

דו"ח סיכום פרויקט: מיוחד

סימולטור סאונד Gaming Audio Simulator

מבצע:

Aviv Azran אביב עזרן

מנחה:

Hadas Ofir הדס אופיר

סמסטר רישום: אביב תש"פ

תאריך הגשה: ספטמבר, 2020

20-2-5844P

הדו"ח המסכם הזה והפרויקט כולו מוקדשים לאמא שלי, יעל-חיה עזרן ז"ל שהלכה לעולמה בטרם עת ממחלת הסרטן.

ת.נ.צ.ב.ה

תודות

אני מודה מקרב לב לבת הזוג שלי, ליאל בלילה, שהכילה אותי כל פעם שכבר אמרתי נואש, שנתנה ונותנת לי את הכוח להמשיך ליצור. אני מודה גם ליאיר משה שכתב תבנית זו לדו"ח מסכם ובכך עזר לי לכתוב דו"ח מסכם בהצלחה.

תוכן עניינים

1. דרישות מערכת	8
2. אבני היסוד של הסימולטור	9
2.1. מנוע Unreal Engine 4	9
2.2. תוסף Steam Audio	9
2.3. תוסף Unreal Web Server	9
2.4. Python API	9
3. הסאונד בסימולטור	10
3.1. אופן הפקת הסאונד הנשמע	10
3.2. הגדרות הסאונד והתוסף	11
הגדרות הניתנות לשינוי דרך הסימולטור	11
הגדרות הניתנות לשינוי דרך המנוע	12
4. מבנה תכנותי ותקשורת עם הסימולטור	14
4.1. שכבת המנוע – תפקידים ומחלקות	15
מחלקת BP_SoundObjectBase	15
מחלקת BP_SOMActorOrbit	15
מחלקת BP_SoundSource	15
מחלקת BP_Microphone	15
מחלקת BP_PlayerController	16
מחלקת SimulationAction	16
4.2. שכבת פייתון – תפקידים ומחלקות	17
מחלקת UnrealSoundSimulator._CUnrealClient	17
מחלקת UnrealSoundSimulator._Room	18
4.3. התקשורת עם הסימולטור	18
5. ממשק משתמש ופקודות סקריפט	19
5.1. תחילת עבודה עם פייתון	19
5.2. הוספת מקורות קול ומיקרופונים	20
5.3. תנועה אורביטלית	21

21.....	מקור קול.....	5.4.
22.....	מיקרופון.....	5.5.
22.....	חדר.....	5.6.
22.....	הגדרות סאונד גלובליות.....	5.7.
23.....	תצוגת מערכת צירים לוקאלית והרצת סימולציה.....	5.8.
23.....	תפריט שינוי הגדרות גרפיקה.....	5.9.
6. חומר אקוסטי 24		
24.....	בליעה (Absorption).....	6.1.
25.....	העברה (Transmission).....	6.2.
25.....	פיזור (Scattering).....	6.3.
26.....	חומרים מובנים בסימולטור.....	6.4.
7. סיכום 27		
27.....	מטרת הפרויקט והיעדים שהושגו.....	7.1.
27.....	קשיים בהם נתקלתי במהלך הפיתוח.....	7.2.
28.....	מה רכשתי מהפרויקט.....	7.3.
8. נספחים 29		
29.....	נספח - הרחבת הסימולטור ושינוי הגדרות פנימיות.....	8.1.
30	התקנת Intel Embree.....	
30	התקנת המנוע.....	
31	הורדת ופתיחת הפרויקט.....	
33	Packaging.....	
34	הוספת חומר אקוסטי חדש.....	
36	שינוי HRTF Interpolation method ו-Spatialization Method.....	
37	שינוי ההגדרות של תוסף Steam.....	
39	נספח - קישורים שימושיים.....	8.2.
40	נספח - רשימת פקודות Python API.....	8.3.

רשימת איורים

9	איור 1: תשתית הסימולטור.....
11	איור 2: Steam Audio Pipeline.....
14	איור 3 : מחלקות הסימולטור והקשרים ביניהן.....
15	איור 4: שכבת המנוע.....
17	איור 5: שכבת פייתון.....
18	איור 6 : Simulator Sequence Diagram.....
19	איור 7: סימולטור סאונד - מסך ראשי.....
20	איור 8: תפריט רדיאלי להוספת אובייקט סאונד.....
20	איור 9: תפריט בחירת מאפייני אובייקט סאונד.....
21	איור 10: תפריט תנועה אורביטלית.....
21	איור 11: תפריט מקור קול.....
22	איור 12: תפריט מיקרופון.....
22	איור 13: תפריט הגדרות חדר.....
23	איור 14: תפריט הגדרות סאונד גלובליות.....
24	איור 15: תדר מרכזי.....
25	איור 16 בליעה.....
25	איור 17: העברה.....
25	איור 18: ערך פיזור '0'.....
26	איור 19: ערך פיזור '1'.....
32	איור 20: פתיחת הפרויקט לעריכה.....

רשימת טבלאות

17.....	טבלה 1 : מחלקות שקולות.....
24	טבלה 2: תדרים מרכזיים בתוסף.....
26	טבלה 3: חומרים מובנים בסימולטור.....
30	טבלה 4: תוספים למנוע.....
39	טבלה 5: קישורים שימושיים.....

תקציר

Sound Simulator הוא סימולטור זמן-אמת שפותח במנוע Unreal Engine 4.25 ומשתמש ב-STEAM AUDIO. הסימולטור מאפשר למשתמש לבחון את השינויים שחווה סאונד הנשמע בתוך חדר, כתלות בשינויים בממדי החדר, בחומר ממנו הוא עשוי, במיקום מקור(ות) הקול ובמיקום ואוריינטציית המיקרופון דרכו הקול נשמע. החומרים מהם ניתן לבחור מוגדרים ע"י קבועים המייצגים תכונות אקוסטיות של החומר: absorption, transmission, scattering. כל אחד מהקבועים מקושר לטווח מסוים של תדירויות השייך, בחלוקה גסה, לאחד משלושה טווחים: תדרים גבוהים, תדרים בינוניים ותדרים נמוכים. המשתמש יכול למקם מספר כלשהו של מקורות קול בחדר, כל אחד מהם מחזיק קובץ wav כלשהו אותו הוא יכול להשמיע. בנוסף ניתן למקום מספר מיקרופונים אותם ניתן אף לסובב. ניתן להגדיר תנועה אורביטלית אותה יבצעו המיקרופונים ומקורות הקול במהלך הסימולציה. כאשר המשתמש משנה את הפרמטרים הנשלטים של הסימולציה בזמן אמת, הוא יוכל להבחין בתופעות פיזיקליות הקשורות לסאונד. בנוסף, המשתמש יוכל לבנות את הסביבה בקונפיגורציה מסוימת ולהריץ סימולציה מלאה אשר תקליט את הסאונד בחדר כפי שהוא נקלט בכל אחד מהמיקרופונים ותייצא את הפלט לקבצי wav להמשך ניתוח. לבסוף, ניתן לשלוט בסימולטור באמצעות ממשק משתמש אינטואיטיבי מובנה, או דרך סקריפט פייתון לצורך אוטומציה וביצוע סימולציות מורכבות יותר.

Abstract

Sound Simulator is a real-time simulator developed with Unreal Engine 4.25 and STEAM AUDIO. The simulator enables the user to explore the changes of sounds played in a room due to changes in the room's dimension, the material it's made of, the location of the sound source and the microphone's location and orientation. The materials from which the room is made of are defined by some absorption, transmission and scattering values, each correspond with a range of frequencies which can be broadly grouped as "Low/Mid/High" ranges. The user can place multiple sound sources in the room, each assigned with a unique wav file and multiple microphones. In addition, microphones and sound sources can be configured to perform orbital motion during the simulation. As the user adjusts the different parameters in real-time, physical sound-related phenomenon will be heard. In addition, the user will be able to prepare a setup of the scene and run a full simulation which will record the sound as it is heard to each microphone and produce a corresponding wav file for further research. Finally, Controlling the simulator can be done via the GUI, as well as with Python for advanced simulations.

