

Akustik Simulation im Browser

Raummodenberechnung mit Hilfe der Finiten Elemente Methode für Nicht-Experten zugänglich machen

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Medieninformatik und Visual Computing

eingereicht von

Mag. Andreas Melcher

Matrikelnummer 00004876

an der Fakultät für Informatik
der Technischen Universität Wien

Betreuung: Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Mitwirkung: Assistant Prof. Dipl.-Ing. Dr.techn. Florian Toth

Wien, 27. Juni 2024

Andreas Melcher

Eduard Gröller

Acoustics Simulation in the Browser

Making Roommode Calculation with the Finite Element Method Accessible for Non-Experts

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Media Informatics and Visual Computing

by

Mag. Andreas Melcher

Registration Number 00004876

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller

Assistance: Assistant Prof. Dipl.-Ing. Dr.techn. Florian Toth

Vienna, 27th June, 2024

Andreas Melcher

Eduard Gröller

Erklärung zur Verfassung der Arbeit

Mag. Andreas Melcher

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 27. Juni 2024

Andreas Melcher

Acknowledgements

A huge thank you goes to Katharina Krösl, who broke the deadlock in my months-long search for a supervisor and connected me with Eduard Gröller, who deserves an equally big thank you, because without him this work would have been lost in the noise of a somewhat overwhelming everyday business life.

I would also like to thank Florian Toth. On the one hand for his help with openCFS, on the other hand for his willingness to act as supervisor of this work if I had not found anyone in the faculty of informatics.

Kurzfassung

Räume, die für Kommunikation oder Unterhaltung genutzt werden, benötigen eine geeignete Raumakustik. Häufig ist die Akustiksimulation ein wesentlicher Bestandteil der Planung solcher Räume. Während es teure und/oder schwer zu bedienende Softwarepakete gibt, die eine breite Palette von wissenschaftlichen Anwendungen haben, gab es in der Vergangenheit keine preiswerten und einfach zu bedienenden Möglichkeiten.

Im Rahmen dieser Arbeit wurde ein Prototyp entwickelt, der den speziellen Anwendungsfall der Berechnung und Visualisierung von Raummoden, einer spezifischen unerwünschten Erscheinung in der Raumakustik, vereinfachen soll.

Sinnvolle Werte werden nach Möglichkeit automatisch und ohne Benutzereingabe angenommen. Ein hoher Automatisierungsgrad und eine gute Benutzerführung stehen dabei im Vordergrund.

Interviews mit Testnutzern haben gezeigt, dass sie nicht nur die Raummoden ohne Anleitung visualisieren können, sondern auch, dass sie bei der ersten Nutzung nur 1-2 Minuten benötigen, um ihr erstes Rechenergebnis zu erhalten.

Abstract

Rooms that are used for communication or entertainment need suitable room acoustics. Often acoustics simulation is a fundamental part of the design of such rooms. While expensive and/or hard to use software packages exist that have a wide range of scientific applications, cheap and easy to use possibilities have not been available in the past.

In the course of this work, a prototype was developed that aims to simplify the specific use case of calculating and visualizing room modes, a specific undesirable phenomenon in room acoustics.

If possible, meaningful values are automatically assumed without user input. High automation and good user guidance are the focus.

Interviews with test users have shown that not only do they manage to visualize room modes without instruction, but also that they only need 1-2 minutes during their first use to obtain their first calculation result.

Contents

Kurzfassung	ix
Abstract	xi
Contents	xiii
1 Introduction	1
1.1 Why calculate Acoustics?	1
1.2 Numerical Acoustics	2
1.3 What are Room Modes?	3
2 State of the Art	5
3 Description of Approach	11
3.1 The User Journey	12
3.2 Easy and fast 3D Modeling	15
3.3 Reacting to Invalid Input	18
3.4 Modern Browser Technologies to Use the GPU	22
3.5 FEM on the Server	23
4 Technical Implementation	27
4.1 3D Graphics in the Browser using Threejs	27
4.2 Distributed Architecture	30
4.3 FEM Processing - Communicating with GMSH and OpenCFS	32
4.4 Input and Output Files	34
Results	35
Conclusions	37
List of Figures	38
Bibliography	41

Introduction

1.1 Why calculate Acoustics?

Various situations in our daily lives demand good acoustics. What good means depends on the needs of the individuals involved and the purpose of the room in which they are in.

In a concert hall for classical music, for example, the audience wants to be enveloped [Kaa13, Ch.2.3.8]. To ensure this, the room needs a certain amount of reverberation [LKV09], [FCP08], [Bar88], [AO09], [HN22], [ALTG10], [ALD11].

In a classroom, on the other hand, speech intelligibility is paramount and long reverberations would be counterproductive [SWLL05], [RBH12]. The induced sound must be distributed so that it is neither too loud nor too quiet anywhere. Additionally, no disturbing echoes should occur.

The task is to design spaces so that they acoustically function for their intended purpose. Standards propose or prescribe these acoustic properties for various uses. Acousticians, studio operators, architects, music enthusiasts—many individuals need to better understand the acoustic behavior of specific spaces and adapt them to well-known standards. Those who work professionally in this field invest time and money in existing acoustics simulation software to meet these needs.

1.2 Numerical Acoustics

Various calculation methods utilize digital models of rooms to simulate the physics of sound in different ways. Similarly to Physically Based Rendering [PJH16] in computer graphics, one uses knowledge about the radiation behavior of sources, interaction with objects, and human perception to digitally depict what happens in reality. Typically, there is the option to invest more computation time to obtain a more accurate solution. Knowledge about human perception is crucial because what pleases or bothers us in reality is particularly relevant for optimizing the algorithms. Innovations in this field are often achieved through new algorithms that achieve the same realism with less computation time. However, there is a crucial difference in acoustics compared to optics. The human eye perceives light waves with a wavelength of 380 nm to 780 nm. The ear functions much more "broadband". Wavelengths between 2 cm and 20 m contribute to the auditory experience. Even longer waves are not heard but can be perceived as vibrations. In optics, the lowest and highest visible frequencies are approximately a factor of 2 apart (an "octave"). In acoustics, they are approximately a factor of 1000 apart (10 octaves). One of the fundamental challenges in acoustics simulation is the necessity to consider a wide range of frequencies in the calculation. This is particularly crucial because waves behave differently if their wavelength is significantly smaller, approximately the same, or significantly larger than the dimensions of the obstacle. Unlike in optics, where most objects and structures are much larger than the wavelength of visible light, nearly all surface structures in our rooms range from 2 cm to 20 m, i.e., the wavelengths of acoustics waves in our audible frequency band, as stated above. This underscores the importance of considering wave phenomena such as refraction, diffraction, and scattering in acoustics simulations.

1.3 What are Room Modes?

Almost always, one wants to encounter "neutral" acoustics. For example, music or spoken words should have enough clarity and loudness throughout the room. This work focuses on software simplifying the calculation of room modes, an acoustic phenomenon that jeopardizes this "neutrality".

Room modes are resonance phenomena within the volume of a room. A sound source creates small air pressure fluctuations in its immediate surroundings. This local overpressure or underpressure equalizes in all directions. The pressure fluctuation propagates at the speed of sound, approximately 343 m/s [TA]. In the open air, the vibration diminishes with increasing distance from the sound source. However, in enclosed spaces, this sound wave inevitably encounters boundaries and is reflected often before fading away. At the same time, the sound source continues to stimulate the room with new sound energy.

Vibrating systems always have resonances. An example everyone is familiar with is a simple swing. There is the so-called "resonant frequency", at which it is very easy to maintain one's own or someone else's swinging motion.

Imagine trying to push a child with a much higher frequency on the swing. One can only push it at specific times because you must wait for the swing to return. If you really want to move it faster, you would have to take the swing and the child and move them back and forth with muscle power. The system is not swinging on its own in this case; it is actively moved. The same applies if you want to swing much slower. For that, too, you would have to constantly exert force to slow down the swing's natural (faster) motion.

At the resonant frequency, the swing's moving system reacts much more sensitively to energy input. A brief push is enough, and depending on friction, the system partially swings for a long time.

The frequency of the resonance does not depend on the excitation, such as the speed of pushing, but only on the system's boundary conditions. In the case of the swing, these are the length of the ropes and the weight of the person.

The vibrating air volume in rooms operates similarly. Sound energy is usually radiated "broadband", meaning at many frequencies simultaneously, into the room. However, the sound energy in the room remains longer at resonance frequencies than between them, letting us perceive these tones longer and louder than other tones, as sound energy accumulates.

And that is precisely the problem of room modes. When listening to the solo of a bassist, a listener wants to hear the individual notes as short and loud as the bassist intended. Room modes, however, cause specific notes to be louder and longer while other notes are barely perceptible. Additionally, room modes are a local phenomenon. Different positions in the room have different notes that are too loud or not audible. Depending on where you are in the listening space, the solo can sound differently, which is typically undesirable for all involved.

State of the Art

There are countless possibilities to calculate room modes. The following diagram roughly clusters the market by price and difficulty to calculate and visualize room modes.

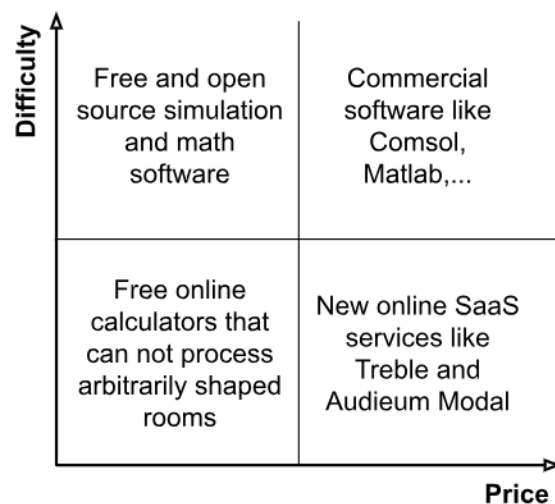


Figure 2.1: A rough categorization of existing software to simulate room modes

There is software like Matlab [mat] or Comsol Multiphysics [com], together with many add-ons that engineers and universities use to simulate physics. As these software packages can do much more than calculate room modes, their usability is not optimized for this use case. The price is in the range of thousands of euros per license. So, all of those applications are in the upper/right corner of Diagram 2.1. If a company or institution needs to simulate much more than room modes, those software packages can be the right choice. Once users have climbed the steep learning curve, they can calculate pretty much anything humanity currently knows.

