Creating Audio Games Online with a Browser-Based Editor

Michael Urbanek
TU Wien
Vienna, Austria

Michael Habiger
TU Wien
Vienna, Austria

Florian Güldenpfennig
New Design University
St. Pölten, Austria

ABSTRACT

Play has been identified as a fundamental human desire (see, e.g., Huizinga's seminal work on "Homo Ludens"). To little surprise then, people have also used sound in play and to create games. Since the advent of the personal computer, the genre of *audio games* invites sighted and visually impaired people alike to play interactive computer games solely based on sound renderings. While audio games are popular, especially among blind people, there is a lack of development tools to support audio game design and to foster further growth of this genre. For this reason, we demonstrate a browser-based audio game editor that we have developed over the last year or so, drawing on the experience and needs of seven long-term audio gamers. To the best of our knowledge, it is the first application or tool of its kind. Its key features are *easy usage* (including instant game play and sound rendering) and *open source* development to increase sustainability and possible impact.

AM'19, September 18–20, 2019, Nottingham, United Kingdom

ACM ISBN 978-1-4503-7297-8/19/09.
https://doi.org/10.1145/3356590.3356636

[1]The *Scary Shadow Syndrome* [1], for example, describes how the projection (a shadow) of a 'monster' can be a more scary experience than when directly seeing the monster (if the projection is a monster at all).

[2]https://audiogames.net/list-games/.

[3]http://www.blastbay.com/bgt.php.

CCS CONCEPTS

• **Applied computing** → **Computer games**; • **Software and its engineering** → **Interactive games**; **Designing software**.

KEYWORDS

Audio Games, Audio Game Editor, Audio Game Engine, Web-based, Sound Library

ACM Reference Format:
Michael Urbanek, Michael Habiger, and Florian Güldenpfennig. 2019. Creating Audio Games Online with a Browser-Based Editor. In *Audio Mostly (AM'19), September 18–20, 2019, Nottingham, United Kingdom.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3356590.3356636

INTRODUCTION & RELATED WORK

Playing games that are composed of sound only have two appealing features: They can deliver a fascinating experience drawing on mental imagery[1], and they can also be played by people with impaired vision. While audio games are popular, their success cannot be compared to the *video* game industry (i.e., games primarily based on visuals). Indeed, commercial studios show little interest in audio games, and so most available titles were released by amateurs or (semi-)professionals as freeware [2] and propagated by the influential, non-commercial *audiogames.net* community[2].

One key challenge in audio game design – and of the project described and demonstrated in the present contribution – is to *advance professional tools for game development*. Given the lack of resources and money in audio games, only few audio game-specific development tools are available. The most influential one, the Blastbay Game Toolkit[3] (BGT), is a game engine that has led to the publication of several audio games. It is favored by the audio game community today, even though its support has ended and the software has become outdated. This points at the pressing need for developing or finding up-to-date solutions. For *video* game development, there are several 3D game engines on the market that are used for commercial, research or private projects. Game engines like Unity3D or Unreal provide designers and developers with a plethora of possibilities in designing video games. Technically, these engines could be used for creating audio games too, providing audio game designers and developers powerful tools for creating non-visual experiences. However, most game engines are neither accessible nor tailored specifically to the needs of audio game designers.

In academia and in the scientific literature, only in recent years, researchers have called attention to this situation and proposed first (work-in-progress) tools and development concepts in order to support the process of designing audio games (e.g., [3, 4]). We connect to this work by proposing an (online) audio game editor, which we developed based on the feedback of seven experienced audio gamers and based on the inspirations we drew from toolkits such as Unity3D.

Scenario of use: Margit wants to create a classic Frogger game, i.e., the player has to safely cross a street and avoid cars. She opens the audio game editor and drags the player onto the GameArea. Next, she defines the safe endpoint on the other side of the street, also via drag and drop. In a similar way, she defines 'enemies' or cars. Using the menu of the right column, she defines the pathways of these cars. Now, it's time for a test! She puts on her headphones and clicks the play button. She closes her eyes and uses her keyboard for navigation. She safely reaches the other side and decides to add more cars to the GameArea.

"Audiogames, as opposed to video games are computer games who's [sic] main output is sound rather than graphics.

Using sound, games can have dimensions of atmosphere, and possibilities for gameplay that don't exist with visuals alone, as well as providing games far more accessible to people with all levels of sight."

– audiogames.net

[4]Every action that can be executed with the visual interface can also be executed via the API.