1. דרישות מערכת

הסימולטור יכול לרוץ על כל מחשב עם מעבד אינטל התומך ב- SSE, AVX, AVX2, and AVX-512, ומותקנת עליו מערכת הפעלה Windows 64bit. התקורה החישובית של הסימולטור מוגדרת עפ"י חוזק המפרט של המכונה עליו מריצים את הסימולטור, כמו גם הגדרות הסאונד בסימולטור, מספר מקורות הקול בסימולציה, הגדרות הגרפיקה בסימולטור וכו'. לצורך פינוי משאבים לטובת חישוב הסאונד במקרה של צורך לסמלץ ריאליזם גבוה, הוספתי תפריט פנימי בסימולטור שמטרתו לאפשר שינוי הגדרות גרפיות בצורה נוחה. ניתן ללחוץ על מקש Esc ולפתוח את התפריט. השיפור המשמעותי ביותר יגרם ע"י שינוי הרזולוציה לרזולוציה נמוכה יותר. כדי להשתמש בPython API, יש להתקין Python 3.8 ואת החבילה requests. מומלץ להתקין את ההפצה של אנקונדה.

2. אבני היסוד של הסימולטור

בפרק זה אסקור בקצרה את התשתית עליה מבוסס הסימולטור. ניתן לחלק את התשתית לשני חלקים כפי שמתואר באיור 1: תשתית הסימולטור. האובייקטים תחת Unreal Engine Infrastructure הינם פנימיים בתוך המנוע עצמו, בעוד שממשק הפיתוח חיצוני למנוע. בלוק המנוע לא תלוי תכנותית בבלוק הפיתוח ויכול לרוץ בצורה עצמאית ללא התקנת פיתוח במערכת.



איור 1: תשתית הסימולטור

2.1 מנוע Unreal Engine 4

המנוע הגרפי עליו בנוי הסימולטור. המנוע יצא לראשונה בשנת 1998 עם יציאתו של משחק היריות בגוף ראשון Unreal, ומאחר יותר עם עוד מגוון רחב של משחקים אחרים. ל Unreal Engine יכולות גרפיות גבוהות מאז הגרסה הראשונה ונמצא בשימוש של חברות רבות המפתחות משחקי מחשב. הוא נחשב למתחרה העיקרי של המנוע הגרפי Unity. המנוע כתוב בשפת C++ ומאפשר הרחבות שונות ע"י תוספים. נחשב לפרויקט קוד פתוח ולכן מסופק בחינם יחד עם גישה לקוד המקור שלו למעט הפצת הפרויקט למטרות רווח.

2.2 תוסף Steam Audio

מגיע כתוסף מובנה בתוך מנוע Unreal. תוסף זה אחראי על חישוב הסאונד בצורה פיזיקלית ואופן הפקת הסאונד באמצעותו מפורט ב-3.1: אופן הפקת הסאונד הנשמע. הגדרות הקשורות לתוסף וניתנות לשינוי מפורטות ב-3.2: הגדרות הסאונד והתוסף.

2.3 תוסף Unreal Web Server

תוסף אותו ניתן להוריד מ- Unreal Market (בתשלום). תוסף זה מאפשר יצירת שרת פנימי בתוך מנוע Unreal. לשרת זה ניתן להגדיר פורט דרכו ניתן להתחבר אליו כלקוח ולשלוח בקשות Http אשר יתורגמו בתוך המנוע לפקודות מתאימות.

2.4 Python API

אוסף הפקודות המאפשרות לשלוט בסימולטור בצורה תכנותית. האוסף מכיל את כל הפקודות הדרושות ליצירת הסצנות בסימולטור והרצת הסימולציות.

3. הסאונד בסימולטור

בפרק זה אסביר את כיצד מיוצר הסאונד במנוע ובאיזה שלבים בתהליך מתבצעת התממשקות עם תוסף Steam Audio. בנוסף, אפרט מה הן הגדרות הסאונד הניתנות לשינוי מתוך הסימולטור ואילו מצריכות עריכה של קוד המקור ואריזה מחדש.

3.1. אופן הפקת הסאונד הנשמע

על מנת לאפשר המון תהליכים מקבילים, המנוע עובד בצורה של multithreading. הפקת הסאונד בסימולטור הוא תהליך הנמצא באחריותם של המנוע עצמו ושל התוסף של Steam. נסביר בקצרה על התהליכונים (threads) הרלוונטיים לתהליך, כאשר את תרשימים הבלוקים שמתאר את פעולת התוסף ניתן לראות באיור 2: Steam Audio Pipeline.

תהליכון המשחק (Game Thread)

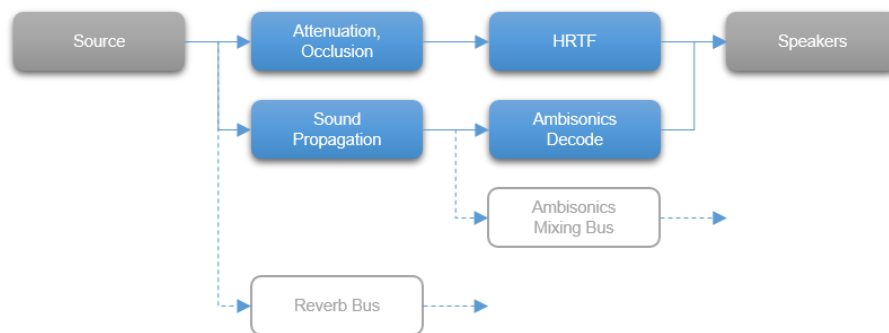
זה התהליכון השולט על מצב המשחק. בעת הצורך, הוא שולח את המידע שלו לתהליכון הסימולציה. תהליכון זה מנוהל ע"י המנוע והוא רץ בקצב בו המנוע בוחר להריץ אותו (בהתאם למשתנים כמו מפרט המחשב עליו הסימולטור רץ ועוד). הקצב הזר לרוב יהיה 60 Hz (בהנחה ש-VSync פעיל – בסימולטור הוא פעיל).

תהליכון הסימולציה (Simulation Thread)

זה התהליכון שמבצע בפועל את סימולציית התפשטות הסאונד ועליו מתבצע חלק הארי של העבודה החישובית. הוא משתמש במידע על המקור והמאזין המסופק לו ע"י תהליכון המשחק ומחשב את ה-Ray Impulse Response בה משתמש תהליכון הרינדור (Rendering Thread). זהו התהליך המשלב בתוכו Ray-tracing. תהליכון זה מנוהל פנימית ע"י התוסף של Steam Audio, והוא רץ בקצב המהיר ביותר האפשרי עבורו, אך לעולם לא מהר יותר מתהליכון הרינדור.

תהליכון הרינדור (Rendering Thread)

זהו התהליכון שמפעיל את את האפקטים הקשורים ל-Direct Occlusion, 3D audio ואפקטים סביבתיים לכל אחד מהמקורות. התוסף מוסיף שלב נוסף בין אות הסאונד הבדיד המופק מתהליכון הסאונד לבין כרטיס השמע של המשתמש. בשלב זה מתבצע קוד DSP המבצע קונבולוציה מולטי-ערוצית עם תגובות ההלם שמופקות ע"י תהליכון הסימולציה. חוט זה רץ בקצב של ה-Audio DSP rate – בד"כ 1024 דגימות לפריים 44100 דגימות לשניה.



איור 2: Steam Audio Pipeline

3.2. הגדרות הסאונד והתוסף

בפרק זה נסביר על הגדרות הסאונד השונות בסימולטור והשפעתן על הסאונד המופק. ראשית, חשוב להבין כי לא כל ההגדרות ניתנות לשליטה דרך הסימולטור עצמו. זאת כיוון שבתהליך האריזה, מחלקות מסוימות של התוסף נארזות על הגדרותיהן ומאיתנו נמנעת הגישה להגדרות הללו. הדרך לשנות את ההגדרות הללו, כמו גם פעולות מסוימות נוספות, מצריכה התקנה של המנוע ועריכה של קוד המקור של הפרויקט. הסברים כיצד לעשות זאת מופיעים בפרק: נספח - הרחבת הסימולטור ושינוי הגדרות פנימיות.

הגדרות הניתנות לשינוי דרך הסימולטור

Direct Occlusion Mode

Direct Occlusion Mode specifies how to model sources that are occluded by solid objects.

- **None.** Occlusion calculations are disabled. Sounds can be heard through walls and other solid objects.
- **Direct Occlusion, No Transmission.** Occlusion calculations are enabled. Occluded sound is inaudible.
- **Direct Occlusion, Frequency Independent Transmission.** Occlusion calculations are enabled. Occluded sound is attenuated as it passes through geometry, based on the material properties of the occluding object. The attenuation is independent of frequency.
- **Direct Occlusion, Frequency Dependent Transmission.** Occlusion calculations are enabled. Occluded sound is filtered as it passes through geometry, based on the material properties of the occluding object. The filtering is dependent on frequency, so for example high frequencies may be attenuated more than low frequencies as the sound passes through geometry.