2. STATE OF THE ART

There are free alternatives to those commercial products, but this does not help with the difficulty of using them for our particular use case. Once one can solve differential equations, the software can theoretically be used to do the math, and once you can visualize the numbers as 2D and 3D plots, one can even visualize the resulting room modes. Of course, different open-source modules for different software packages can be used to ease the simulation. The many possibilities of combining different free software to build something new do not necessarily help to flatten the learning curve.

For those with simpler needs, there are free and user-friendly online tools available, e.g. [rmca][rmcb][rmcc][rmcd]. These tools excel at calculating and visualizing room modes, albeit are limited to rectangular rooms due to their use of an analytic method. While they may offer fewer features, they are advantageous for education and quick estimates. Their main advantage lies in their straightforward user interface, designed with a singular focus on providing an easy and effective solution for calculating and visualizing room modes.

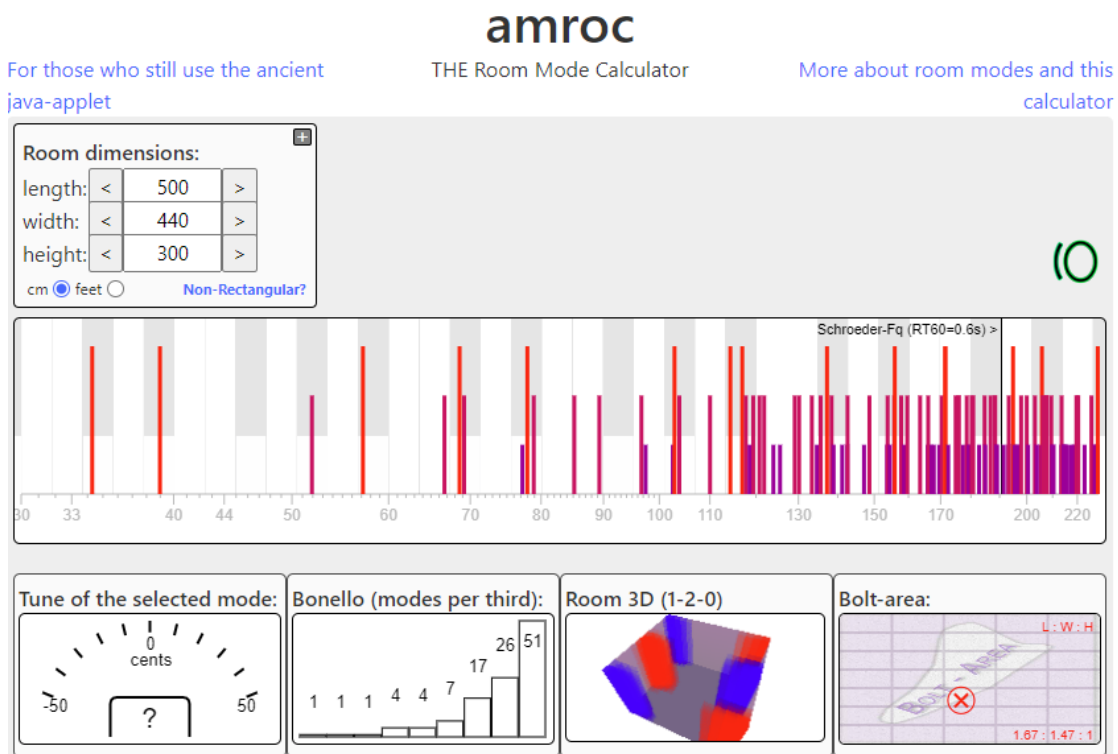


Figure 2.2: Free online room mode calculator, that can only calculate room modes for rectangular rooms [amra]

When the topic of this thesis was agreed on, the bottom right quadrant of the interface in Figure 2.1 was empty. There was a gap between the free online tools that could not handle non-rectangular rooms and the available math software that was difficult to use

for such a specific use case. In the last year, two new online tools emerged: Treble [tre] and Audieum Modal [aud].

Tremble is a software-as-a-service (SaaS) solution that simulates acoustics online and takes modal behavior into account. It is a big undertaking with a big team, and it looks like they have stepped up to replace any other room acoustics simulation software [tre]. While they are making the gap between ease of use and feature-richness smaller in room mode calculation, it is not directly focusing on room mode analysis.

Audieum Modal also is a SaaS product and completely focuses on analyzing room modes [aud]. In terms of its target and features it is the closest one to the prototype presented in this work.

The main differences as of April 2024 are:

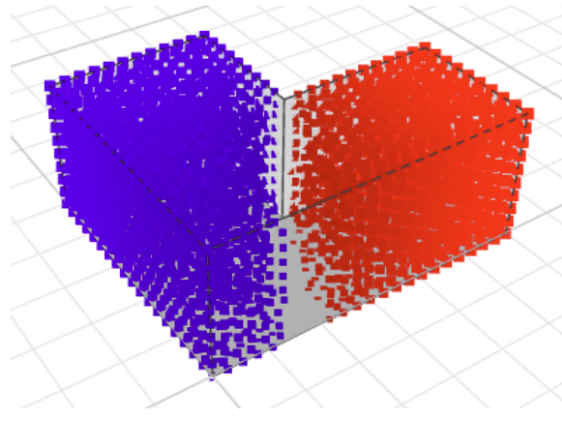
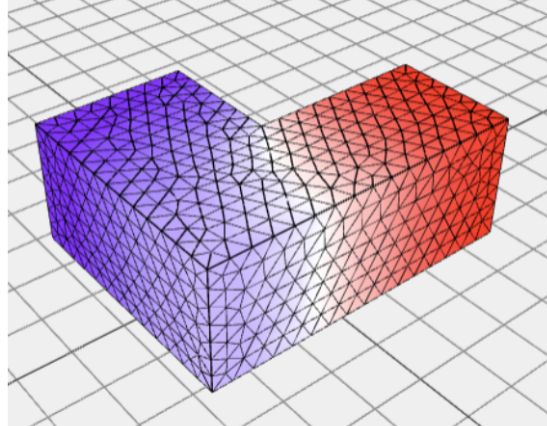
The prototype presented in this work	Audieum Modal
can handle room volumes of arbitrary size	can only handle room volumes up to some limit
automatically chooses a sane limit frequency, no range limit	has a maximum frequency range of 20-140Hz
supports modeling in its own editor but is not capable of every room shape yet	offers room shape presets and supports potentially arbitrary room shapes by uploading SketchUp files. But they need to review each SketchUp file manually, so processing is delayed by the time they need (2 weeks in a case of the author).
	

Table 2.1: Comparison of the prototype presented in this work and the new online room mode calculator Audieum Modal

The prototype can handle rooms of arbitrary volume and automatically chooses a sane upper limit frequency, whether the volume is very small or big. This is done by calculating

an appropriate grid resolution when automatically meshing the model. Audieum Modal seems to use a fixed grid resolution. At least, this would explain the limits for maximum frequency and volume.

For room modeling, the prototype described here comes with its own editor. Audieum Modal allows users to select one of five fixed room shapes (see Figure 2.3).

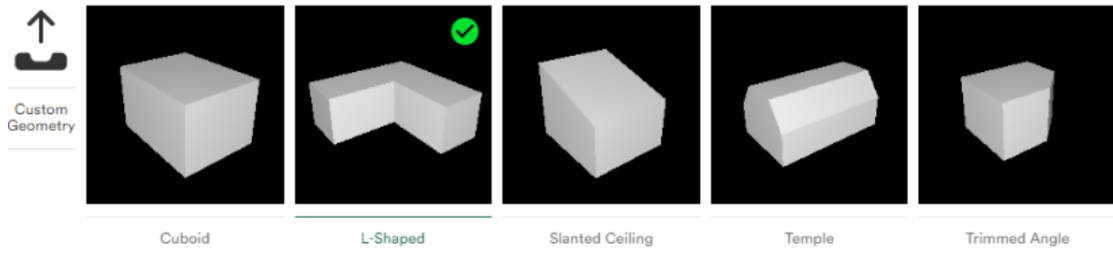


Figure 2.3: Audieum Modal room shape presets

When using one of the given room shapes, the user can enter dimensions. In Figure 2.4 an example of an L-shaped room is shown.

