sony MDR-CD380). A single trackable controller with multi-function trackpad, grip, trigger, and other buttons were used to interact with the system, as shown in Figure 2a. Tracking the locations of the headset and the controllers as users moved through space was enabled by beaming infrared light from two HTC VIVE base stations. Headset, headphones, and infrared stations were connected directly to a server running Windows 10, equipped with an Intel i7 processor (six cores), NVIDIA GeForce GTX 1080 graphic card, 16 GB of RAM, and a 500 GB solid state drive for storage. This apparatus was located in a quiet room (average $RT_{60} = 322$ ms and Noise Criterion $NC = 25$ [500]). The actual simulation area within this room was about 2 m^2 . An overview of the settings of our system is shown in Figure 2b.

3.2. Cellular Automata

We restricted our simulation to a cubic space with an edge length $12 \leq l \leq 2048$ cells, defined by the user. As in Life, each cell has only two states and the rules for birth, survival, and death were built based upon the same four arguments r_1, \dots, r_4 previously described in Section 2.1. Users can also determine the behavior of the automaton at its boundaries: if “periodic”, opposite sides of the l -cube are treated as contiguous so that cells in these sides become adjacent.

A large number of cells in the automaton that may need to be processed at a certain time could generate considerable latency between generations. For instance, the system may not be able to compute the next generation when it needs to be displayed. To minimize latency between the computation of the CA and its display (video and audio), we divided the workload into three parallel threads: A main thread to manipulate all visual assets

of the game engine, and two auxiliary threads, one to compute the CA model according to the user-defined parameters (rules, periodicity, etc.), and the other dedicated solely to rendering audio.

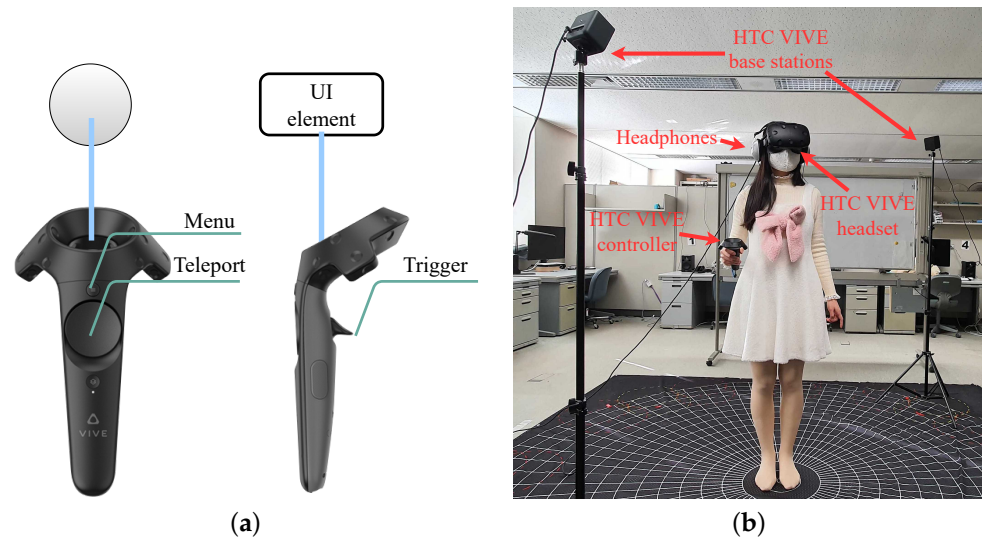


Figure 2. HTC VIVE controller with implemented user interface functionalities and a user in the real settings. (a) Controller. (b) Real settings.

The current population of alive cells is maintained in a hash table of cell IDs. Iterating through all elements in this table, the number of alive neighbors n for a given cell is computed and added to a dictionary comprising pairs of cell ID and its corresponding n . Then, death and birth rules (in this order) are used to update the hash table, deleting or adding IDs as needed. Simultaneously, a 3D sparse matrix is gradually filled with alive cell coordinates. When a new CA iteration is completely computed, its corresponding matrix is added to a queue of matrices where future iterations are stored. When the main thread requires a new CA iteration, the oldest matrix in the queue is extracted. In addition, to minimize the latency introduced by refreshing the display, the main thread uses a sub-routine for swapping between two buffers of visible alive cells. A foreground buffer contains the cells currently in display while in a background one cells are being rendered. Further, the audio and video representations are synchronized in the same sub-routine by setting sonification parameters (i.e., file to playback, center frequency f_c , and Q -factor of a filter), as explained in Section 3.4.

3.3. Navigation

Our simulation was comprised of Menu and Rendition modes, as shown in Figure 3. Initially presented with the Menu mode, users are able to switch between modes by pressing the Menu button of a controller (see Figure 2a). In the Menu mode (Figure 3a), users can adjust aspects of the CA such as length of the cube l , rule parameters, initially alive cells, behavior of the automaton at the boundaries, etc. Users can also store interesting CAs or retrieve previously stored ones for further inspection. The Menu mode allows users to define some aspects of the rendition such as its speed (in beats per minute), whether presenting the alive cells simultaneously or in a sequence of slices determined by their z -coordinate, start or stop the automaton, and when the sequential presentation is enabled, whether or not to follow the current slice, or choose a given slice for better inspection. Interactions with the user virtual interface are done by pointing to a given element (text-, check-box, slider, etc.) with a virtual laser-like beam and pressing the trigger button in the controller. When text boxes are selected, input was registered from a virtual keyboard.