Direct Occlusion Method

This dropdown is enabled whenever Direct Occlusion Mode is set to anything other than None. Specifies the algorithm used by Steam Audio for modeling occlusion. The options are:

- **Raycast.** Performs a single ray cast from source to the listener to determine occlusion. If the ray is occluded, direct sound is blocked.

- **Partial.** Performs multiple ray casts from source to the listener, treating the source as a sphere with a specified radius. The volume of the sound source is adjusted based on the portion of the source visible from the listener. Transmission calculations, if enabled, are only applied to the occluded portion of the direct sound.

Direct Occlusion Source Radius

Specifies the radius of the sphere to use when modeling Partial occlusion. Ignored if Direct Occlusion Method is set to Raycast.

Physics-Based Attenuation

When checked, physics-based distance attenuation (inverse distance falloff) is applied to the audio.

Air Absorption

When checked, frequency-dependent, distance-based air absorption is applied to the audio. Higher frequencies are attenuated more quickly than lower frequencies over distance.

Speed of Sound

This slider sets the speed of sound in the scene in meters per second.

הגדרות הניתנות לשינוי דרך המנוע

Ambiosonics Order

This determines the directionality of environmental effects. Increasing this increases the compute complexity quadratically. Use zero order Ambiosonics if no directionality is needed in environmental effects. Otherwise, first order Ambiosonics should provide a good tradeoff between directionality and CPU usage.

Impulse Response Duration

This is the length of the impulse responses to generate, in seconds. Increasing this improves the quality of the simulation, but beyond a certain point (depending on the number of sound sources), may result in audio glitching.

Indirect Spatialization Method

If set to Panning, Steam Audio will apply a standard panning algorithm to render the Ambiosonics-encoded environmental effects. If set to HRTF, Steam Audio will decode the environmental effects using an HRTF-based binaural rendering algorithm, for improved spatialization of indirect sound.

Irradiance Min Distance

The minimum distance between a source and a scene surface, used when calculating the energy received at the surface from the source during indirect sound simulation. Increasing this number reduces the loudness

of reflections when standing close to a wall; decreasing this number results in a more physically realistic model.

Max Sources

This is the maximum number of sound sources that can have source-centric reverb enabled. For the purposes of this setting, listener-centric reverb counts as a source. For example, if Max Sources is set to 8, and you are using listener-centric reverb, then you can have up to 7 sources that use source-centric reverb.

Real-Time Quality Settings

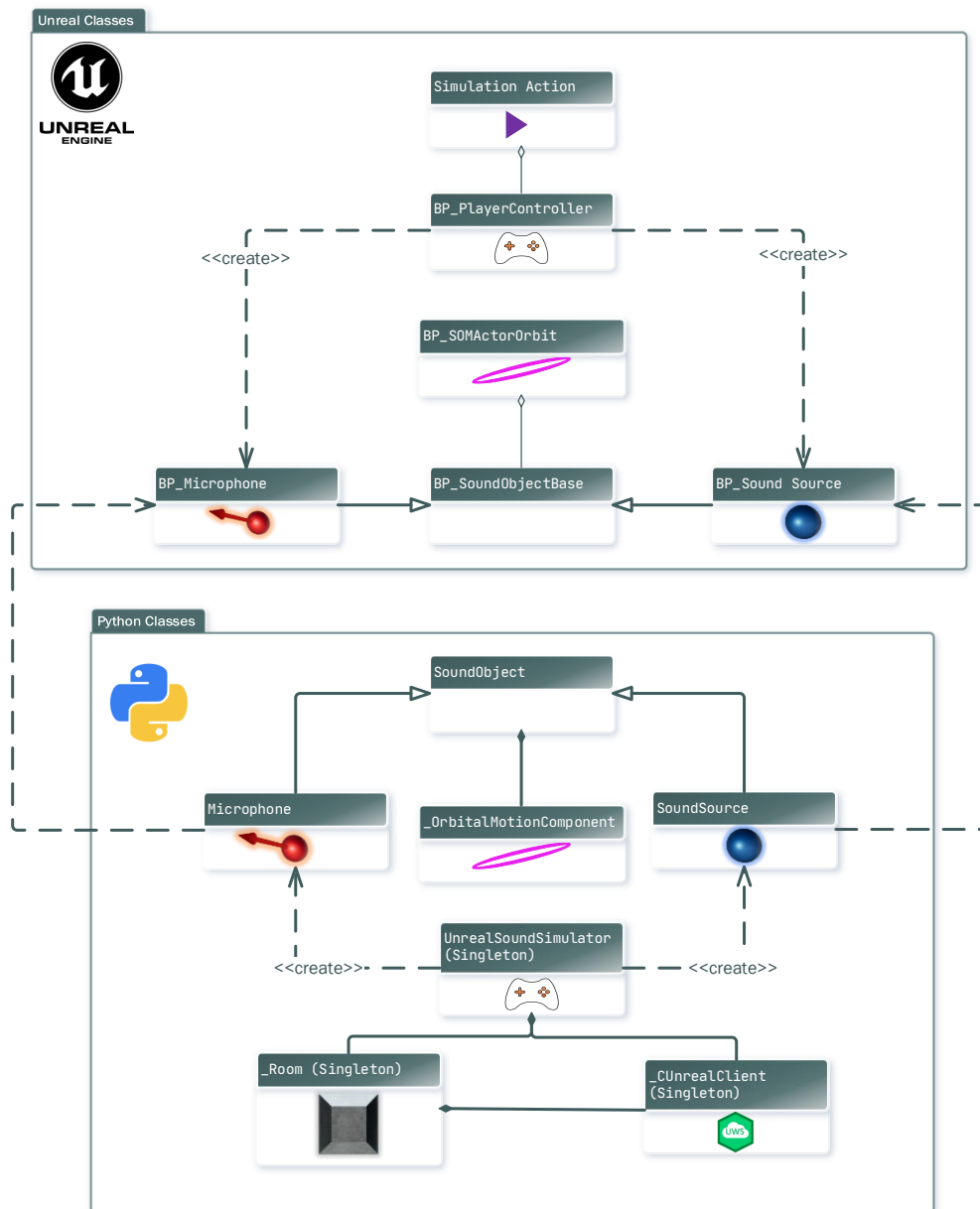
These settings let you fine-tune how Steam Audio simulates physics-based indirect sound. You can select one of the presets from the Quality Preset drop-down, or select Custom, and adjust the following settings.

Real-time CPU Cores (%). Percentage of CPU cores to use on an end user's machine for performing real-time computation of environmental effects. The percentage can also be interpreted as the number of threads to create as a percentage of the total logical cores available on the machine of an end user. Increasing real-time CPU usage leads to faster update of the simulation and lower latency.

- **Rays.** This is the number of primary and reflection rays to trace from the listener position for real-time computation of environmental effects. Increasing this improves the quality of the simulation, at the cost of performance.
- **Secondary Rays.** This is the number of directions that are sampled when simulating diffuse reflection. Setting this number too low may reduce the overall quality.
- **Bounces.** Number of times the rays are allowed to bounce off of solid objects in real-time. Increasing this improves the quality of the simulation, at the cost of performance.

4. מבנה תכנותי ותקשורת עם הסימולטור

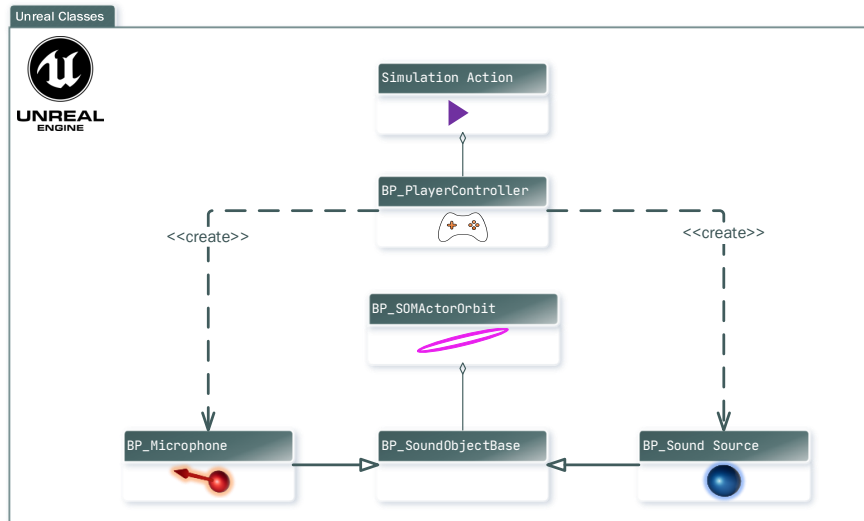
הסימולטור בנוי עפ"י עקרונות OOP, וניתן לתאר את המבנה שלו בשתי שכבות, שכבת המנוע ושכבת פייתון. היות וניתן להשתמש בסימולטור כ-standalone, שכבת המנוע מכילה את הלוגיקה כולה בעוד ששכבת הפייתון מאפשרת למשתמש לכתוב סקריפטים לביצוע סימולציות בצורה אוטומטית ותפקידה אך ורק לקשר בין הפקודה לאפשרות הGUI המתאימה. במילים אחרות, פקודה בפייתון משנה את הGUI ולא את האובייקטים עצמם. האובייקטים מגיבים לשינוי הGUI. צורת תקשורת זו מטרתה לשמר עקביות של מצב הGUI עם הסצנה לאחר הרצת פקודות בפייתון. המשמעות היא שבכל שלב ניתן לעבור ממצב שליטה ידנית לשליטה דרך קוד וההיפך. באיור 3 : מחלקות הסימולטור והקשרים ביניהן ניתן לראות את מבנה המחלקות הכללי.