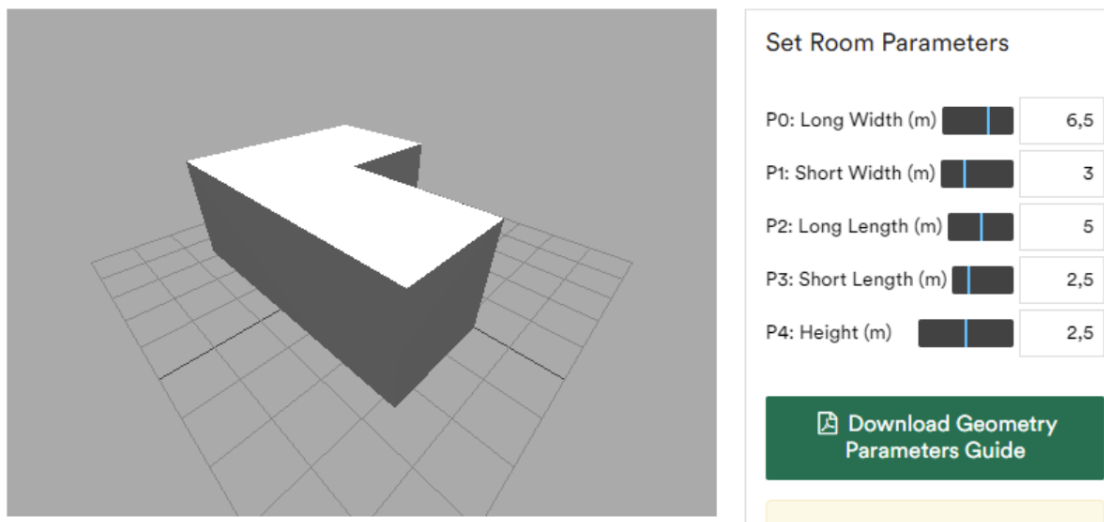


Figure 2.4: Audieum Modal settings for an L-shaped room

In addition to these five presets a user can upload a custom geometry in the form of a SketchUp file. Unfortunately they need to check the uploaded model manually before the processing can start.

The results from Audieum Modal and our prototype are similar. Both are presented as visualized pressure distribution in the 3D model of the room. While our prototype visualizes the mode pressure field with a cloud of sprites, Audieum Modal visualizes the pressure on the room's boundaries.

From a user perspective, using our prototype is faster for a room shape without a slanted ceiling or wall. If the analyzed room differs from this constraint, Audieum Modal offers the same features, but one has to use another software (SketchUp) and wait for a manual check of the model by the Audieum Modal Team, before they get the results. They state that it can take up to five work days. In our case, they needed two weeks.

Description of Approach

As described in Chapter 2, until recently, considerable time and/or money had to be invested to calculate room modes in arbitrarily shaped rooms. The approach presented here aims to simplify the calculation and visualization of room modes in non-rectangular rooms for the end user.

In broad terms, there are two focal points to simplify room mode calculations significantly. The first point is the ease of providing the software through a website. Modern browsers make it possible to provide the necessary functionality without the need to download and install extra software, keeping the entry effort low (Section 3.1).

The second point is a strong focus on usability. To keep the approach simple, features are only integrated once a quick and easily understandable approach has been found. Some parameters are set automatically to hide the complexity from the user. Letting experienced users decide for themselves is planned, but will be decoupled as an expert mode to prevent clogging up the user interface. Presenting the results as interactive 3D visualization, makes the result intuitively understandable (Sections 3.2, 3.3 and 3.4).

The complex calculations are outsourced to the server. This fundamentally keeps all options open regarding physics computing. Various platforms can be utilized for calculations, and scalability up to operation on server clusters is possible. Depending on the parameters (model size, resolution,..) FEM calculations can take a lot of time. Splitting the user interface in the browser from the calculation system on servers is a very good way to relieve the computer of the user and instead notify him/her asynchronously about the result when it is done (Section 3.5).

3.1 The User Journey

A possible user journey could start in one of the acoustics forums on the web. A user could want to decide which one of two possible rooms in a house she/he should use as a home cinema and asks for additional opinions. Other forum visitors advise to measure both rooms or get a fast evaluation of their room mode distribution. They post a link to the tool described here. The user clicks the link and can immediately work with the tool.

The approach for easy modeling is described in Section 3.2. After modeling, the screen looks like in Figure 3.1. Note that walls are transparent if viewed from the outside to let the user see the inside of the room.

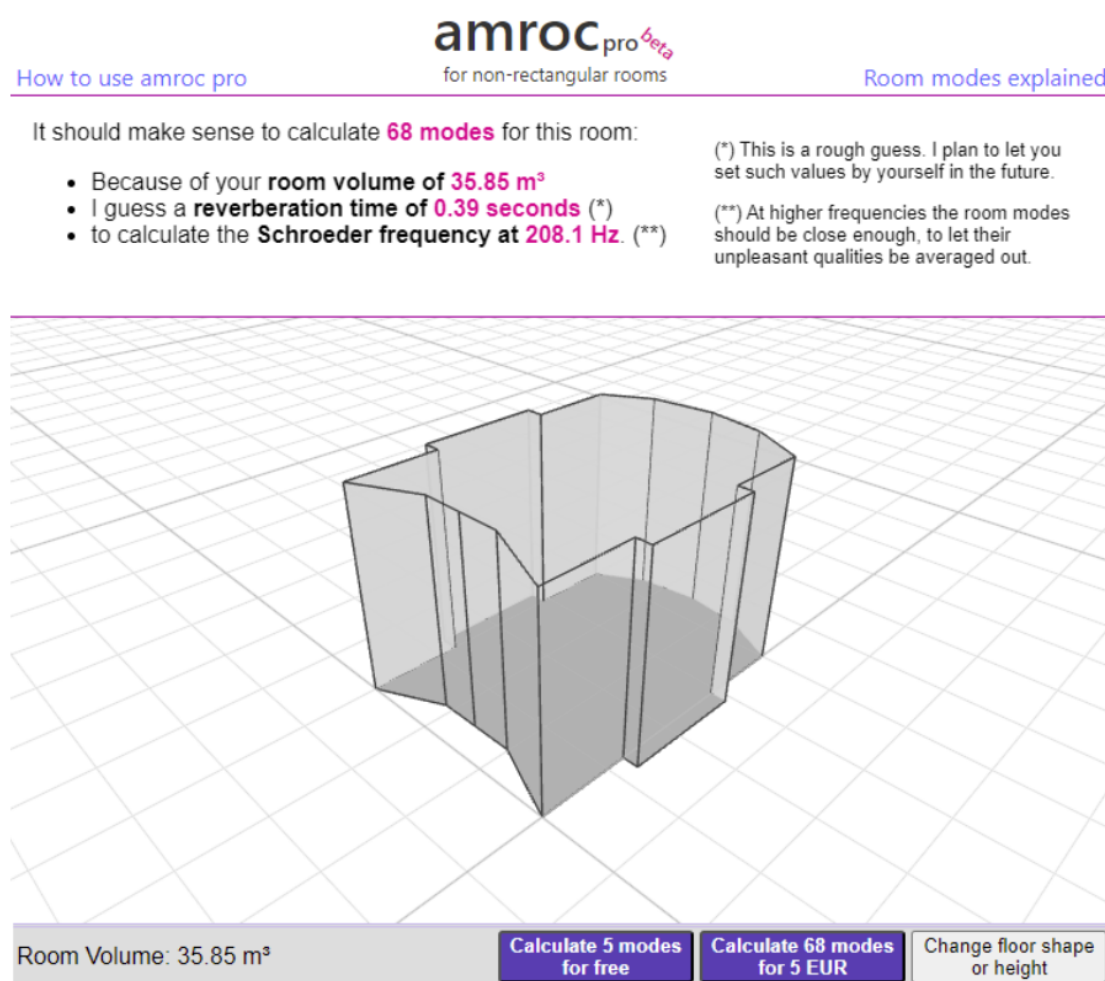


Figure 3.1: Displaying the 3D model to the user together with some values about the room before doing the acoustics calculation

After clicking one of the mode calculation buttons seen in Figure 3.1, the server does the simulation. Status messages are pushed from the server to the user to inform about

important steps in the process: "Splitting your room's volume into small pieces...", "Calculating room modes..." and "Preparing your results for download...".

When the calculation is finished, the panel above the model changes. A frequency diagram with vertical lines appears above the 3D model instead of the pre-calculation information (see Figure 3.2).

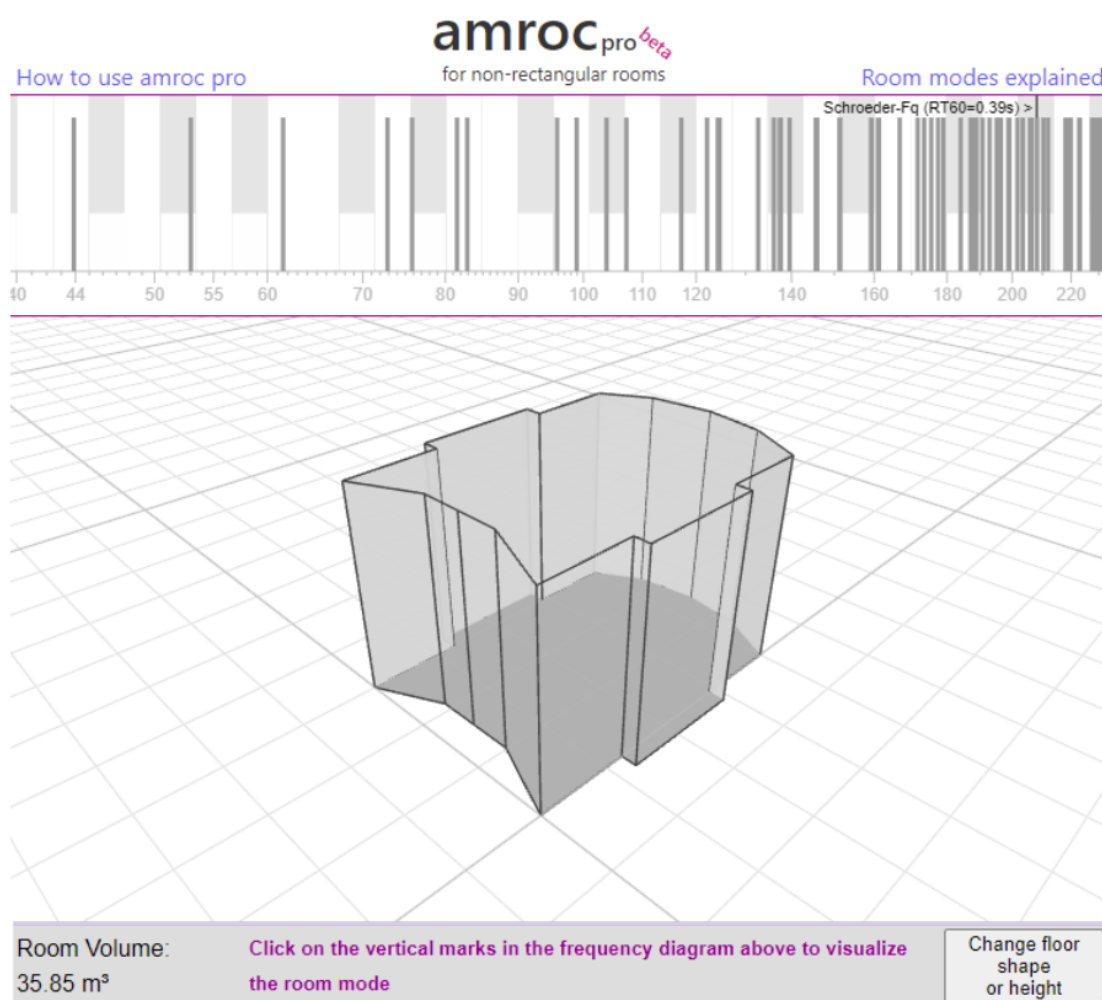


Figure 3.2: Showing a frequency diagram over the 3D model with the selectable mode frequencies as vertical lines

The status message gives a hint about what to do next. In the background the user also gets an email with a link to be able to come back if he closed the browser in between.