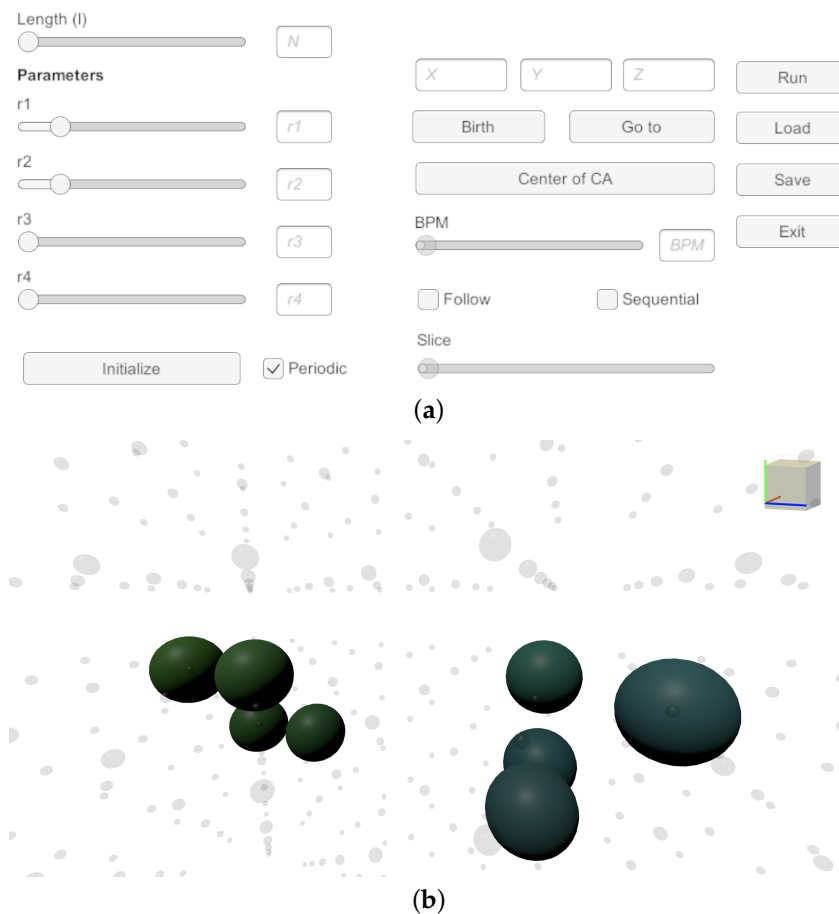


Figure 3. Menu and rendition mode screen shots. (a) Menu Mode. (b) Rendition Mode.

While the size of the universe is controllable by the user, displaying a large number of sources simultaneously becomes impractical. To avoid that, we fixed the display to a radius of six cells around the user in the Rendition mode. Only alive cells within this radius (max. 1728) are actively rendered: visually, by spheres whose color depended on their absolute location in the mesh (mapping Red, Green, and Blue channels to x , y , and z axes, respectively); aurally, by a sound depending on its current location. Dead cells, on the other hand, are represented by small semi-transparent black spheres with no sound.

Initially, users are located at the center of the l -cube, but they are free to move to different locations of the automaton by physically walking around the simulation area, entering a set of coordinates in the Menu mode, by locating the camera at the center of the universe, or by means of the teleporting method of the VIVE Input Utility [30]. Besides the teleportation method, the displacement by controller/joystick method was also considered. However, and as it has been reported by Boletsis et al. [31], due to the motion sickness produced in pilot experiments, this method was discarded. The current location of a user within the universe is always visible at the top-right corner of the visual display, as shown in Figure 3b. In the VR version, video and audio are presented from an endocentric point of view [32], i.e., cells in the near field are centered at the middle of the head. For the desktop version, while audio is still endocentric, the video is presented from an egocentric point of view, effectively tethering a visual camera behind the head of a user's avatar.

3.4. Sonification

To represent the CA with sound, we opted for an auditory display in which features of the same musical timbre changed relative to the location and state of a cell. Since one of the design goals was to create musical sounds from CAs, the large maximum number of cells in each dimension (2048) became challenging, especially for representing pitch. There are

many ways to map music scale features onto 3D spaces [33], but we resorted to Shepard tones [34] to finesse the problem of mapping pitch. Shepard tones take advantage of the circularity judgement of pitch (chroma) by which octaves of the same tone are considered to be at the same location, independent of their absolute frequency.

For our environment, individual WAV files (sampled at 16 bit/48 kHz) for each Shepard tone were created in Matlab [35]. Twelve Shepard tones separated 100 cents from each other (i.e., a 12-TET—Tone Equal Temperament tuning) were created, each tone comprising 10 octaves. The lowest Shepard tone was set to have a first harmonic $f_0 = 22$ Hz. These complex tones were 250 ms long, and amplitude modulated to have instantaneous attack time, 63 ms of decay and release time, and -4.46 dB (re. full-scale) sustain level.