איור 3 : מחלקות הסימולטור והקשרים ביניהן

4.1. שכבת המנוע – תפקידים ומחלקות

היות ואין תלות תכנותית בין ה-Python API לבין הסימולטור עצמו, בשכבה זו ממומשת למעשה הלוגיקה כולה, החל מהמחלקות המתארות את אובייקטי הסאונד ועד GUI. כמו כן, בשכבה זו ממומשות קריאות ה-Webserver דרכו מתבצעת התקשורת בין Python לסימולטור. באיור 4: שכבת המנוע מתוארות המחלקות הרלוונטיות לשכבה זו.



איור 4: שכבת המנוע

מחלקת BP_SoundObjectBase

מחלקה אבסטרקטית שמייצגת אובייקט סאונד. ממחלקה זו יורשים אובייקטי המיקרופון ומקור הקול והיא מספקת את התשתית הפונקציונלית הרלוונטית לשניהם. מחלקה זו מחזיקה גם את הרכיב האחראי לתנועה האורביטלית ואת ממשקי ה-GUI הרלוונטיים לפונקציונליות שלה.

מחלקת BP_SOMActorOrbit

המחלקה האחראית על התנועה האורביטלית של אובייקט הסאונד. מופיעה כרכיב במחלקת BP_SoundObjectBase אשר דרכה ניתן לגשת לפונקציונליות של המחלקה.

מחלקת BP_SoundSource

המחלקה המייצגת מקור קול. יורשת מ-BP_SoundObjectBase ומוסיפה את הפונקציונליות הדרושה למקור קול. מחזיקה גם את ממשקי ה-GUI שלה. הסאונד הנפלט מאובייקט של המחלקה נפלט בצורת ספירה (Sphere) ולכן אין אפשרות לשנות את האוריינטציה המרחבית של האובייקט.

מחלקת BP_Microphone

המחלקה המייצגת מיקרופון. יורשת מ-BP_SoundObjectBase ומוסיפה את הפונקציונליות הדרושה למיקרופון. מחזיקה גם את ממשקי ה-GUI שלה. היות וקליטת הסאונד מתבצעת בשני ערוצים תוך שימוש

בHRTF (Head Related Transfer Function), שינוי האוריינטציה של המיקרופון שקולה לסיבוב הראש של המאזין. לכן, בניגוד למחלקת BP_SoundSource, אוריינטציית המיקרופון ניתנת לשינוי.

מחלקת BP_PlayerController

למחלקה זו מספר תפקידים עיקריים, ביניהם:

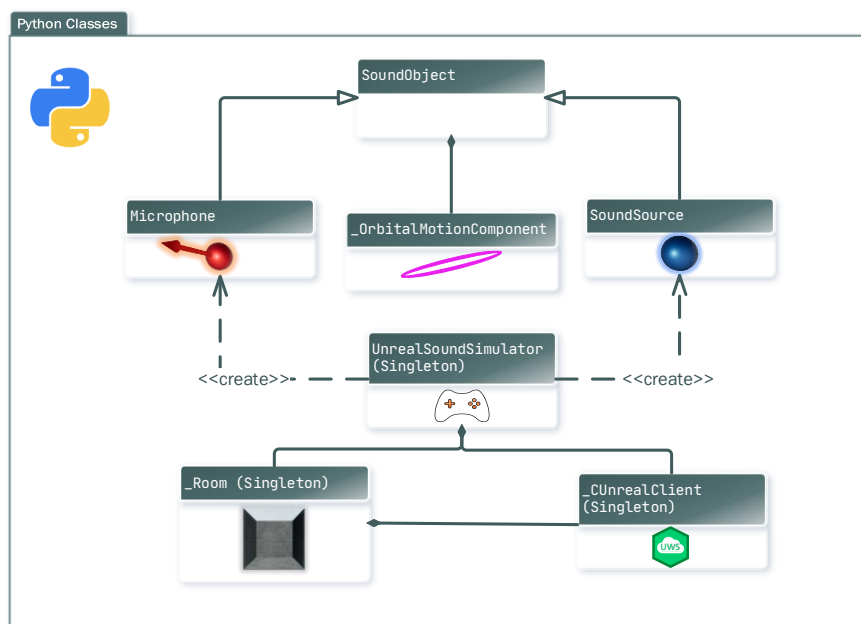
1. אתחול הסצנה והחדרים
2. הפעלת השרת ורישום הפקודות אותו הוא מזהה
3. מעקב אחר האובייקטים בסצנה ווידוא תנאים לסימולציה
4. יצירת GUI וקשירת האלמנטים השונים לפונקציונליות הנדרשת.
5. יצירת מחלקת SimulationAction האחראית על ביצוע הסימולציה

מחלקת SimulationAction

המחלקה המבצעת בפועל את הסימולציה. אחראית לסדר ביצוע פעולות הסימולציה תוך שמירה על סינכרון בין איטרציה לאיטרציה.

4.2. שכבת פייתון – תפקידים ומחלקות



כאמור, שכבת הפייתון אינה מבצעת אף לוגיקה ותפקידה היחיד הינו לתפעל את הסימולטור באמצעות קוד. באיור 5: שכבת פייתון ניתן לראות את מבנה השכבה.



איור 5: שכבת פייתון

בטבלה 1: מחלקות שקולות, מתוארים קשרי השקילות בין המחלקות בפייתון למחלקות במנוע, כאשר בשקילות הכוונה היא שתקשורת עם אובייקט מסויים מתבצעת דרך המחלקה השקולה בפייתון

טבלה 1: מחלקות שקולות

	
BP_SoundObjectBase	SoundObject
BP_SOMActorOrbit	_OrbitalMotionComponent
BP_SoundSource	SoundSource
BP_Microphone	Microphone
BP_PlayerController	UnrealSoundSimulator

מחלקת UnrealSoundSimulator._CUnrealClient

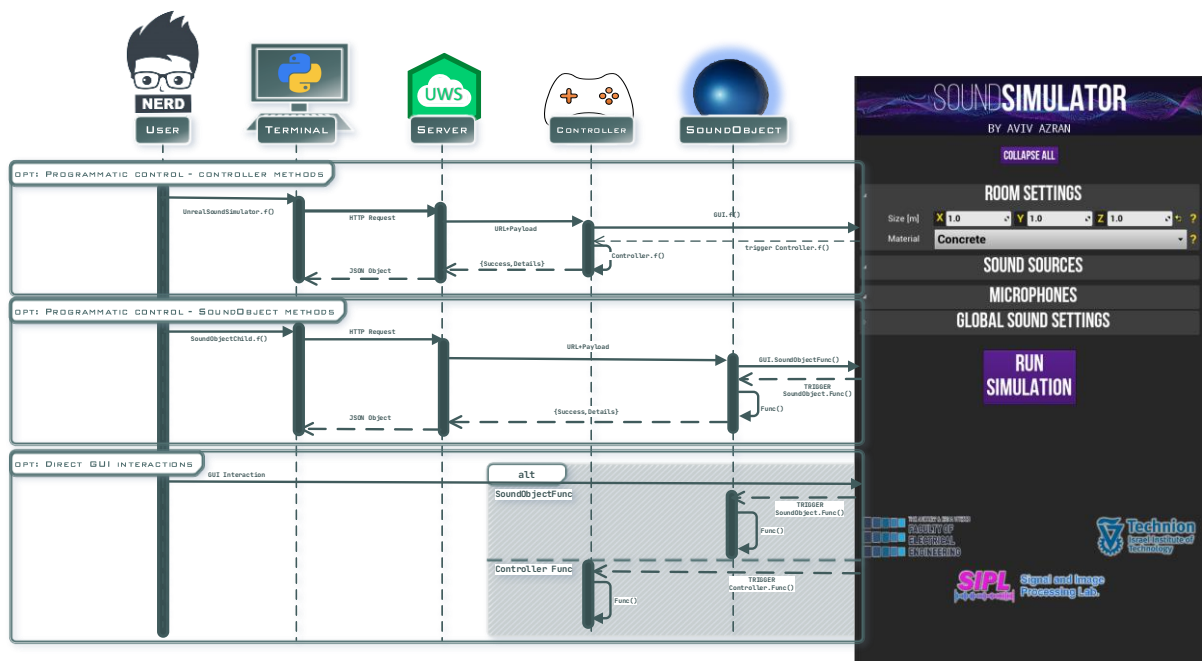
מחלקה זו הינה סינגלטון אחראית על התקשורת עם הסימולטור. היא מקבלת את הפקודה המבוקשת והפרמטרים שלה ושולחת Http Request מתאימה. מחזירה את תשובת הסימולטור.