With just a few clicks and the help of some status messages, the user now has an overview of the modes in the room and can click the marking line at the mode's frequency to see the pressure zones of each specific mode (see Figure 3.3).

3. DESCRIPTION OF APPROACH

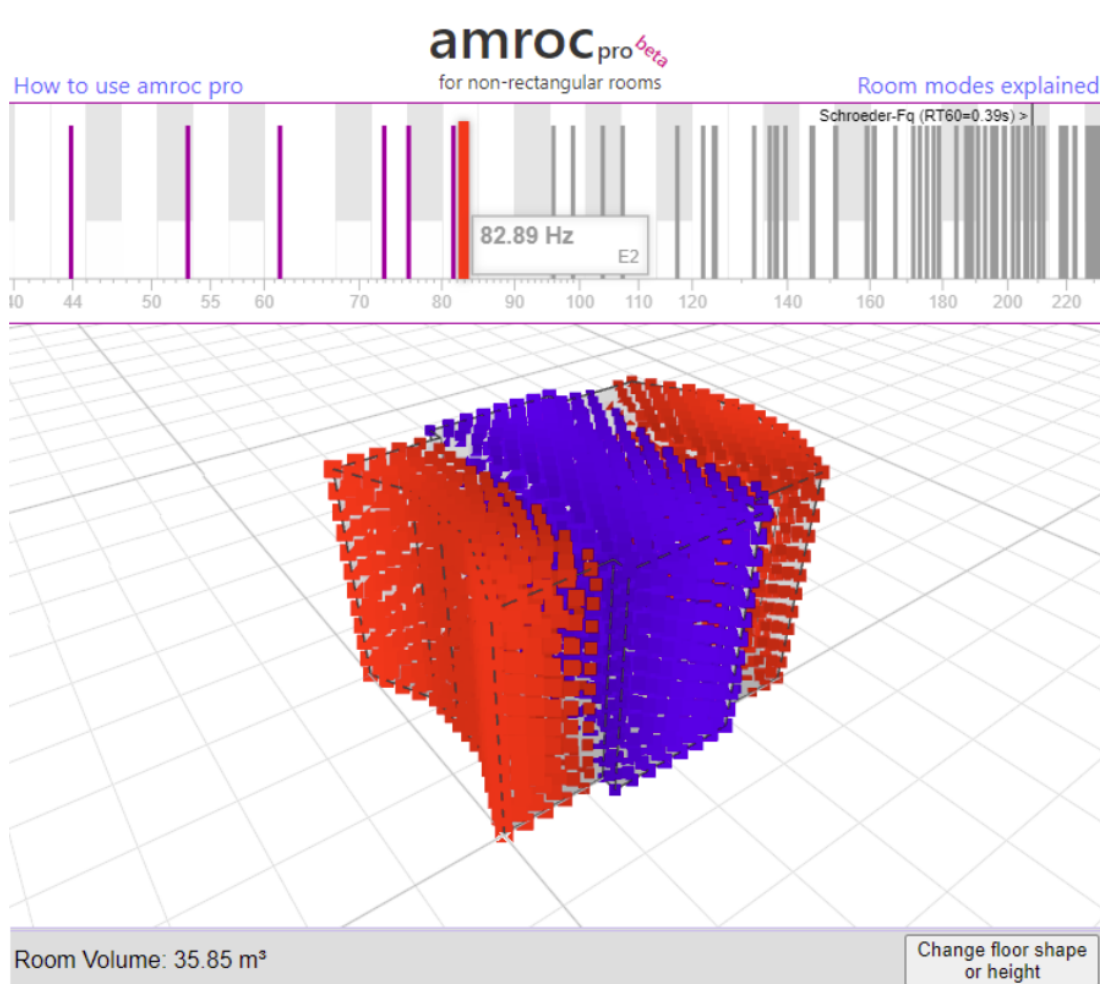


Figure 3.3: Visualizing the pattern of the sound pressure of one selected mode via a cloud of sprites. Red and blue both mean pressure (loud). In between the red and blue zones the mode frequency is barely audible. This pattern is known to the users from the free version for rectangular rooms and described in the article about the calculator and the theory of room modes on the website [roo].

3.2 Easy and fast 3D Modeling

While software developers strive to create highly usable features, the reality is that software development is a complex process that necessitates trade-offs. Usability is not without its costs (Figure 3.4). One needs time to plan, test, and ask for feedback and to iterate and improve.

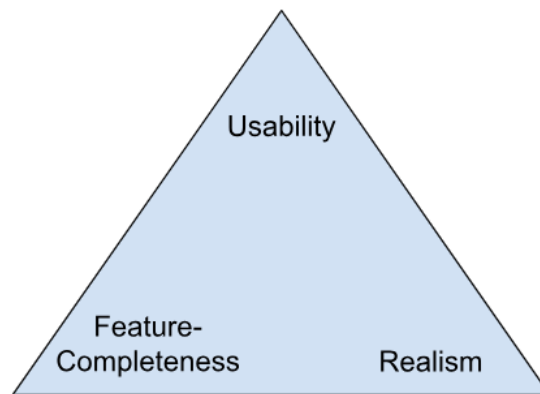


Figure 3.4: Assuming the same development time, deciding to focus on usability means deciding against some other step in the software development, like feature-completeness or realism.

It makes sense to actively decide for a priority of software qualities and the focus of the tool described here is on usability. This means, if a very important feature can not be built in an easily understandable way, other features get a higher priority.

A lot of rooms have a flat floor and a flat ceiling. This makes it possible to only model the floor (in 2D) and extrude the whole 3D volume. Nevertheless, a poll shows that the most often requested features are related to extending modeling to non-flat floors and ceilings (see Figure 3.10).

Modeling in 2D is basically drawing. It simplifies the creation of the model a lot. When the website loads the user can immediately start to model/draw on the 2D canvas (Figure 3.5).



Figure 3.5: The 2D canvas shown to the user when the app loads.

The colored boxes in the corner invite the user to explore. The short explanation text intentionally uses the word “floor”. Together with the two only other ways of interaction (the text input for the room height and the 3D button), the user interface reduces the process of modeling a 3D room to three steps:

- Shape your floor
- Select a height
- Extrude the model (\Rightarrow 3D)

Keeping the interface clean is one part, intuitive interaction on the drawing canvas is another vital usability detail.

If the user moves the mouse over one of the colored corner boxes, the color of the specific box changes to invite the user to grab and move the box and, therefore, the corresponding corner. Some additional labels appear to show the angles and lengths of the walls involved in the movement (Figure 3.6).

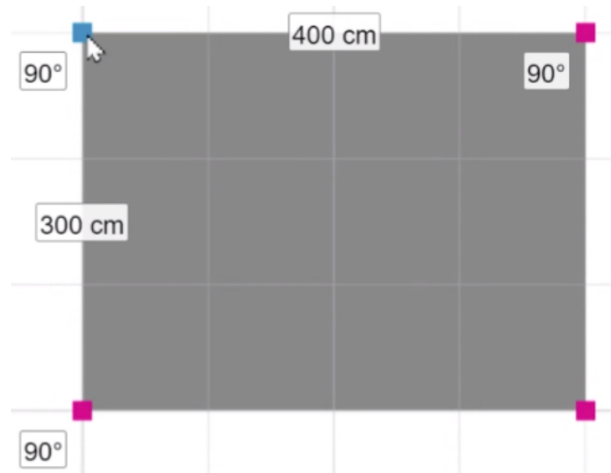


Figure 3.6: A color change of the box indicates the interactivity of the handle

If the user's mouse is close to the model (the gray area), a new box appears, showing the user that new corners can be created and moved wherever he/she clicks and grabs (Figure 3.7 and 3.8).