In his invention, Shepard also included a band-pass filter. The effect of this filter is to reinforce sensation of a pitch at its center frequency f_c and to let the tone harmonics to gradually fade as they recede from this frequency. In our implementation, f_c took values between 100 Hz and 10 kHz in l equal intervals of a logarithmic scale and were mapped onto the x (lateral) axis, while Shepard tones in ascending order were mapped onto the y (vertical) axis, repeating each tone every twelve cells. The quality factor Q of the band-pass filter was mapped to the z (longitudinal) axis in equal intervals of a linear scale between 1 and 100. Whereas Shepard tones were computed in advance, filtering was implemented in Faust [36] and integrated in the audio pipeline so that it was computed online, previous to spatialization, as shown in Figure 4.

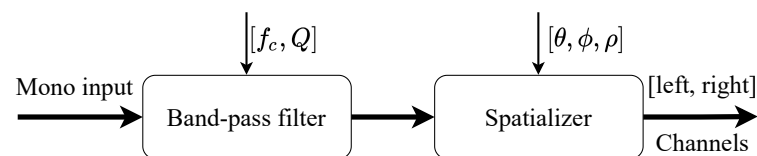


Figure 4. DSP in the audio thread. Mono input signals coming from audio files are first band-pass filtered and then spatialized, creating a stereo signal that is sent to the audio buffer in Unity.

3.5. Audio Spatialization

Each alive cell in our system is an audio source whose location relative to the user is accordingly simulated. In Unity, audio sources are divided into audible (real) and inaudible (virtual) ones [37]. There are up to 255 audible sources at any given time in Unity. They are determined by a programmable priority, distance to a listener (closer/louder sources are promoted), and everything being equal, whichever audio is queued first, in that hierarchy. In our simulation, 1728 cells could be alive at any point. These are visually rendered without difficulty by the application but, because of the aforementioned restrictions, only 255 are audio rendered.

To spatialize these audio sources, we implemented a plugin filter based on a reconstruction of Head-Related Transfer Functions (HRTFs) by Eigen decomposition, as described in [38]. In short, this method reconstructs HRTFs by adding to the mean HRTF computed from a distance-dependent HRTF database [25] the products of Eigenvalues and Eigenvectors specific to a given location until a spectral distortion criterion is reached (<1 dB between 0.1 and 16 kHz). Convolution of an HRTF with a monophonic audio is an effective way to produce a spatialized version of it at the location where the HRTF was captured [39].

Not all possible locations in a virtual scene are included in HRTF databases. In such cases, interpolation between the closest HRTFs to the desired location is commonly used [40], but it is also possible to use only the closest HRTF when processing time is an issue. In our plugin, space around a listener is discretized differently depending on spherical coordinates: For elevation, $-40^\circ \leq \phi \leq 90^\circ$, in 2° steps. For azimuth θ , the discretization depends additionally on elevation: $\theta_\phi = 359^\circ \cos(\phi) + 1^\circ$. Finally, for distance ρ , the original intervals at which the HRTFs were captured in [25] (i.e., [20, 30], [30, 40], ..., [130, 160] cm) are sub-divided into ten equal parts. With these changes, the HRTF database used for our spatializer comprises 1,212,680 locations. No HRTF interpolation is applied in our solution, so cell locations are approximated by the closest entry in our database.

The spatialization is performed on the previously mentioned audio thread by doing frequency-domain multiplication of the HRTF retrieved from the database and the fast-Fourier transformed audio signal associated with each alive cell. The simulation area was somewhat larger than the maximum distance recorded in our database, so the actual distance in the room was scaled down to fit that of the spatializer. Furthermore, audio sources are disabled once their corresponding audio file has been played. That way, the audio thread can incorporate incoming sources from the next iteration of the CA.

To demonstrate our system, we present video recordings of a user interacting with the system on our repository [41]. Note that in order to render this monoscopic video from the original one (stereoscopic), some visual elements (such as the three-axes for navigation) are displaced. However, the auralization is correctly preserved. In addition, we shipped our system with initial configurations that yielded interesting patterns after 100 iterations of the automaton. We mainly found oscillators such as those reported previously in [9], for which a detailed description is given in [7]. These initial configurations can be loaded from the Menu mode, as previously explained.

4. Objective Evaluation

Performance tests were carried out on a MacBook Pro 2019 (processor Intel core I9-9880H with 8 cores at 2.3 GHz and DDR4 SDRAM memory with 32 GB at 2666 MHz) running Unity version 2020.1.9f1. Audio in the desktop and VR versions ran at the same sampling frequency (48 kHz) but differ in the size of the block used for processing audio: 256 and 1024 samples for the Desktop and VR version, respectively. The block size determines the maximum time that the audio thread could use to avoid audio dropouts (5.333 and 21.333 ms, respectively).

To test the spatializer capabilities, we replaced audio files corresponding to cells in the CA by longer WAV files (60 s long sampled at 48 kHz/16-bits) since our short files posed no processing difficulties. These longer files were successively added every 5 s and the CPU time consumed by the spatializer was monitored using the Unity profiler [42]. Results of these tests are shown in Figure 5.