מחלקת UnrealSoundSimulator._Room

מחלקה זו הינה סינגלטון ואחראית על הפעולות הקשורות לחדר עצמו – שינוי מימדי החדר והחומר ממנו הוא עשוי. כמו כן יש לה מתודות get לצורך קבלת מצב החדר הנוכחי והחומרים מהם אפשר לבחור.

4.3 התקשורת עם הסימולטור

הסימולטור ניתן לשליטה בשני דרכים – דרך GUI המובנה ודרך ה-API בפייתון. באיור 6 : Simulator Sequence Diagram מתואר מהלך התקשורת בין האובייקטים בסימולטור בהתאם לאופן בו בוצעה ההתקשרות.



איור 6 : Simulator Sequence Diagram

5. ממשק משתמש ופקודות סקריפט

כאמור, הסימולטור ניתן לשליטה בשני דרכים – דרך GUI המובנה ודרך ה-API בפיתון. בפרק זה אסקור את ה-GUI על אפשרויותיו השונות כאשר לכל אפשרות מצורף הסבר קצר על הפונקציונליות שלה בנוסף לפקודות הפיתון הקשורות אליה. נפרט גם על ה imports הדרושים וכיצד לייצר ולפנות לאובייקטים דרך פיתון. באיור 7: סימולטור סאונד - מסך ראשי ניתן לראות כיצד נראה הסימולטור והאובייקטים השונים הקיימים בו.



איור 7: סימולטור סאונד - מסך ראשי

5.1 תחילת עבודה עם פיתון

כדי לעבוד עם חבילת הפיתון, ראשית יש להתקין Python 3.8. מומלץ להתקין את הפצת פיתון של אנקונדה. לאחר מכן, יש להתקין את חבילת requests ע"י פתיחת cmd כ-Administrator והרצת הפקודה 'pip install requests'. לאחר יצירת קובץ סקריפט ופתיחתו לעריכה באמצעות Python IDE, נפעיל את הסימולטור ונרשום.

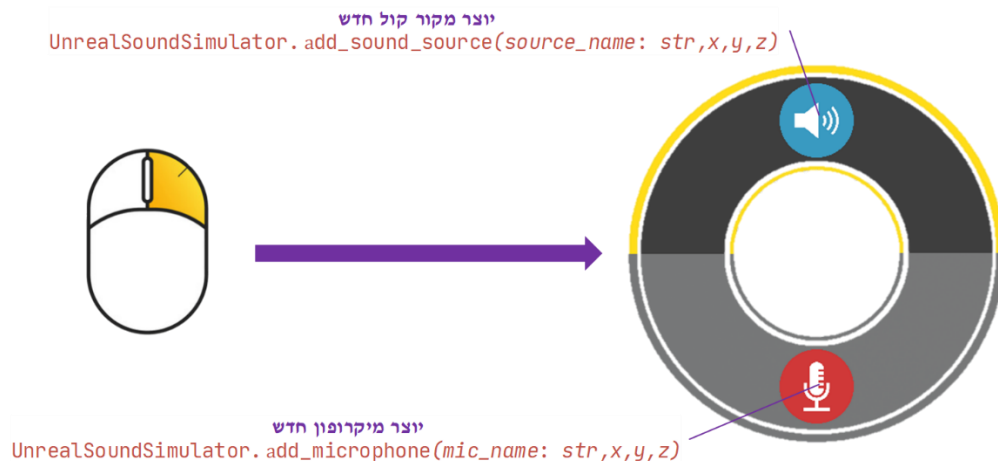
```
from CUnrealObjects import UnrealSoundSimulator
ue = UnrealSoundSimulator()
```

עתה ניתן להשתמש בפקודות המפורטות בפרק. אוסף הפקודות המלא מפורט בנספח - רשימת פקודות

Python API

5.2. הוספת מקורות קול ומיקרופונים

לאחר שהסימולטור נטען, קליק ימני על העכבר יפתח את התפריט הרדיאלי כפי שנראה באיור 8: תפריט רדיאלי להוספת אובייקט סאונד יש לבחור את סוג האובייקט אותו אנו רוצים להוסיף ובתפריט שנפתח (איור 9: תפריט בחירת מאפייני אובייקט סאונד) יש לתת לו שם משמעותי כלשהו באנגלית ולבחור מיקום התחלתי לאובייקט. יש לשים לב שלא יכולים להיות בסצנה שני אובייקטים בעלי שם זהה או ללא שם. לחיצה על כפתור האישור תסגור את חלון ההוספה והאובייקט החדש ייוצר וימוקם בסצנה. בנוסף, בהתאם לאובייקט, יתווספו פרמטרי השליטה לתפריט השליטה מצד ימין.



איור 8: תפריט רדיאלי להוספת אובייקט סאונד

The screenshot shows a dialog box titled 'ADD SOUND SOURCE' with a blue speaker icon. It has a 'NAME' field with a placeholder 'Must be unique'. Below it is a 'LOCATION [M]' section with three input fields for X, Y, and Z, each with a value of 0.0 and a small square icon to its right. At the bottom, there are two buttons: 'ADD' in blue and 'CANCEL' in red.

איור 9: תפריט בחירת מאפייני אובייקט סאונד

5.3. תנועה אורביטלית

תפריט זה (איור 10: תפריט תנועה אורביטלית) יהיה זמין כאשר נוצר אובייקט סאונד כלשהו וההגדרות שבו יחולו על כל אובייקט בנפרד.

איפשור תנועה אורביטלית
`SoundObject.orbit_start()`

מחזיר את ההגדרות הנוכחיות
`SoundObject.get_orbital_settings()`

מרכז ההקפה
`SoundObject.set_orbit_center(x, y, z)`

רדיוס ההקפה
`SoundObject.set_orbit_radius(x, y)`

מהירות זוויתית של ההקפה (מעלות לשניה)
`SoundObject.set_movement_speed(speed: float)`

מהירות זוויתית של ההקפה (מעלות לשניה)
`SoundObject.set_orbit_start_angle(angle: float)`

האוריינטציה של האובייקט על ההיקף
`SoundObject.set_default_rotation(x, y, z)`

מהירות זוויתית של האובייקט ביחס לציר הנבחר (מעלות לשניה)
`SoundObject.set_rotation_speed(vx, vy, vz)`

קובע האם סיבוב האובייקט מתבצע ביחס למסלול ההיקפי עליו הוא נע
`SoundObject.set_rotate_relative_to_orbit(bValue: bool)`

מדליק/מכבה תנועה היקפית כאשר בהדלקה נקודת ההתחלה מחושבת לפי הזמן שעבר מתחילת התנועה
`SoundObject.enable_orbital_motion(bValue: bool)`

הצגה של המסלול עליו האובייקט נע
`SoundObject.toggle_show_orbit(bValue: bool)`

ORBITAL MOVEMENT

- Orbit Center [m] X 1.255315 Y -0.218439 Z 0.0
- Orbit Radius X 2.110529 Y 1.670949 Z 0.0
- Movement Speed [deg/sec] 30.0
- Start Angle [deg] 0.0
- Orbit Rotation [deg] X 0.0 Y 0.0 Z 0.0
- Default Rotation [deg] X 0.0 Y 0.0 Z 0.0
- Rotation Speed [deg/sec] X 0.0 Y 0.0 Z 0.0
- Rotate relative to Orbit ☐
- Toggle Orbital Motion ON/OFF ☐
- Toggle Show Orbit ☒

איור 10: תפריט תנועה אורביטלית

5.4. מקור קול

איור 11: תפריט מקור קול נוצר בהוספת מקור קול והוא ספציפי למקור הקול ששמו מופיע ב- Current Sound Source.