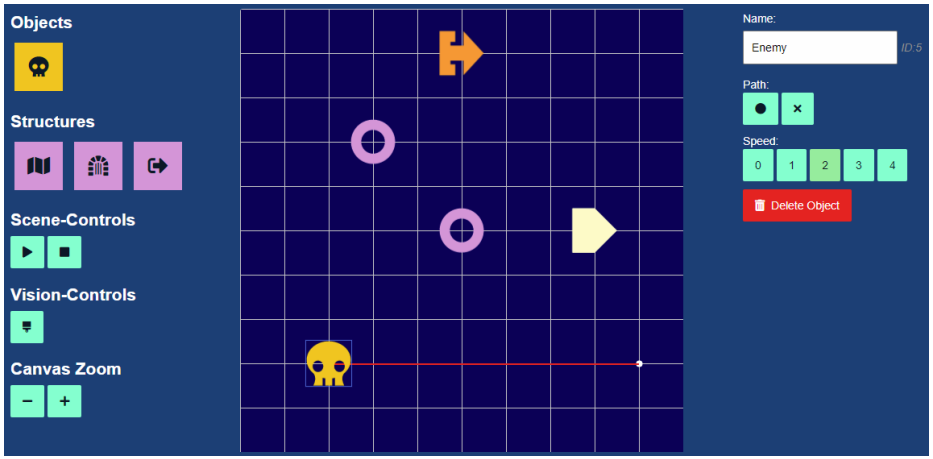

Figure 1: The graphical user interface (GUI) of the (online) audio game editor. Different template game-objects can be dragged from the left column onto the scene/*GameArea* (middle) by the users.

AUDIO GAME EDITOR - CONCEPT, INTERFACE, AND IMPLEMENTATION

Concept. The guiding principle of our (online) audio game editor is allowing the users to create games by combining different sounds and play logic, in a fashion as easy as possible. To facilitate this process, the users can *drag & drop* these game elements onto a 2-dimensional representation of the *GameArea* (see Figure 1 and below section for details). Most notably, this setup allows the users or designers to instantly playtest the game using headphones, as the interactive soundscape can be rendered in the same environment immediately (instant sonification). When the game is completed and ready for publication, the game can be shared and played online in the same environment (Note: this latter sharing feature is not fully implemented yet and will not be the focus of an onsite live demonstration.). The intended target group of this editor are people interested in audio games with or without visual impairments. In order to support individuals with low vision, the contrast of the application can be enhanced. Moreover, it comes with an *Application Programming Interface*[4] (API) that supports text-based (conventional) coding as an alternative.

User Interface and Accessibility. Our goal, to make audio games and audio game design more accessible, is reflected by the design decisions about the user interface in multiple ways. For one thing, audio games are less attractive to *sighted* players compared to people with visual impairments. The reason

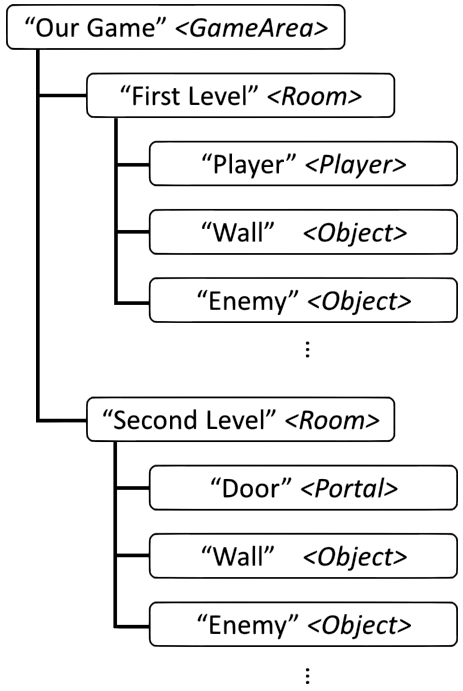

Figure 2: A typical scene structure of an audio game created by means of our editor. *GameArea* is located on the top level and holds several rooms. Each room in turn has several objects (e.g., the player, enemies, multiple bricks composing a wall) that belong to the respective room.

[5]https://github.com/resonance-audio/resonance-audio-web-sdk.

[6]In this version of the editor, we support the creation of adventure, puzzle or dungeon types of audio games.

for this – as simple as it might seem – is the absence of visuals that makes audio games less interesting for them compared to conventional video games. Thus, by providing visual elements and controls for designing and playing audio games as well as a textual API, we seek to create a tool that is appealing to broader audiences. Moreover, we want to provide audio game designers with little to no programming skills a low-barrier entry into the creation of audio games and empower them to make their own games. To this end, we designed the GUI to be composed of only three parts (see Figure 1). From its left column, the audio game designers can choose several blueprints that they can drag onto the *GameArea* in the middle part of the GUI. Nonetheless, they are not limited to these blueprints but can create their own objects as well. When the designers click on an object in the *GameArea* (Figure 1, middle), the selected object's respective properties are shown on the right in the properties screen, which allows easy modification of states. For instant playtesting, they can start and stop the game using the scene-control buttons (Figure 1, left). When the designer stops the execution of the game, the original game state will be restored and the designers can continue their work. Per default, the game is controlled with a conventional keyboard.