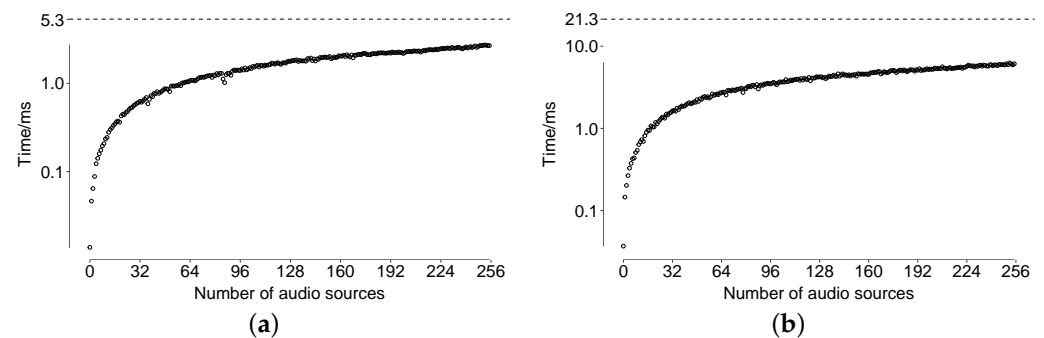


Figure 5. CPU time used for the spatializer for different number of audio sources with different audio block sizes: 256 samples (Desktop version) and 1024 samples (VR version). Dashed lines indicate the block processing time in each case. (a) 256 samples. (b) 1024 samples.

For a block of 256 samples, the most stringent case, we found that on average the maximum CPU time used by the spatializer was 2.735 ms, whereas for the 1024 sample block, this time was 6.247 ms. These figures represent 51% and 29% of the block times, indicating that the maximum number of audio sources determined by default in Unity (255) can be spatialized without audio drop-outs with our current hardware set-up.

5. Subjective Evaluation

We performed a subjective experiment to compare the desktop and VR versions of our system. In addition to the performance achieved by the subject (i.e., covered distance and number of targets identified, as explained later), we measured user experience (UX) by

means of the UEQ questionnaire [43], a set of 26 questions and interpretations, and intrinsic motivation (IMI) [44], another multidimensional scale, by means of 20 questions.

5.1. Participants

Permission for performing this experiment was obtained following the University of Aizu ethics guidelines. A total of 31 young adult students (22 year old, $\sigma = 2.0$) from the University of Aizu volunteered for this experiment. Data from one of them was excluded from further analyses, as this participant had hearing thresholds > 20 dB (HL) in at least two bands (0.125–8 kHz). Hearing thresholds were measured with an MA25 audiometer (MAICO, Eden Prairie, MN, USA). Participants were mostly Japanese right-handed males (only 4 were non-Japanese, 5 left-handed, and 3 females) who claimed to play regular video-games 8 h a week on average (min. = 0, max. = 35, median = 5). Only three participants claimed to regularly play VR games (1, 5, and 7 h a week).

5.2. Materials

The experiment was conducted with a set-up similar to that described in Section 3.1. For this experiment, a pair of HD 380 pro (closed) headphones (Sennheiser, Wedemark, Germany) was used instead. Sound level was adjusted to be comfortable for the participants.

A game based on our CA visualization system was implemented. This game was comprised of two scenarios (demonstration and main task), and was built for VR and regular desktop. The main difference between the two versions lies in their navigation interface: While the navigation for the VR version is the same described in Section 3.3, the keys 'W,A,S,' and 'D' of a QWERTY keyboard were used to move forward, left, back, and right, respectively, as it is commonly found in PC games. In addition, left-click was enabled to rotate the camera and interact with UI elements in the Menu mode, and right-click for identifying a target pattern.

The game had a fixed duration of 10 min (the remaining time was always visible), and participants earned points by finding a target pattern (pressing the trigger button in VR settings, or right-clicking in the PC). The current score was also always visible. Correctly identified patterns were indicated by visual effects and a unit increment in their score. Participants were unable to further interact with correctly identified patterns. Incorrectly identified patterns were also indicated by different visual effects, but participants were able to continue interacting with them.

The demonstration scenario was presented first. It was comprised of four patterns, two repeated targets, and two distractors. For the main task scenario, elapsed time, pattern (target or distractor), and user locations were registered for analysis. In all cases, the patterns were oscillators to prevent that possible interactions between patterns could cause its annihilation. The automaton was set in advance, with 400 patterns (100 targets) distributed among the universe.

5.3. Procedure

The task was to play the described game and earn as many points as possible. After that, participants were instructed to fill out a questionnaire. Since one of the UEQ dimensions is novelty, and VR-based presentations are usually regarded as a more novel than desktop-based ones, participants were randomly assigned to two groups: the control group played the PC version of the game and the experimental group played the VR version.

A session started with a brief explanation of 3D cellular automata. After that, an explanation of how to interact with the game was verbally provided. During the demonstration scenario, participants were asked to memorize the target pattern by looking at it from different perspectives. They were also monitored on how they performed the task, providing additional instructions when necessary. When the participants were judged to be proficient at the game, the main task scenario was presented. No feedback or further explanations were provided.