מוחק את מקור הקול הנוכחי
`UnrealSoundSimulator.destroy_sound_source(source_name)`

המקור הנוכחי בו שולטים

פותח תיבת בחירת קובץ .wav שיקושר למקור
`SoundSource.set_wav_path(path: str)`

אם הקובץ נטען בהצלחה - מציג את המסלול לקובץ

קביעת הזמן מתחילת הסימולציה בשניות אחריו יתחיל המקור לנגן
`SoundSource.set_delay(delay: float)`

קביעת מכפיל הווליום של המקור בתחום [0.001,2] כאשר 1 הוא הווליום המקורי
`SoundSource.set_volume(volume: float)`

אם מקושר קובץ .wav - מנגן/מפסיק
`SoundSource.play_sound()`

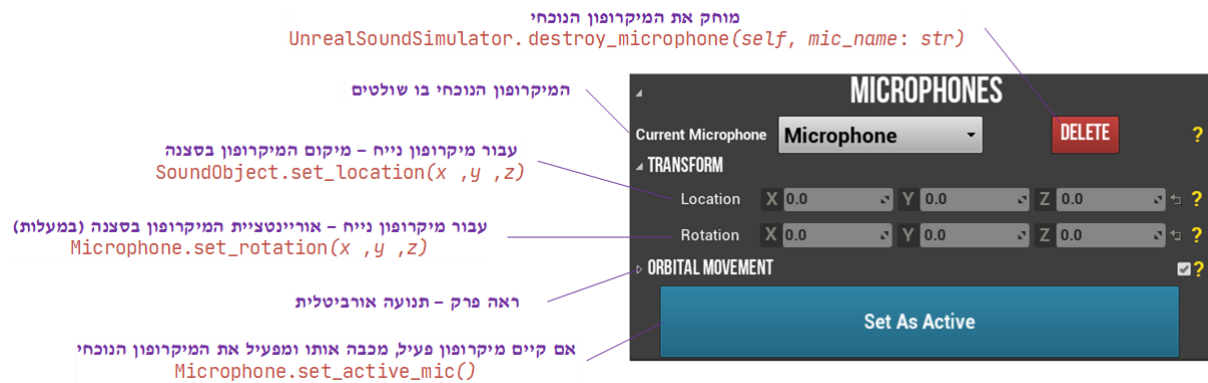
SOUND SOURCES

- Current Sound Source: Sound Source [DELETE]
- WAV
- SET WAV
- Loaded Sound Wav: Shows the path to the loaded sound wav if valid
- Delay Time: 0 +00:00
- Volume: 1.00
- PLAY

איור 11: תפריט מקור קול

5.5. מיקרופון

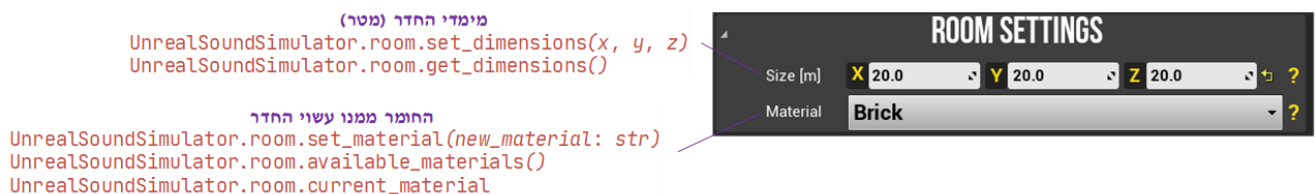
תפריט זה (איור 12: תפריט מיקרופון) נוצר בהוספת מיקרופון והוא ספציפי למיקרופון ששמו מופיע ב- Current Microphone.



איור 12: תפריט מיקרופון

5.6. חדר

תפריט זה (איור 13: תפריט הגדרות חדר) נוצר בהפעלת הסימולטור ודרכו ניתן לשלוט במימדי החדר והחומר ממנו הוא עשוי.



איור 13: תפריט הגדרות חדר

5.7. הגדרות סאונד גלובליות

תפריט הגדרות סאונד גלובליות קיים בתפריט מרגע הפעלת הסימולטור ודרכו ניתן לשלוט במאפיינים המפורטים באיור 14: תפריט הגדרות סאונד גלובליות.

מחזיר את הגדרות הנוכחיות
`UnrealSoundSimulator.get_global_sound_settings()`

רדיוס מסביב למקורות הקול אשר האובייקטים שבתוכו יחשבו בחישוב הסאונד הפיזיקלי
`UnrealSoundSimulator.set_sound_occlusion_source_radius(radius: float)`

בחירת מאפיינים להתחשב בהם בחישוב גלי הקול העוברים (ראה פרק - הגדרות סאונד)
`UnrealSoundSimulator.set_direct_occlusion_mode_none()`
`UnrealSoundSimulator.set_direct_occlusion_mode_no_transmission()`
`UnrealSoundSimulator.set_direct_occlusion_mode_frequency_dependent_transmission()`

שיטת חישוב הסאונד הפיזיקלי (ראה פרק - הגדרות סאונד)
`UnrealSoundSimulator.set_direct_occlusion_method_partial()`
`UnrealSoundSimulator.set_direct_occlusion_method_raycast()`

מאפשר חישוב פיזיקלי של ניות גל הקול כפונקציה של המרחק העובר לפי Inverse distance falloff
`UnrealSoundSimulator.set_physics_based_attenuation(bValue: bool)`

מאפשר חישוב הספיגה של האוויר בחישוב הסאונד
`UnrealSoundSimulator.set_physics_based_attenuation(bValue: bool)`

שינוי מהירות הקול בסצנה (מטר לשנייה)
`UnrealSoundSimulator.set_speed_of_sound(speed_of_sound: float)`

מחזיר הגדרות ברירת-מחדל
`UnrealSoundSimulator.set_default_sound_settings()`

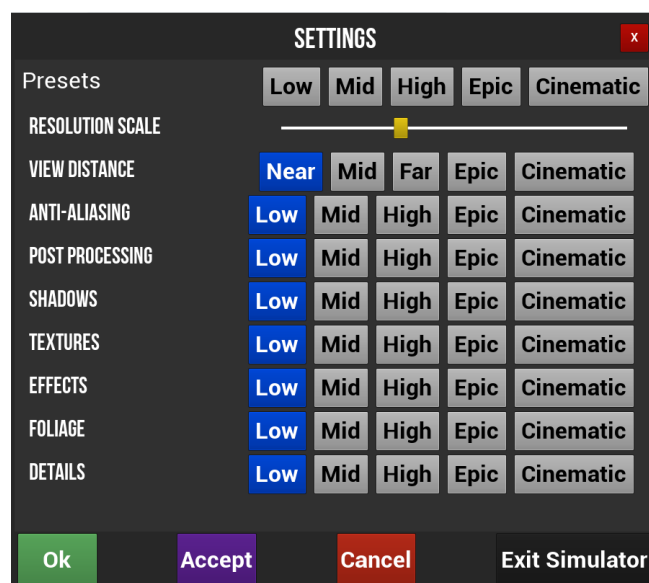
איור 14: תפריט הגדרות סאונד גלובליות

5.8. תצוגת מערכת צירים לוקאלית והרצת סימולציה

מאפשר או מכבה את תצוגת מערכת הצירים המקומית של מקורות הקול והמיקרופונים.
אם מתקיימים התנאים הנדרשים, מתחיל הרצה של סימולציה
`UnrealSoundSimulator.set_path_to_export(path: str)`
`UnrealSoundSimulator.set_saved_wav_file_name(filename: str)`
`UnrealSoundSimulator.start_simulation(save_path: str)`
`UnrealSoundSimulator.cancel_simulation()`

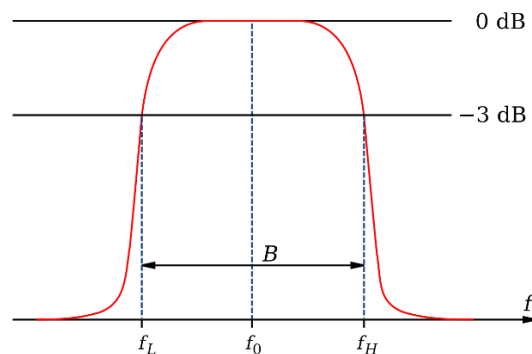
5.9. תפריט שינוי הגדרות גרפיקה

תפריט זה נוסף על מנת לאפשר הפניית משאבים נוספים לצורך חישובי הסאונד כאשר מריצים את הסימולטור על מכונות בעלות מאפייני חומרה שאינם High end. לחיצה על מקש Esc תפתח את התפריט. לחיצה על Accept תחיל את ההגדרות שנבחרו ללא סגירת התפריט. לחיצה על Ok תחיל את ההגדרות ותסגור את התפריט. לחיצה על Cancel – תסגור את התפריט ולא תחיל את ההגדרות. לחיצה על Exit Simulator תסגור את הסימולטור ותצא חזרה ל – Windows.