Finally, from a technological perspective, we will make the source code accessible and available by publishing it under an *open source* license in order to increase the sustainability of the project. We go on to provide some details about its technical implementation.

Implementation. From a technical perspective, the central organizing concept or frame of each audio game created with our editor is the *GameArea* (see Figure 2). Here the audio game 'takes place'. In analogy to a scene graph, it can be seen as the root of the current scene, which is regularly called by our underlying *game engine* for updates. The *GameArea*, again, can have multiple *Rooms*, which on the one hand can be seen as an analogy to 'levels' in games, and on the other hand are tailored to the room concept of Google's Resonance[5], which places an audio scene into a room and then renders it (see also below). The *GameArea* triggers the update game state function of each room. *Rooms* may have several game *Object*s, who get triggered by their respective room to update their state.

Every game *Object* of the audio game engine has at least three properties: a position in 3D space, a 3D rotation, and a 3D size. Further properties are a game object's *name*, a unique *ID*, a *tag*, a *type*, etc. (note Figure 2). Typically, such objects are enemies, friendly characters or walls.[6] Almost all classes inherit from game *Object* so that the basic aforementioned functionality is available to all such instances.

We decided to use several existing technologies to implement our engine. First, as mentioned before, we wanted to make our engine *open source* so that everyone can benefit from using it, without limitations (hence, we did not use commercial libraries etc.). Second, we wanted to make both the editor and the resulting games platform independent. Third, since we wanted to provide a highly reactive environment for creating, designing, and playing audio games, we decided to use a client-side

solution. The combination of these three requirements led to a JavaScript (JS) browser solution, since JS still is the most common client-side browser language used in web.[7]

With these deliberate constraints in mind, we were evaluating existing audio libraries that can be used for web applications so that we can easily utilize audio for our engine. Due to the needs of Virtual Reality (VR) applications, audio libraries for web recently received some attention. After some research, we opted for Google's Resonance (former Songbird) due to its support for web. The advantages of Resonance are built-in means for sound spatialization and reverberation based on pre-defined room sizes.

Existing JS game engines were either tailored to graphics or have a strong focus on physics. As these concepts are not or less[8] important in audio games, we wrote our own audio game engine from scratch, but used existing JS classes when appropriate (e.g., *Vector3* of three.js[9]). As JS standard, we used ECMAScript 6 (ES6) and Flow[10] as static type checker for our development. Our game engine incorporates a traditional game loop, based on the browser's built-in `requestAnimationFrame()`, that handles user input, updates the game state and renders the current scene.

CONCLUSION AND FUTURE WORK

We implemented and described an online editor for creating audio games. To the best of our knowledge, there is no similar application available. The project is timely, because the audio game genre is currently lacking advanced tools for development. As next steps, we will finish the implementation of advanced features and then release it to the public. As we are researchers in interaction design, we further plan to investigate the use of the online editor by the designers/users. In doing so, we hope to learn more about the audio game *design process*, i.e., how people go about to create games. In contrast to video games, there is very little research available about the audio game design process *per se*. In parallel, we will release the source code under an *open source* license to support the sustainability of our project and to enable fellow developers to re-use (parts of) our code.

REFERENCES

[1] Mats Liljedahl, Nigel Papworth, and Stefan Lindberg. 2007. Beowulf: An Audio Mostly Game. In *Proceedings of the International Conference on Advances in Computer Entertainment Technology (ACE '07)*. ACM, New York, NY, USA, 200–203.

[2] Niklas Röber and Maic Masuch. 2005. Leaving the Screen: New Perspectives in Audio-only Gaming. In *Proceedings of the International Conference on Auditory Display (ICAD2005)*. 92–98.

[3] Michael Urbanek and Florian Güldenpfennig. 2017. Tangible Audio Game Development Kit: Prototyping Audio Games with a Tangible Editor. In *Proceedings of the Eleventh International Conference on Tangible, Embedded, and Embodied Interaction (TEI '17)*. ACM, New York, NY, USA, 473–479.

[4] Michael Urbanek, Florian Güldenpfennig, and Manuel T. Schrempf. 2018. Building a Community of Audio Game Designers - Towards an Online Audio Game Editor. In *Proceedings of the 2018 ACM Conference Companion Publication on Designing Interactive Systems (DIS '18 Companion)*. ACM, New York, NY, USA, 171–175.

[7] Furthermore, this editor is planned to be part of an online community platform where audio games can be created, shared, and discussed. Hence, a browser-based solution was our preferred choice.

[8] Laws of Physics may play an important role in audio games as well, however, the physically correct movements of objects often may be more coarsely approximated compared to video games.

[9] https://threejs.org/.

[10] https://flow.org/.