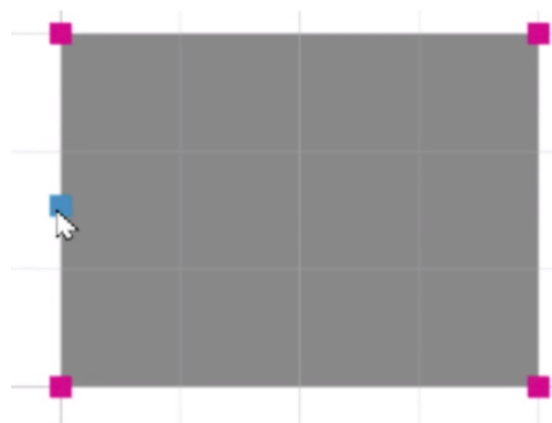


Figure 3.7: Creation of new corners

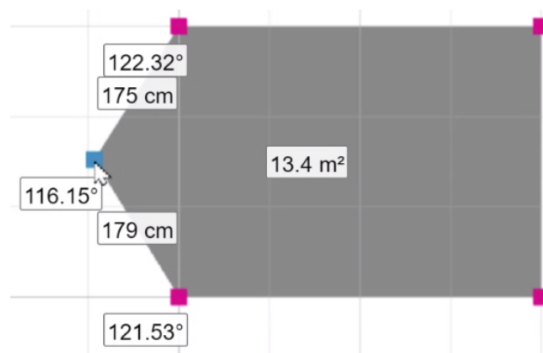


Figure 3.8: Shaping the floor

The user can click the =>3D button at any time. This button is disabled only if the model is not valid for the acoustic simulation.

3.3 Reacting to Invalid Input

For the finite element method the model needs to have a closed surface without intersections (closed manifold) [SE02]. In the current state of the app the floor can not be open as it is not possible to delete all points. At least three points and the connecting walls are always present, ensuring that the extruded volume is always closed.

The second invalid property (intersection) is possible in the UI but the moment the user moves the handles in a way where the walls intersect, the drawing becomes red to give fast feedback (Figure 3.9).

Providing fast feedback is essential. Faulty models are a common problem when simulating acoustics, and it is harder to find a problem if it has existed silently for a long time.

Usually, existing CAD models are reused for acoustic simulation. The manual pre-production steps to make the models valid for the chosen acoustic simulation engine are one big cause of pain in the more feature-rich approach to using CAD and FEM software directly.

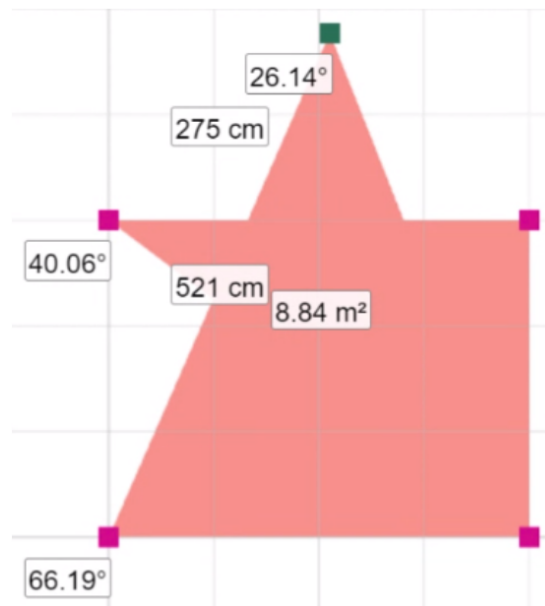


Figure 3.9: Invalid floor shapes immediately turn red to provide fast feedback.

3.3.1 The problem with loading 3D models created in other software

Adding the possibility of loading existing models is a feature that is hard to add in an easy-to-use way. As explained in Section 3.3, the tool needs one closed volume to calculate resonances for. Typical architectural models include many volumes (objects) of different sizes. A step is needed where the user can select the one specific volume the resonances shall be calculated for.

In addition to that, 3D models are often optimized for visual rendering. They neither need to have one closed surface nor are intersections a big problem. Such models do not need to be watertight, therefore the tools they are created with do not help much with this requirement. Surface intersections and holes are not the exception but very common in such models.

To allow importing arbitrary architectural models in an easy way, the acoustics simulation tool needs to help the user to adjust the model. There are several ways to implement this step, with more or less manual work done by the user. But this complexity moves it back on the feature priority list described above.

There is excellent 3D modeling software like SketchUp [ske] and Blender [ble] that already provide relatively easy user interfaces to model a large variety of shapes. One still has a learning curve. The average inexperienced user will be able to download and start such software in a couple of minutes. The first easy modeling steps will be possible quite fast. Other features will require expert user knowledge; weeks, months, or even years of experience will make the modeled work stand out.

The purpose of the tool presented in this work, is highly specific. It is designed to model only closed 3D volumes (rooms). Intricate room details that do not affect room modes are not needed, therefore structures smaller than a few centimeters do not need to be modeled.

3.3.2 The problem with modeling a room by extruding its 2D floor

The tool presented in this work is already online under the name amroc pro. As soon as it went online a poll was added to let users influence the prioritization of planned features (see Figure 3.10).

The two topmost feature requests require replacing or extending the way of modeling. Extruding a 2D shape is easy, but not every shape is possible. So, a future addition has to be other modeling techniques that allow for slanted ceilings and other details.

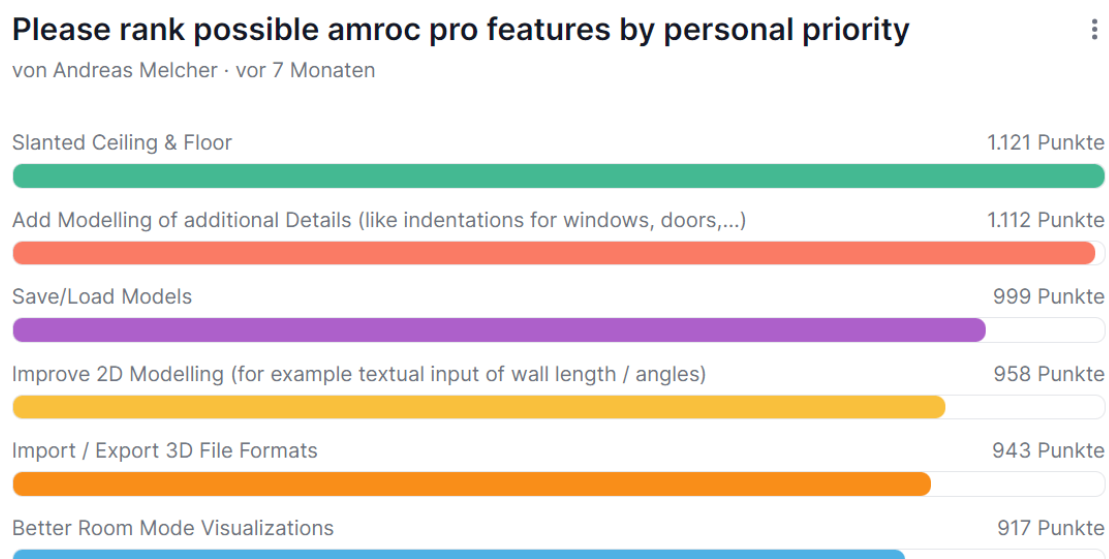


Figure 3.10: A survey among users about most wanted next features in amroc pro (the new online name of the application described in this work [amrb])

3.4 Modern Browser Technologies to Use the GPU

As mentioned earlier, a primary focus was to provide a tool that is easily accessible through a website. While developing software for browsers has its pros and cons, it offers the significant advantage of being accessible on all widely used operating systems without the need for installation. However, it is important to note that the restrictions to the technologies supported by browsers can pose a challenge. Fortunately, modern browsers have made significant strides in providing access to 3D graphics and the power of GPUs.

Different APIs allow the development of efficient interactive 3D visualizations by directly using the GPU in the browser. At the time of writing, WebGL has been available for around 10 years and WebGPU is starting to replace it.

WebGL is a cross-platform, royalty-free API used to create 3D graphics in a Web browser. Based on OpenGL ES 2.0, WebGL uses the OpenGL shading language, GLSL, and offers the familiarity of the standard OpenGL API [weba].

WebGPU is the successor to WebGL, providing better compatibility with modern GPUs, support for general-purpose GPU computations, faster operations, and access to more advanced GPU features [webb].

Several libraries and frameworks use those browser APIs and try to make the development of 3D content as easy as possible [lib].

3.5 FEM on the Server

3.5.1 Automatic Pre-Processing

As explained in Section 1.2, our ear can hear a wide range of frequencies. The Finite Element Method is suitable for low-frequency analyses. Unlike geometrical acoustic models (e.g., simple ray tracing), it captures the relevant physics, like the phase or diffraction. The problem with the FEM is that it becomes resource intensive pretty quickly if either the model or the analyzed frequency increase. As a few grid nodes per wavelength are necessary, the number of nodes grows with an increasing model size or decreasing wavelength. The scientific community is working hard on improved computational approaches, but until today, one needs to focus on a reasonable upper frequency and grid resolution.

"To model the auditorium, with a volume of 2271 m³, approximately 150 million degrees of freedom were required. They were able to model an impulse response of 3 s up to 3 kHz, using 512 CPU cores. Solution took about 2.5 h" [Pri23, Ch.6.1.2]

In order to respond to a calculation request quickly, the grid size and upper frequency must be chosen sensibly. At the same time, the tool described here is meant to be easily usable by non-experts, so it is important to choose reasonable values for those parameters automatically.

As the grid needs a minimal number of nodes per wavelength, choosing the grid resolution depends on knowing the maximum frequency that needs to be analyzed. Doubling that frequency multiplies the number of necessary nodes by 8 (2^3 as we double a grid in 3 dimensions). Keeping the frequency as low as possible while still finding all "relevant" room modes is therefore crucial for an efficient simulation.

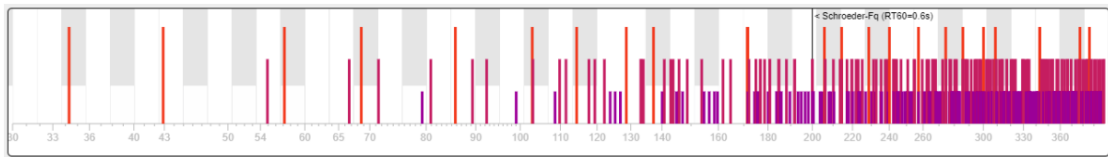
The number of modes per frequency band increases exponentially (Figure 3.11). Therefore the modes are so dense at higher frequencies that their adverse effects average out.