6. חומר אקוסטי

חומר אקוסטי מיוצג במנוע על ידי האובייקט Phonon Material, אותו ממקמים בסצנה כילד של אובייקט מסוג static mesh. האובייקט הנ"ל מגדיר למעשה את האפקטים שיופעלו על הסאונד אותו נשמע כאשר נאזין דרך מיקרופון מסויים. בהפעלת האפקטים, התוסף מתייחס (במידה ונגדיר זאת בממשק) בצורה נפרדת לשלושה תחומי תדירויות: Low, Mid ו-High, כאשר תחום תדירות מוגדר עפ"י התדירות המרכזית שלו. באיור 15: תדר מרכזי מוסבר המושג ובטבלה 2: תדרים מרכזיים בתוסף ניתן לראות את התדרים המרכזיים לפיהם מתבצעת החלוקה לתחומי התדר בתוסף.



איור 15: תדר מרכזי

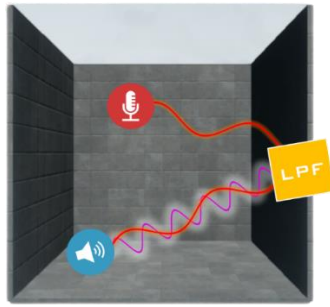
טבלה 2: תדרים מרכזיים בתוסף

Low Frequencies	Mid Frequencies	High Frequencies
800 Hz	4 KHz	15 KHz

עבור חומר מסויים ניתן להגדיר שלושה פרמטרים: בליעה (Absorption), העברה (Transmission) ופיזור (Scattering) כאשר שני הראשונים מוגדרים לכל תחום תדירויות בנפרד בעוד האחרון פועל על גל הקול כולו כיחידה. שלושת הפרמטרים מקבלים ערכים בתחום $[0,1]$ ובהמשך נביא דוגמא להשפעת ערכי הקצה עבור כל פרמטר.

6.1 בליעה (Absorption)

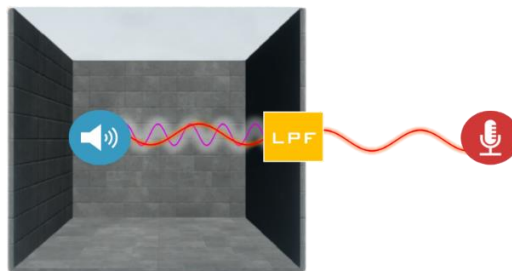
מאפשר להגדיר כמה סאונד ייבלע ע"י החומר לכל אחד מתחומי התדירויות. לדוגמא, הצבת הערך "1" לפרמטר High Frequency Absorption אומר שהחומר בולע את כל התדרים בתחום High מגל קול המגיע אליו. בפועל, על הגלים החוזרים מהחומר מופעל מסנן LPF (Low Pass Filter) כמתואר באיור 16 בליעה.



איור 16 בליעה

6.2. העברה (Transmission)

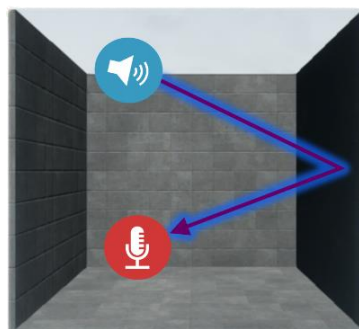
מאפשר להגדיר כמה סאונד יעבור דרך החומר לכל אחד מתחומי התדירויות. לדוגמא, הצבת הערך "0" לפרמטר High Frequency Absorption יגרום לכך שלא יעברו גלים בתחום התדרים הגבוהים דרך החומר. בפועל, על הגלים העוברים דרך החומר מופעל מסנן LPF כמתואר באיור 17: העברה.



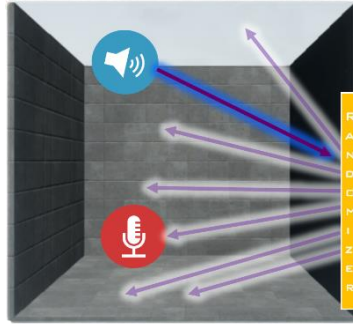
איור 17: העברה

6.3. פיזור (Scattering)

מאפשר להגדיר עד כמה "מחוספס" החומר האקוסטי בהקשר של החזרות. הצבת ערך "0" יגרום לחומר להתנהג כמו מראה כפי שנראה באיור 18: ערך פיזור '0', בעוד שהצבת ערך "1" יפזר את הסאונד בצורה רנדומלית כפי שנראה באיור 19: ערך פיזור '1'.



איור 18: ערך פיזור '0'



איור 19: ערך פיזור '1'

6.4. חומרים מובנים בסימולטור

בטבלה 3: חומרים מובנים בסימולטור מפורטים החומרים הקיימים בסימולטור וערכי הפרמטרים שלהם. כאשר עובדים עם הסימולטור דרך פייתון, ניתן לשנות את חומר החדר ע"י הפקודה

```
UnrealSoundSimulator.room.set_material(self, new_material: str)
```

כאשר new_material הוא string המכיל את אחד משמות החומרים כפי שמופיעים בטבלה הנ"ל.

טבלה 3: חומרים מובנים בסימולטור

		Concrete	Brick	Carpet	Wood	Rock	Plaster	Glass	Metal
Absorption	Low	0.05	0.03	0.24	0.11	0.13	0.12	0.06	0.2
	Mid	0.07	0.04	0.69	0.07	0.2	0.06	0.03	0.07
	High	0.08	0.07	0.73	0.06	0.24	0.04	0.02	0.06
Transmission	Low	0.015	0.015	0.02	0.07	0.015	0.056	0.06	0.2
	Mid	0.002	0.015	0.005	0.014	0.002	0.056	0.04	0.025
	High	0.001	0.015	0.003	0.005	0.001	0.004	0.011	0.01
Scattering		0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05

7. סיכום

7.1. מטרת הפרויקט והיעדים שהושגו

מטרת הפרויקט הייתה לייצר פלטפורמה שתאפשר למשתמש לדמות סצנה אקוסטית המורכבת מחדר העשוי מחומר מסוים ובעל מימדים כלשהם, אשר בו ממוקמים מקורות קול ומיקרופונים נייחים\ניידים. נדרשתי לספק אפשרות לשימוש בקבצי קול חיצוניים והרצת סימולציה מלאה שתפיק כפלט את הסאונד כפי שנשמע ע"י כל אחד מהמיקרופונים. ע"י שימוש במנוע Unreal כמנוע גרפי ותוסף Steam Audio לצורך חישובים אקוסטיים בצורה פיזיקלית, הצלחתי ליצור סימולטור בעל ממשק אינטואיטיבי העומד בדרישות הפרויקט. בנוסף, כתבתי API ב Python דרכו ניתן לתקשר עם הסימולטור ולבצע סימולציות מורכבות בצורה טורית. כך ניתן לבצע רצף סימולציות בצורה אוטומטית ולהעביר לניתוח את הפלטים שלהן בצורה ישירה וללא צורך לעבור בין ממשקים ולקטוע את רצף העבודה. בתכנון וכתיבת הסימולטור הושקעה מחשבה רבה תוך ניסיון לחוות את הסימולטור כמשתמש ולהבין מה הם הצרכים הנדרשים וכיצד לממש אותם בצורה שתהיה קלה לעבודה מצד אחד אך תאפשר יצירת סצנות מורכבות מצד שני. מטרת הפרויקט הושגו כולן.