After finishing the main task scenario, a questionnaire including UEQ and IMI questions was answered by the participants in a mandatory fashion. UEQ and IMI questions were answered using a discrete scale marked from 1 to 7. The UEQ dimensions were derived from questions where participants needed to rate the experience between two polar values such as “annoying/enjoyable”, “creative/dull”, etc., mapped to the extremes of the scale. The low and high extremes of the scale for IMI statements (e.g., “I think I am pretty good at this task”, “I thought the task was very boring”, etc.) were marked with “not at all true” and “very true”, respectively. All interactions with the participants (verbal and written) were made in either Japanese or English, according to their preference.

On average, participants spent less than 30 min to finish the whole experience. Finally, as a measure against the spreading of COVID-19, participants were requested to wear masks and nitrile gloves. Likewise, keyboard, headphones, mouse, etc., were disinfected after each participant and the room was ventilated. Only one participant was allowed in the room during the experiment.

5.4. Results

The UEQ questionnaire evaluates UX in six different dimensions: attractiveness, dependability, efficiency, novelty, perspicuity, and stimulation. These dimensions are derived from 7-point scale questions and zero-centered. The UEQ also allows for benchmarking results against data from 468 studies (21,175 persons). The VR version of the game was rated “excellent” in attractiveness when compared with the benchmark data while the PC version was rated “good”. However, differences between the two groups were not significant, as indicated by a series of between-subject ANOVAs performed with the ‘ez’ library [45] in R [46]. These results are shown in Table 1 and Figure 6a.

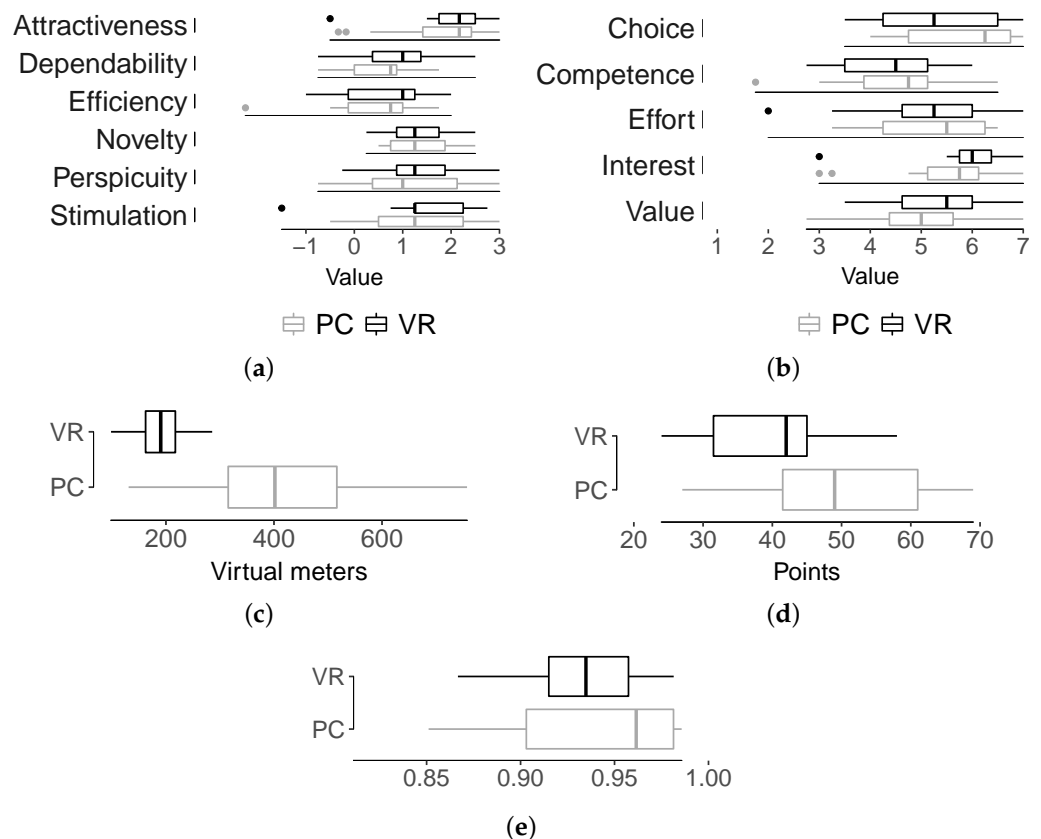


Figure 6. Results from the subjective experiment. The lines inside boxes show the median, first, and third quartiles (hinges), 1.5 times the inter-quartile range left and right the hinges (whiskers), and outlying points. (a) UEQ. (b) IMI. (c) Virtual distance. (d) Scores. (e) Precision.

Table 1. Results from the between-subject ANOVA for UEQ and IMI.

| UEQ | | IMI | |
|----------------|-------------------------------|------------|-------------------------------|
| Dimension | Statistics | Dimension | Statistics |
| Attractiveness | $F(1, 28) = 0.610; p = 0.441$ | Choice | $F(1, 28) = 0.966; p = 0.334$ |
| Dependability | $F(1, 28) = 1.291; p = 0.266$ | Competence | $F(1, 28) = 0.082; p = 0.776$ |
| Efficiency | $F(1, 28) = 0.313; p = 0.580$ | Effort | $F(1, 28) = 0.033; p = 0.856$ |
| Novelty | $F(1, 28) = 0.004; p = 0.948$ | Interest | $F(1, 28) = 1.296; p = 0.264$ |
| Perspicuity | $F(1, 28) = 0.098; p = 0.756$ | Value | $F(1, 28) = 1.152; p = 0.226$ |
| Stimulation | $F(1, 28) = 0.313; p = 0.580$ | | |

The IMI questionnaire evaluates the intrinsic motivation of the participants in five dimensions: (perceived) choice, (perceived) competence, effort, interest, and (perceived) value. No significant differences between the two groups were found for any of these dimensions, as shown in Figure 6b and Table 1.