Manfred Robert Schroeder proposed an equation to calculate a crossover frequency via

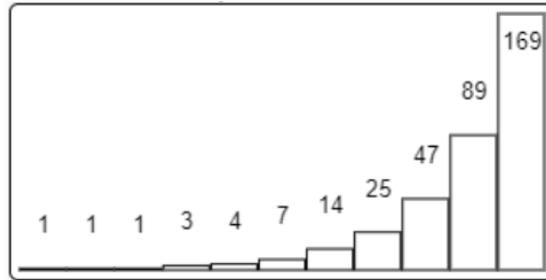
$$f_s = 2000\sqrt{T/V} \quad [\text{MM04, eq.11.3}] \quad (3.1)$$

Where f_s is the so-called 'Schröder frequency', T is the reverberation time of the room and V is the volume of the room. The volume can be derived from the model, but T depends on properties that are unknown at this point. Two equally sized rooms can have different reverberation times because of the different acoustical properties of their walls and furniture. Anechoic rooms exist that are built to have no reverberation at all, and reverberation chambers are built to keep sound energy as long as possible by having very reflective walls. Nevertheless, we are used to common reverberation times and can be baffled if rooms behave differently. To keep handling very simple in the beginning,

3. DESCRIPTION OF APPROACH



(a) Modes in a rectangular room of size 500 x 400 x 300 cm



(b) The number of modes per third rises with the frequency (same room dimensions as above)

Figure 3.11: Increasing mode density with frequency

the user is not asked for the reverberation time. Instead the time is estimated solely depending on the volume for now:

```
1 // f and c are chosen in a way that a volume of 30m^3 (small studio / dry
2 // livingroom) results in a reverberation time rt60 of ~0.3 seconds and a
3 // volume of 16000m3 results in ~2.2 seconds (Viennese Musikverein)
4 const f = 0.7;
5 const c = -0.7;
6 let rt60 = f * Math.log10(volume) + c;
```

Despite being real typescript source code, this and the following code blocks are meant to provide information about how to do this in any language, as it is pure math.

With the volume and the reverberation time, the upper frequency limit can now be calculated with the help of Eq. 3.1.

```
1 const minElementsPerLambda = 6;
2 const speedOfSound = 343; // m/s
3
4 // Taschenbuch der techn Akustik 3.Auflage, 2004, S.332, Formel 11.3
5 const schroederFq =
6     2000 * Math.sqrt(estimatedRt60Info.rt60 / estimatedRt60Info.volume);
7
8 // As my schroederFq is based on some rough estimations I use one musical
9 // third above as the upper limit
10 const upperLimitFq = schroederFq * FACTOR_ONE_THIRD;
```

The grid size can be derived by choosing a specific number of nodes per wavelength of that frequency.

```
1 // wavelength at the upper limit frequency
2 const lambdaUpperLimit = speedOfSound / upperLimitFq;
3
4 const maxDistanceBetweenNodes =
5     lambdaUpperLimit / (minElementsPerLambda - 1); // -1 as 3 nodes have
6                                                       // just 2 distances
7                                                       // between them
8                                                       // (x <-> x <-> x)
```

3.5.2 Automatic Post-Processing

In the case of this tool, server-side post-processing is just data transformation to speed up the data download. Pressure information for each node is saved as specific files. It can, therefore, be derived from this cache without further processing whenever a user wants to visualize the pressure zones of that room mode on the front end.

Technical Implementation

4.1 3D Graphics in the Browser using Threejs

It was clear that using the GPU would be necessary to build performant 3D visualizations. As stated above, modern browsers can use the GPU via specific APIs.

As an API change is currently being carried out slowly (WebGL -> WebGPU) one goal was to hide much of the inner workings of WebGL by using a library.

The web research revealed two leading contenders:

- Threejs [thr]
- Babylon.js [bab]

The decision has been made for threejs, mainly because of its smaller runtime size, and the author already had a bit of experience using it.

In threejs one initializes a renderer and a scene and adds objects to that scene. One can then retrieve a reference of the dom element to add it to the website's HTML code.

```
1  this.renderer = new WebGLRenderer({ antialias: true });  
2  this.renderer.setSize(divRenderer.clientWidth, divRenderer.clientHeight);  
3  divRenderer.append(this.renderer.domElement);
```

Calling `renderer.render(scene, camera)` updates the visualization. As seen above, the `WebGLRenderer` is used in the current implementation.

A scene is composed of Threejs Object3D objects [T3o]. These can be Lights [T3l] or Meshes. A Mesh consists of a Geometry [T3g] and a Material [T3m].

The following code block shows an example for building a threejs Mesh object.

If the `ExtrudeGeometry` is built with an array of materials, the first is used to render the two sides that were shaped and the second is used to render all other sides.

In the materials `side:Backside` is used to let the walls of the room only be visible if they are visible from inside. That way, the inside of the room is always visible and therefore we have clear view on the room modes.

```
1 private createExtrudedMeshFromFloorPointsAndHeight(  
2   floorPoints: readonly Vector3[], heightInMeters: number  
3 ) {  
4   const extrudeSettings = { depth: heightInMeters, bevelEnabled: false };  
5   const floorPoints2D = floorPoints.map(v3 => new Vector2(v3.x, v3.y));  
6   const shape = new Shape(floorPoints2D);  
7   const extrudeGeometry = new ExtrudeGeometry(shape, extrudeSettings);  
8  
9   // If the ExtrudeGeometry is built with an array of materials, the first  
10  // is used to render the two sides that were shaped and the second is  
11  // used to render all other surfaces  
12  // 'side: Backside' is used to only render surfaces if they are seen from  
13  // the inside of the room. That way, the interior (room modes) is not  
14  // hidden by the walls surfaces  
15  const materials = [  
16    // Floor / Ceiling  
17    new MeshStandardMaterial({ color: 0xA9A9A9, side: BackSide }),  
18    // Walls  
19    new MeshStandardMaterial({  
20      color: 0xCCCCCC, opacity: 0.95, transparent: true, side: BackSide  
21    })  
22  ]  
23  
24  const roomMesh = new Mesh(extrudeGeometry, materials);  
25  return roomMesh;  
26 }
```

In addition to the room model and the visualized mode results, the scene holds an `AmbientLight` for general illumination and a `SpotLight` to improve the distinctiveness of walls with its angle-dependent illumination.

A `PlaneGeometry` with a `ShaderMaterial` adds the infinite grid on the floor to improve the feeling of size and direction. The `ShaderMaterial` class is used when a custom shader is written [gri].

To render the calculation results, a `FemResultsSceneModifierService` listens for downloaded results and adds them to the scene. The result is a list of pressure values at specific coordinates in the room. For now, the `FemResultsSceneModifierService` simply adds a rectangular sprite at every coordinate and updates the size and color of each sprite depending on the pressure value (see Figure 4.1). Other volume visualization techniques like iso surfaces or clipping planes are planned.

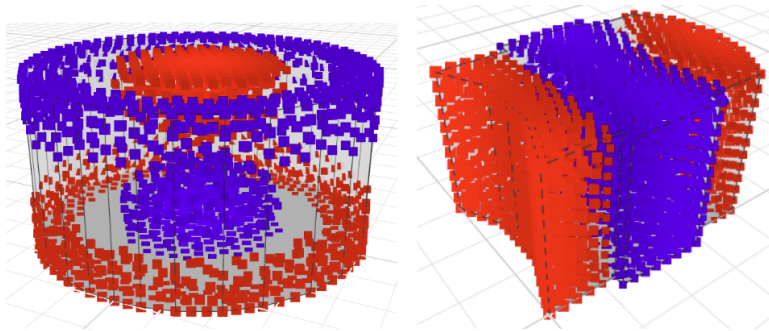


Figure 4.1: Visualizing the pressure at specific modes via a cloud of sprites. Red and blue both mean mean pressure (loud). In between the red and blue zones the mode frequency is barely audible. This pattern is known to the users from the free version for rectangular rooms and described in the article about the calculator and the theory of room modes on the website [roo].

4.2 Distributed Architecture

As stated in Chapter 3, one of the main architectural decisions is to provide the tool as an easily accessible website and do the physics calculations on the server. This leads to a distributed system with different parts communicating over an API.

In addition, the tool is not its own website but is meant to be a micro frontend (MFE) embeddable in an existing website (amcoustics.com). A website of which the author of this thesis is the founder and developer.

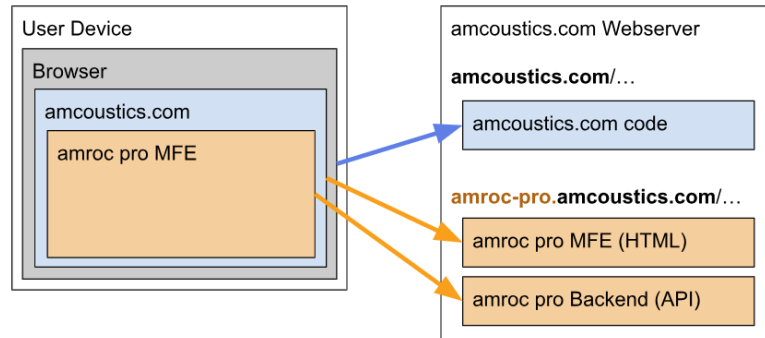


Figure 4.2: The tool is embedded in a website as micro frontend (MFE). The coupling between host and MFE is very low, meaning the website does not know of any details of the tool and vice versa. Therefore, the MFE talks to its own backend endpoints.

The backend is deployed as a docker container to enable easy switching of server hardware or hoster. The main parts of the backend app are the spaces and FEM parts. Those can be seen as self-contained modules and could also be deployed as two separate apps. The spaces part includes all logic about building, loading, and saving models (spaces), and the FEM part includes all logic regarding FEM processing and retrieving FEM results. If other acoustic calculation methods should be added in the future, the spaces part can stay the same, and a new calculation part would be added. To allow for this easy expansion the spaces part does not know about the FEM part at all.