The PC group traveled on average more than twice the distance traveled by the VR group, as shown in Figure 6c. This difference was significant [$F(1, 28) = 27.296; p < 0.001; \eta_G^2 = 0.493$]. In a similar fashion, the PC group earned on average 10 points more than the VR group. This difference, shown in Figure 6d, was also significant [$F(1, 28) = 6.017; p = 0.021; \eta_G^2 = 0.001$]. However, the PC group also misidentified more targets than the VR group and this difference was not significant, as indicated by an ANOVA on the arcsine-transformed fractions between correct and number of responses [$F(1, 28) = 0.029; p = 0.865$] and shown in Figure 6e.

6. Discussion and Future Work

Auralization in our case is limited by the number of audio sources that Unity can render simultaneously. Our results show that the implemented spatializer was able to successfully handle these audio sources without dropouts. Life models capable of presenting even thousands of cells dwarf our proposal. However, we found that our small mesh was large enough to challenge the limitation of the underlying VR platform (Unity) and to successfully generate musical patterns with interesting rhythmical possibilities. We are now working on offering greater freedom in the mapping of acoustic features to cell properties by means of an editor.

In a previous report [7], we included the results of a subjective experiment ($n = 21$) performed with the desktop version of our system. We reported that immersion and perceived quality was rated significantly higher when the audio spatialization was used than when no audio was presented. We also found no significant differences between the effects of sequential and simultaneous presentation. Our study suggests that any differences between VR and PC presentations are rather small and not observable with our sample size. These results are consistent with those of An et al. [47], who found no performance difference between the two groups (PC and VR) while playing a cultural game. More recently, the effect of platform (PC vs. VR) has been found to be less important than embodiment (opportunities to interact with and manipulate virtual assets) [48]. Furthermore, a limitation in our study due to the familiarity of our cohort with PC video games and lack of experience with VR games could explain the similarities between these two groups. Familiarity with PC video games could also explain the larger distance averaged by our PC cohort since few VR participants used the teleporting method. The method was decided due to the motion sickness produced in pilot experiments using other methods of displacement, such as controller/joystick method. Thus, we are interested in repeating this experiment with a larger and more balanced sample in terms of familiarity with both platforms.

The results obtained from our CA visualization and auralization indicate that they are beneficial for the exploration of patterns within these models. In terms of immersion and perceived quality, subjective experiments based on a desktop version of our system also indicate improvements over visual-only explorations. In our simulations, we opted for an auditory display where band-pass filtered tones vary on pitch, center frequency,

and quality factor of the filter, depending on the length of the CA. Although this arbitrary mapping helps to aurally explore the CA, it falls short in our ambitions to generate enticing music from it. As much as we adhere to the definition of music offered by Berio that “Music is everything that one listens to with the intention of listening to music” [49], we consider that other mappings could generate musical patterns more familiar to many users, for example by using the circle of fifths (a different circular judgement of western musical scales). In the same vein, since some rules in certain CAs have been proven to be universal Turing machines, like rule 110 in one-dimensional CA [50] or the Game of Life [51], we are investigating the generation of arbitrary melodies using Genetic Algorithms. In this case, we look for a set of rules and an initial state that results in a given melody after a certain number of CA iterations. We defer, however, such endeavors to further reports.

7. Conclusions

A VR tool for the exploration of three-dimensional CAs was introduced. In addition to the visual representation and exploration by means of HMDs, the implemented tool renders the CA using spatial sound. The audio spatializer in our research is capable of rendering the maximum number of audio sources allowed by the game engine. The spatializer developed for our research uses actual HRTF measurements, so accurate renditions of sources in the near field can be achieved.

Contrary to our expectations, we found no benefit of exploring CAs using VR relative to non-immersive explorations in terms of user experience, intrinsic motivation, and subjective performance. The reasons for this outcome are not well understood, but they have been reported in the literature. It is possible that familiarity with desktop interfaces has a larger effect than we expected.

Author Contributions: Formal analysis, J.V.; Investigation, C.A., Y.K. and J.V.; Resources, C.A. and Y.K.; Software, C.A. and Y.K.; Validation, J.V.; Visualization, C.A. and J.V.; writing—original draft preparation, C.A., Y.K. and J.V.; writing—review and editing, C.A. and J.V. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: The study was conducted following the University of Aizu ethics guidelines.

Informed Consent Statement: Verbal informed consent was obtained from all subjects involved in the study.

Data Availability Statement: Not applicable.

Acknowledgments: The authors express their gratitude to Michael Cohen, anonymous reviewers of this manuscript, and editors of Electronics for their invaluable comments and suggestions.

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

References

1. Von Neumann, J. The general and logical theory of automata. In *Cerebral Mechanisms in Behavior; The Hixon Symposium*; Jeffress, L.A., Ed.; Wiley: Oxford, UK, 1951; pp. 1–41.
2. Grünbaum, B.; Shephard, G.C. Tilings by regular polygons. *Math. Mag.* **1977**, *50*, 227–247. [[CrossRef](#)]
3. Wolfram, S. *A New Kind of Science*; Wolfram Media: Champaign, IL, USA, 2002.
4. Hermann, T.; Hunt, A.; Neuhoff, J.G. *The Sonification Handbook*; Logos: Berlin, Germany, 2011.
5. Roginska, A.; Childs, E.; Johnson, M.K. *Monitoring Real-Time Data: A Sonification Approach*; Georgia Institute of Technology: Atlanta, GA, USA, 2006.
6. Rendell, P. *Turing Machine Universality of the Game of Life*; Springer: Berlin/Heidelberg, Germany, 2016.
7. Kariyado, Y.; Arévalo, C.; Villegas, J. Auralization of Three-Dimensional Cellular Automata. In *Artificial Intelligence in Music, Sound, Art and Design*; Springer International Publishing: New York, NY, USA, 2021; pp. 161–170. [[CrossRef](#)]
8. Gardner, M. The fantastic combinations of John Conway’s new solitaire game “life”. *Sci. Am.* **1970**, *223*, 120–123. [[CrossRef](#)]
9. Demidov, E. Kaleidoscope of 3D Life. 2000. Available online: <https://www.ibiblio.org/e-notes/Life/Game.htm> (accessed on 1 February 2022).
10. Simon Norton. Figure Eight. 1970. Available online: https://conwaylife.com/wiki/Figure_eight (accessed on 1 February 2022).

11. Bays, C. Candidates for the game of life in three dimensions. *Complex Syst.* **1987**, *1*, 373–400.
12. Demidov, E. CA3D. 2000. Available online: <https://grelf.net/gr3d/ca3d.html> (accessed on 1 February 2022).
13. Delahaye, J.P. Le royaume du Jeu de la vie (The kingdom of the Game of Life). *Pour Sci.* **2009**, *378*, 86–91. (In French)
14. Fernández, J.D.; Vico, F. AI methods in algorithmic composition: A comprehensive survey. *J. Artif. Intell. Res.* **2013**, *48*, 513–582. [[CrossRef](#)]
15. Miranda, E.R. Granular synthesis of sounds by means of a cellular automaton. *Leonardo* **1995**, *28*, 297–300. [[CrossRef](#)]
16. Miranda, E.R. Cellular automata music: From sound synthesis to musical forms. In *Evolutionary Computer Music*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 170–193. [[CrossRef](#)]
17. Morris, H.; Wainer, G.A. Music Generation Using Cellular Models. In Proceedings of the 2012 Symposium on Theory of Modeling and Simulation-DEVS Integrative M&S Symposium, Orlando, FL, USA, 26–30 March 2012.
18. Haron, A.; Soon Xuan, Y.; Chee Onn, W. Cellular music: An interactive game of life sequencer. In Proceedings of the 26th ACM international conference on Multimedia, Seoul, Korea, 22–26 October 2018; pp. 2081–2083. [[CrossRef](#)]
19. Nedjah, N.; Bezerra, H.D.; Mourelle, L.M. Automatic generation of harmonious music using cellular automata based hardware design. *Integration* **2018**, *62*, 205–223. [[CrossRef](#)]
20. Association of MIDI Manufacturers. In *The Complete MIDI 1.0 Detailed Specification*, 2nd ed.; Association of MIDI Manufacturers: Los Angeles, CA, USA, 2006
21. Delarosa, O.; Soros, L.B. Growing MIDI Music Files Using Convolutional Cellular Automata. In Proceedings of the 2020 IEEE Symposium Series on Computational Intelligence (SSCI), Canberra, Australia, 1–4 December 2020; pp. 1187–1194. [[CrossRef](#)]
22. Summers, J.E. What exactly is meant by the term “auralization?”. *J. Acoust. Soc. Am.* **2008**, *124*, 697. [[CrossRef](#)] [[PubMed](#)]
23. American National Standards Institute. *Standard Acoustical & Bioacoustical Terminology Database*; ANSI/ASA S1.1 & S3.20; American National Standards Institute: New York, NY, USA, 2013.
24. Hosoe, S.; Nishino, T.; Itou, K.; Takeda, K. Measurement of Head-Related Transfer Functions in the Proximal Region. In Proceedings of the Forum Acusticum, Budapest, Hungary, 29 August–2 September 2005; pp. 2539–2542.
25. Qu, T.; Xiao, Z.; Gong, M.; Huang, Y.; Li, X.; Wu, X. Distance-Dependent Head-Related Transfer Functions Measured with High Spatial Resolution Using a Spark Gap. *IEEE Trans. Audio Speech Lang. Process.* **2009**, *17*, 1124–1132. [[CrossRef](#)]
26. Wierstorf, H.; Geier, M.; Spors, S. A free database of head related impulse response measurements in the horizontal plane with multiple distances. In *Audio Engineering Society Convention*; Audio Engineering Society: New York, NY, USA, 2011.
27. Epic Games. Unreal Engine. v.4.27. Available online: <https://docs.unrealengine.com/> (accessed on 1 February 2022).
28. Unity Technologies. Unity. v.2020.1.6f1. Available online: <https://unity.com> (accessed on 1 February 2022).
29. Barczak, A.; Woźniak, H. Comparative Study on Game Engines. *Studia Inform. Syst. Inf. Technol.* **2019**, *1*, 5–24.
30. Corporation, H. Vive Input Utility—Developer Resources. Available online: <https://developer.vive.com/resources/vive-sense/tool/vive-input-utility/> (accessed on 29 September 2021).
31. Boletsis, C.; Cedergren, J.E.; Porta, M. VR Locomotion in the New Era of Virtual Reality: An Empirical Comparison of Prevalent Techniques. *Adv. Hum.-Comp. Int.* **2019**, *2019*, 7420781. [[CrossRef](#)]
32. Cohen, M.; Villegas, J. Applications of audio augmented reality. Wearware, everywhere, anywhere, and awareware. In *Fundamentals of Wearable Computers and Augmented Reality*, 2nd ed.; CRC Press: Boca Raton, FL, USA, 2015; Chapter 13, pp. 309–329.
33. Villegas, J.; Cohen, M. Mapping Musical Scales Onto Virtual 3D Spaces. In *Principles and Applications of Spatial Hearing*; Suzuki, Y., Brungart, D., Kato, H., Iida, K., Cabrera, D., Iwaya, Y., Eds.; World Scientific: Singapore, 2011. [[CrossRef](#)]
34. Shepard, R. Circularity in Judgments of Relative Pitch. *J. Acoust. Soc. Am.* **1964**, *36*, 2345–2353. [[CrossRef](#)]
35. Mathworks. Matlab. Software. 2020. Available online: www.mathworks.com (accessed on 1 February 2022).
36. Lab, G.R. Faust: Functional Programming Language for Real Time Signal Processing. v.2.33.1. Available online: <https://faust.grame.fr> (accessed on 1 February 2022).
37. John. 10 Unity Audio Optimisation Tips—Game Dev Beginner. 2019. Available online: https://gamedevbeginner.com/unity-audio-optimisation-tips/#voice_limits (accessed on 1 February 2022).
38. Arévalo, C.; Villegas, J. Compressing Head-Related Transfer Function databases by Eigen decomposition. In Proceedings of the 2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP), Tampere, Finland, 21–24 September 2020. [[CrossRef](#)]
39. Iida, K. *Head-Related Transfer Function and Acoustic Virtual Reality*; Springer: Berlin/Heidelberg, Germany, 2019.
40. Villegas, J. Locating virtual sound sources at arbitrary distances in real-time binaural reproduction. *Virtual Real.* **2015**, *19*, 201–212. [[CrossRef](#)]
41. Yuta, K. Auralization of Three-Dimensional Cellular Automata. Available online: <https://github.com/YKariyado/LG/> (accessed on 29 September 2021).
42. Unity Technologies. Unity—Manual: Profiler Overview. Available online: <https://docs.unity3d.com/2020.1/Documentation/Manual/Profiler.html> (accessed on 19 November 2020).
43. Schrepp, M.; Hinderks, A.; Thomaschewski, J. Design and Evaluation of a Short Version of the User Experience Questionnaire (UEQ-S). *Int. J. Interact. Multimed. Artif. Intell.* **2017**, *4*, 40–44. [[CrossRef](#)]
44. McAuley, E.; Duncan, T.; Tammen, V.V. Psychometric Properties of the Intrinsic Motivation Inventory in a Competitive Sport Setting: A Confirmatory Factor Analysis. *Res. Q. Exerc. Sport* **1989**, *60*, 48–58. [[CrossRef](#)] [[PubMed](#)]