The main backend and the FEM processor are deployed as different docker containers (Figure 4.3). This allows for separate scaling. The main backend can be deployed on a low-cost server that is available 24/7. The FEM processor can, for example, be deployed to a high-performance cloud server that is only started on demand.

The current implementation only uses one FEM processor, but it should be easily possible to add the necessary logic to use multiple processors in parallel in the future (Figure 4.4).

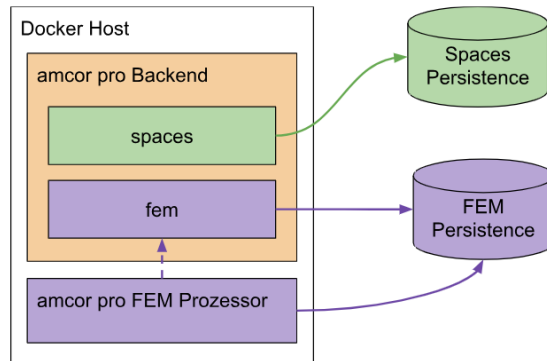


Figure 4.3: The main backend and the FEM processor are deployed as different docker containers to allow for separate deploying and scaling.

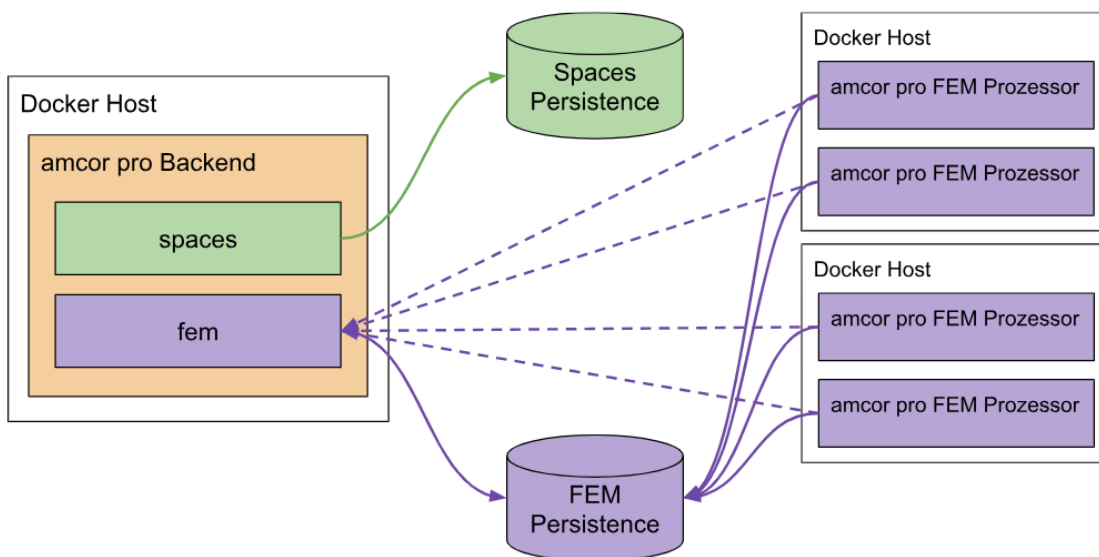


Figure 4.4: The FEM processor can be deployed in parallel if necessary

4.3 FEM Processing - Communicating with GMSH and OpenCFS

The current FEM processor uses Gmsh [gmsb] to create a mesh from the model and openCFS [opec] to calculate the acoustical resonances in that volume (room modes). The server-side code of the app communicates with those two tools by saving files to the file system and executing the tools, pointing them to those input files. The tools output their data to files as well. So, the output of each process step is a file.

The commanding device can be another server, a proxy server, or the user's browser. The FEM command is a request to the FEM processor API. The calculation itself is done asynchronously. This means that the commanding request is returned before the calculation starts. The commanding device can open a WebSocket with the process ID to get updates on the process. As soon as the process is finished, result files can be downloaded. See the process as sequence diagram in Figure 4.5.

Commands to start a FEM process are idempotent. The FEM processor checks which files of the process are missing. If it is a new process (new process id), the first process step is invoked. If the process id already exists but files are missing, the process is restarted at the specific step where output files are missing. If the result files already exist, the processor simply sends an "already done" as result to the command.

On startup, the app checks all persisted processes in this way, which leads to the automatic finishing of every not-yet-finished process. Should, for example, the server be restarted during a FEM calculation, the process is finished as soon as the server is back up. As the frontend tries to reconnect to the backend, the user gets the updates after such a server restart.

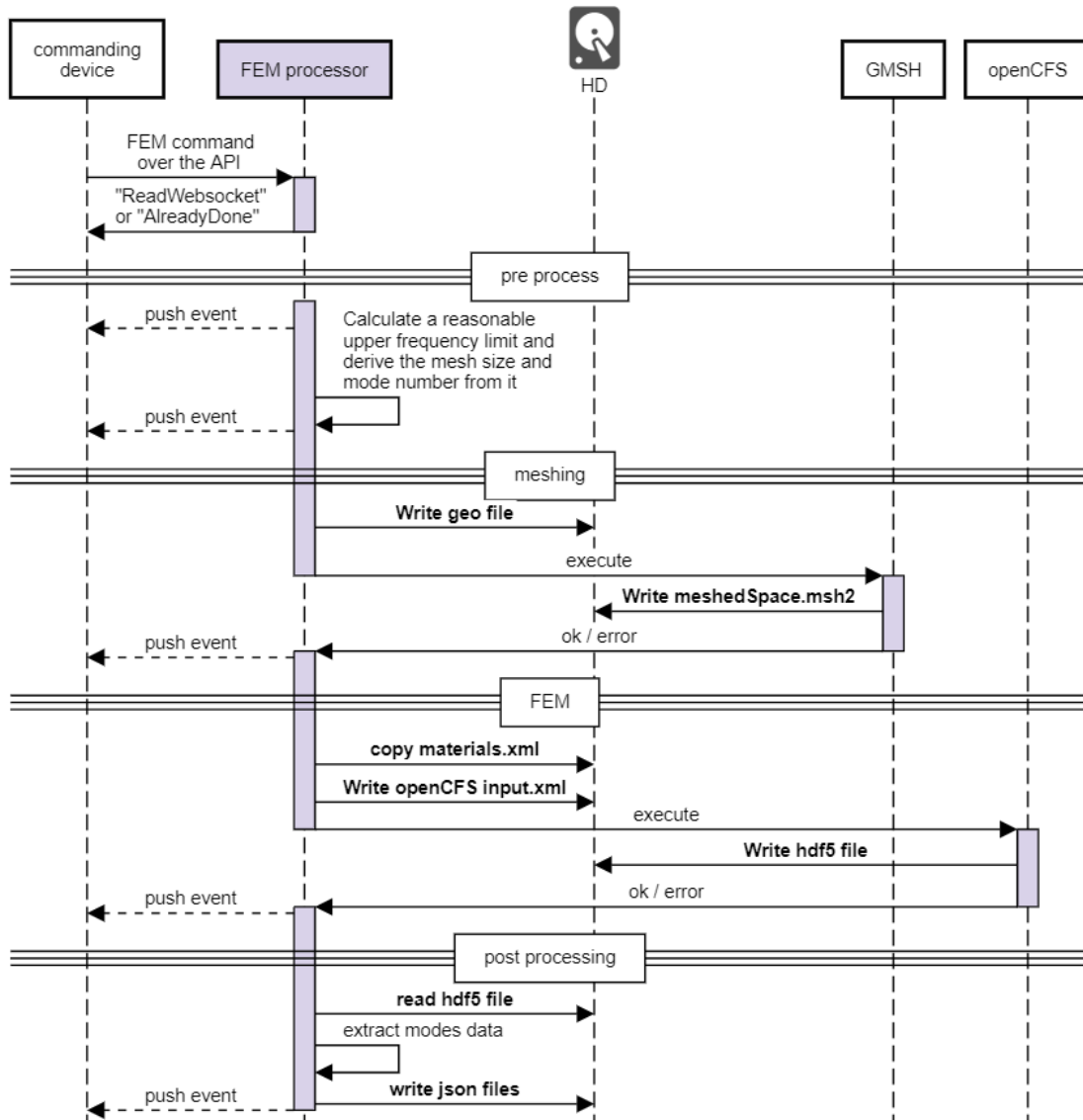


Figure 4.5: The process to do the FEM on the Server

4.4 Input and Output Files

As described in Section 4.3 the different building blocks save configurations and results in files. Here is a short description and references to the used files.

geo file - Gmsh input file

The .geo files are written in Gmsh's built-in scripting language [gmsa, Ch.5]. It is a text file describing the model and some relevant parameters, such as the desired resolution, to use in meshing.

msh file - Gmsh output file

The MSH file is a text file and contains one mandatory section about the file, followed by several optional sections describing the meshed model and post-processing data [gmsa, Ch.10.1].

materials.xml and Input.xml - openCFS input files

The materials.xml file contains a database of materials that can be used in calculations done by openCFS. The materials can be referenced in the main input.xml file.

The input.xml file is the main input file for the openCFS calculation and contains information about where to find the mesh and material data and what should be calculated [opeb].

cfs file - openCFS output files

This file is written in the hdf5 format [opea] a fileformat built for fast I/O processing and storage [hdf].

JSON files - cached result files

Multiple files are saved to disk for fast retrieval.

- `mode_frequencies.json` contains a one-dimensional json array with the frequencies of the modes
- `coordinates.json` contains a one-dimensional json array with the x,y,z coordinates of the points the mode files contain pressure values for. Therefore, the array size is three times the number of pressure points. The pressure points of all modes use the same coordinates. Therefore, the `coordinates.json` data must only be downloaded once, not for every mode result.
- For each mode, a JSON file exists containing one pressure value per pressure point in the same order as the `coordinates` file, which contains the coordinates of those points.