45. Lawrence, M.A. *ez: Easy Analysis and Visualization of Factorial Experiments*; R Package Version 4.4-0; 2 November 2016. Available online: <https://CRAN.R-project.org/package=ez> (accessed on 1 February 2022).
46. R Core Team. *R: A Language and Environment for Statistical Computing*; Version 4.1.2; R Foundation for Statistical Computing: Vienna, Austria, 2021.
47. An, B.; Matteo, F.; Epstein, M.; Brown, D.E. Comparing the Performance of an Immersive Virtual Reality and Traditional Desktop Cultural Game. In Proceedings of the 2nd International Conference on Computer-Human Interaction Research and Applications (CHIRA 2018), Seville, Spain, 19–21 September 2018; pp. 54–61. [[CrossRef](#)]
48. Johnson-Glenberg, M.C.; Bartolomea, H.; Kalina, E. Platform is not destiny: Embodied learning effects comparing 2D desktop to 3D virtual reality STEM experiences. *J. Comput. Assist. Learn.* **2021**, *37*, 1263–1284. [[CrossRef](#)]
49. Berio, L.; Dalmonte, R.; Varga, B.A.; Osmond-Smith, D. *Two Interviews*; M. Boyars: New York, NY, USA, 1985.
50. Cook, M. Universality in Elementary Cellular Automata. *Complex Syst.* **2004**, *15*, 1–40.
51. Rendell, P. A Universal Turing Machine in Conway's Game of Life. In Proceedings of the 2011 International Conference on High Performance Computing & Simulation, Istanbul, Turkey, 4–8 July 2011; pp. 764–772. [[CrossRef](#)]