Results

The goal of making calculations of room modes in non-rectangular rooms easier, has been achieved. The prototype described in this work is online and has been used by hundreds of people by now. Informal discussions with some testers have shown that first results can easily be retrieved in minutes when the service is first used.

The approaches to solving the technical challenges have proven to be effective. Visualization and 3D modeling works with high frame rates thanks to modern browser technologies. The distributed communication with the calculation backend is reliable and fast. Automatic parameter selection leads to very fast setup and computation times.

The market for room mode calculations has changed in the course of the work. While there was no easy to use service to calculate room modes for non-rectangular rooms when the thesis topic was agreed on, there is at least one such service in addition to this prototype available now. A comparison of the approaches and calculation results has been included in Chapter 2.

Conclusions

For the special application of calculating room modes in non-rectangular rooms, the prototype described in this paper offers a much simpler and faster approach than previous methods. Much more powerful, but also more expensive and more difficult to learn software already existed. The focus on the use case and the specially designed user interface complement the acoustician's toolbox with a customized and well-functioning special solution. It allows even non-experts such as hi-fi enthusiasts or many studio builders to quickly answer the question of where problematic room modes interfere with good acoustics.

The next steps will be to add more modeling options and features to facilitate the import of existing models. Further visualization techniques and different simulation types will improve the usefulness now that the technical principle has been proven to be effective. As the current simulation performs far better than expected, more complicated calculations can be considered with confidence.

List of Figures

2.1	A rough categorization of existing software to simulate room modes . . .	5
2.2	Free online room mode calculator, that can only calculate room modes for rectangular rooms [amra]	6
2.3	Audieum Modal room shape presets	8
2.4	Audieum Modal settings for an L-shaped room	8
3.1	Displaying the 3D model to the user together with some values about the room before doing the acoustics calculation	12
3.2	Showing a frequency diagram over the 3D model with the selectable mode frequencies as vertical lines	13
3.3	Visualizing the pattern of the sound pressure of one selected mode via a cloud of sprites. Red and blue both mean pressure (loud). In between the red and blue zones the mode frequency is barely audible. This pattern is known to the users from the free version for rectangular rooms and described in the article about the calculator and the theory of room modes on the website [roo]. .	14
3.4	Assuming the same development time, deciding to focus on usability means deciding against some other step in the software development, like feature-completeness or realism.	15
3.5	The 2D canvas shown to the user when the app loads.	16
3.6	A color change of the box indicates the interactivity of the handle	17
3.7	Creation of new corners	17
3.8	Shaping the floor	18
3.9	Invalid floor shapes immediately turn red to provide fast feedback.	19
3.10	A survey among users about most wanted next features in amroc pro (the new online name of the application described in this work [amrb])	21
3.11	Increasing mode density with frequency	24
4.1	Visualizing the pressure at specific modes via a cloud of sprites. Red and blue both mean pressure (loud). In between the red and blue zones the mode frequency is barely audible. This pattern is known to the users from the free version for rectangular rooms and described in the article about the calculator and the theory of room modes on the website [roo].	29

- 4.2 The tool is embedded in a website as micro frontend (MFE). The coupling between host and MFE is very low, meaning the website does not know of any details of the tool and vice versa. Therefore, the MFE talks to its own backend endpoints. 30
- 4.3 The main backend and the FEM processor are deployed as different docker containers to allow for separate deploying and scaling. 31
- 4.4 The FEM processor can be deployed in parallel if necessary 31
- 4.5 The process to do the FEM on the Server 33

Bibliography

- [ALD11] Niels Adelman-Larsen and Jens Dammerud. A survey of reverberation times in 50 European venues presenting pop & rock concerts. *Proceedings of Forum Acusticum*, 01 2011.
- [ALTG10] Niels Adelman-Larsen, Eric Thompson, and Anders Gade. Suitable reverberation times for halls for rock and pop music. *The Journal of the Acoustical Society of America*, 127:247–55, 01 2010.
- [amra] amroc - THE room mode calculator. <https://amcoustics.com/tools/amroc>. Accessed: 2024-04-24.
- [amrb] amroc pro - room mode calculator for non-rectangular rooms. <https://amcoustics.com/tools/amroc-pro>. Accessed: 2024-06-12.
- [AO09] Marc Aretz and Raf Orłowski. Sound strength and reverberation time in small concert halls. *Applied Acoustics*, 70(8):1099–1110, 2009.
- [aud] Audieum Modal. <https://audieum.com/modal>. Accessed: 2024-04-25.
- [bab] Website of the threejs library. <https://www.babylonjs.com/>. Accessed: 2024-04-08.
- [Bar88] Mike Barron. Subjective study of British symphony concert halls. *Acta Acustica united with Acustica*, 66, 01 1988.
- [ble] Blender. <https://www.blender.org/>. Accessed: 2024-04-30.
- [com] The Comsol Website. <https://www.comsol.com/>. Accessed: 2024-06-10.
- [FCP08] Angelo Farina, Daniel Commins, and Nicola Prodi. Experimental analysis of the acoustical behaviour of Musikverein in concert and ballet configurations. *The Journal of the Acoustical Society of America*, 123:2973, 06 2008.
- [gmsa] Gmsh Reference Manual. <https://gmsh.info/dev/doc/texinfo/gmsh.pdf>. Accessed: 2024-06-10.
- [gmsb] The Gmsh Website. <https://gmsh.info/>. Accessed: 2024-04-30.

- [gri] GitHub - Infinite Grid Shader. <https://github.com/Fyrestar/THREE.InfiniteGridHelper>. Accessed: 2024-04-30.
- [hdf] The HDF5 file format. <https://www.hdfgroup.org/solutions/hdf5/>. Accessed: 2024-06-10.
- [HN22] Takayuki Hidaka and Noriko Nishihara. Favorable reverberation time in concert halls revisited for piano and violin solos. *The Journal of the Acoustical Society of America*, 151(3):2192–2206, 03 2022.
- [Kaa13] Stefan Kaak. Zur Standardisierung der Akustik musikalischer Aufführungsräume. Master’s thesis, TU Berlin, 2013.
- [lib] A list of WebGL and WebGPU frameworks and libraries. <https://gist.github.com/dmnsn/76878ba6903cf15789b712464875cfdc>. Accessed: 2024-02-15.
- [LKV09] Klaus-Hendrik Lorenz-Kierakiewitz and M. Vercammen. Acoustical Survey of 25 European Concert Halls. In *NAG/DAGA 2009 - Rotterdam*, 2009.
- [mat] The Matlab Website. <https://mathworks.com/products/matlab.html>. Accessed: 2024-06-10.
- [MM04] Gerhard Müller and Michael Möser. *Taschenbuch Der Technischen Akustik*. Engineering online library. Springer Berlin Heidelberg, 2004.
- [opea] openCFS User Documentation - Data Output. <https://opencfs.gitlab.io/userdocu/DataExplanations/DataInputOutput/#data-output>. Accessed: 2024-06-10.
- [opeb] openCFS User Documentation - XML-Input-Files. <https://opencfs.gitlab.io/userdocu/XMLExplanations/>. Accessed: 2024-06-10.
- [opec] The openCFS Website. <https://www.opencfs.org/>. Accessed: 2024-04-30.
- [PJH16] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2016.
- [Pri23] Albert G. Prinn. A Review of Finite Element Methods for Room Acoustics. *Acoustics*, 5(2):367–395, 2023.
- [RBH12] Birgit Rasmussen, Jonas Brunskog, and Dan Hoffmeyer. 06 2012.
- [rmca] bob golds room mode calculator. <https://www.bobgolds.com/Mode/RoomModes.htm>. Accessed: 2024-06-10.

- [rmcb] hunecke room mode calculator. <https://www.hunecke.de/en/calculators/room-eigenmodes.html>. Accessed: 2024-06-10.
- [rmcc] mcsquared room mode calculator. <http://www.mcsquared.com/metricmodes.htm>. Accessed: 2024-06-10.
- [rmcd] trikustik room mode calculator. <https://trikustik.at/en/knowledge/room-modes-calculator/>. Accessed: 2024-06-10.
- [roo] Room Modes Theory. <https://amcoustics.com/articles/roommodes>. Accessed: 2024-06-10.
- [SE02] Philip J. Schneider and David Eberly. *Geometric Tools for Computer Graphics*. Elsevier Science Inc., USA, 2002.
- [ske] SketchUp. <https://www.sketchup.com/>. Accessed: 2024-04-30.
- [SWLL05] Jochen Seidel, Lutz Weber, Philip Leistner, and Sebastian Laschczok. 01 2005.
- [T3g] ThreeJS Docs - Geometry Class. <https://threejs.org/docs/#api/en/core/BufferGeometry>. Accessed: 2024-04-11.
- [T3l] ThreeJS Docs - Light Class. <https://threejs.org/docs/#api/en/lights/Light>. Accessed: 2024-04-11.
- [T3m] ThreeJS Docs - Material Class. <https://threejs.org/docs/#api/en/materials/Material>. Accessed: 2024-04-11.
- [T3o] ThreeJS Docs - Object3D Class. <https://threejs.org/docs/#api/en/core/Object3D>. Accessed: 2024-04-11.
- [thr] Website of the threejs library. <https://threejs.org/>. Accessed: 2024-04-08.
- [tre] treble. <https://www.treble.tech/>. Accessed: 2024-04-25.
- [weba] WebGL - Getting Started. https://www.khronos.org/webgl/wiki/Getting_Started. Accessed: 2024-02-15.
- [webb] WebGPU - API. https://developer.mozilla.org/en-US/docs/Web/API/WebGPU_API. Accessed: 2024-02-15.