7.2. קשיים בהם נתקלתי במהלך הפיתוח

במהלך העבודה על הפרויקט התמודדתי עם קשיים רבים, הן עם נושא הפרויקט עצמו והן בפיתוח. הקושי עם נושא הפרויקט היה העובדה שכל מה שקשור לסאונד, אם במציאות ואם במנוע היה זר לי לחלוטין לפני כן. לא הכרתי מונחים ונאלצתי תחילה ללמוד המון כדי להבין כיצד לגשת למימוש. בצד התכנותי, הקשיים בהם נתקלתי היו בעיקר בשכבת המנוע. היה קושי גדול בבחירת התוסף המתאים עקב העובדה שלא ניתן להשתמש ביותר מתוסף אחד בו-זמנית, מה שגרם לכך שמהתוסף הנבחר נדרש כי יכיל בתוכו את כל הפונקציונליות הנדרשת לסימולטור. בנוסף, התוסף והמנוע הציבו מגבלות מסויימות על מימוש פונקציות מסויימות, מה שדרש ממני לגלות יצירתיות תכנותית, כמו גם בעיצוב הסצנה ותכנון המחלקות השונות. העיקרון המנחה ביצירת הסימולטור היה לאפשר הרחבה שלו בצורה פשוטה יחסית. עמידה בעיקרון זה דרשה תכנון מקדים של רוב המימושים במנוע, ללא יכולת לבדוק האם הם יעבדו. ברוב המקרים הדברים עבדו, אולם במקרים בהם הם לא עבדו, נדרש לרוב תכנון מחדש של כמה שכבות נוספות קדימה. קושי עיקרי היה ההתמודדות עם העובדה שUnreal הוא מנוע מקצועי המספק אוסף רחב מאוד של כלים ולא תמיד מסופקת איתם דוקומנטציה מספקת. בהמון מקרים נאלצתי להגדיר מאוד במדויק מה אני מנסה להשיג ואז לפרק את זה לאוסף בעיות קטנות יותר אותן קל יותר להבין כיצד לממש. אסכם את פרק הקשיים בכך שאומר שללא ספק הקושי הגדול ביותר היה לעשות פרויקט מסדר גודל כזה לבד.

במובנים רבים, הפרויקט הזה הוא לא פחות ממוצר וכאשר מפתחים מוצר, יש צוות של אנשים בעלי ידע בתחום הרלוונטי עליו הם אמונים. בפיתוח הפרויקט הייתי צוות התכנות, צוות העיצוב, צוות המנוע, צוות הסאונד וצוות ממשק המשתמש וזה הצריך ממני התמקצעות בכל אחד מהתחומים בזמן מאוד קצר, בנוסף ללמידת האופן בו הסאונד מופק במנוע שכשלעצמה הייתה תהליך מורכב.

7.3. מה רכשתי מהפרויקט

כפי שצינתי בחלק העוסק בקשיים, פיתחתי את הפרויקט לבד. לכן, בראש ובראשונה, רכשתי תחושת מסוגלות עצומה. רכשתי רקע תאורטי בנושאים הקשורים לסאונד פיזיקלי וכיצד מייצגים אותו מתמטית ותכנותית. פיתחתי ראייה מרחבית הדרושה לפיתוח מוצר שלם משלב התכנון ועד שלב ההפצה ותירגלתי עקרונות תכנות שלמדתי במהלך התואר בקורסים כמו מת"מ ותכנות מונחה עצמים. למדתי כיצד להשתמש בחשיבה יצירתית ובתכנון נכון כדי לעקוף מגבלות תוכנה ולהשיג את הפונקציונליות בה אני מעוניין. למדתי כיצד לייעל קוד כך שיעמוד בדרישות סיבוכיות ומקום ושיפרתי מאוד את יכולות התכנות שלי בPython וC++. לסיכום, הפרויקט היווה עבורי אתגר מאוד גדול, הן מנטלית והן הנדסית ובאופן מסוים הוא מהווה עבורי את גולת הכותרת של הלימודים שלי – יצירה הנדסית יחידה במינה שאני מקווה שתעזור לקדם ולפתח את הקהילה האקדמית עבודה כלי כזה יהיה רלוונטי.

8. נספחים

8.1. נספח - הרחבת הסימולטור ושינוי הגדרות פנימיות

במקרים מסויימים, המשתמש עלול לא להסתפק בפונקציונליות שמציע הסימולטור וירצה לפתוח את קוד המקור במנוע לצורך הרחבה של הפונקציונליות או שינוי פרמטרים מסויימים. למה בכלל צריך לפתוח את המנוע עצמו ולא ניתן לגשת לכל פרמטר ישר מהסימולטור?

המנוע, בעודו מאפשר המון חופש פעולה, מציב הגבלות מסוימות מבחינת גישה לפרמטרים מסויימים ושינויים לאחר תהליך האריזה. זאת בשל שיקולים כגון הגנה על הקוד, כמו גם מהסיבה שהמנוע בראש ובראשונה מיועד ליצירת משחקי מחשב\סרטים ואמצעי בידור נוספים ואילו בפרויקט זה אנו מנצלים את יכולותיו לצורך ביצוע סימולציות. מגבלות אלו ומגבלות נוספות המוכתבות ע"י תוסף Steam Audio הן הסיבה בגינה לעיתים נאלץ לפתוח את קוד המקור, לבצע שינויים ולארוז מחדש את הפרויקט. על מנת להקל על המשתמש הרוצה להרחיב את הפרויקט, ניסיתי לשים עצמי במקומו וחשבתי איזה פעולות ירצה המשתמש לבצע או להרחיב.

בפרק זה ריכזתי הסברים צעד-אחר-צעד כיצד לבצע במנוע את הפעולות עליהן הצלחתי לחשוב כחיוניות.

אזהרה:

הלוגיקה הפנימית של הסימולטור מורכבת וקל מאוד לבצע שינוי כזה שיגרם לרצף של תקלות אחריו הסימולטור לא יעבוד כמצופה, במיוחד עבור מי שאינו בעל ניסיון עם המנוע ולא מכיר את התשתית התכנותית של הסימולטור. לכן, לפני כל שינוי מומלץ לגבות את הפרויקט כיוון שמרגע השמירה, לא ניתן לבטל בצורה אוטומטית את השינויים. בנוסף, מומלץ שלא לנסות לבצע שינויים למעט אלו עליהם אסביר בפרק זה. במידה ועולה צורך לפונקציונליות נוספת מעבר למפורט כאן, ניתן ליצור איתי קשר במייל aviv.zeus@gmail.com על מנת לקבל הדרכה בנושא.



הערה:

1. את שלבים 1+2 של Packaging יש לבצע בכל פעם ששינינו משהו שקשור לאקוסטיקה כולל הוספת חומר אקוסטי חדש או שינוי הגדרות סאונד פנימיות.
2. את שלבי ההתקנה יש לבצע בסדר בו הם מופיעים

התקנת Intel Embree

1. יש ללחוץ [כאן](#) ולהתקין את הקובץ.

2. יש לבצע את השלבים הבאים במדויק:

Embree supports using the Intel® Threading Building Blocks (TBB) as the tasking system. For performance and flexibility reasons we recommend to use Embree with the Intel® Threading Building Blocks (TBB) and best also use TBB inside your application. Optionally you can disable TBB in Embree through the **EMBREE_TASKING_SYSTEM** CMake variable.

Embree will either find the Intel® Threading Building Blocks (TBB) installation that comes with the Intel® Compiler, or you can install the binary distribution of TBB directly from www.threadingbuildingblocks.org into a folder named **tbb** into your Embree root directory. You also have to make sure that the libraries **tbb.dll** and **tbb_malloc.dll** can be found when executing your Embree applications, e.g. by putting the path to these libraries into your **PATH** environment variable. Embree supports the Intel® SPMD Program Compiler (ISPC), which allows straightforward parallelization of an entire renderer. When installing ISPC, make sure to download an ISPC version from ispc.github.io that is compatible with your Visual Studio version. After installation, put the path to **ispc.exe** permanently into your **PATH** environment variable

התקנת המנוע

1. יש להתקין גרסה 4.25.3 לפי ההוראות [כאן](#).

2. לאחר התקנת המנוע יש להתקין את התוספים הדרושים ע"י לחיצה על הלינק המתאים בטבלה

4: תוספים למנוע.

טבלה 4: תוספים למנוע

Plugin	Mandatory	Paid/Free
Web Server for Unreal	Yes	Paid
Graph Formatter	No	Free
SSZCode	Yes	Free
Easy File Dialog	Yes	Free
Auto Size Comments	No but very recommended	Free
Asset Assistant	Yes	Free
Electronic Nodes	No but very recommended	Paid

3. יש להתקין Visual Studio 2019 עפ"י ההוראות [כאן](#).

הורדת ופתיחת הפרויקט

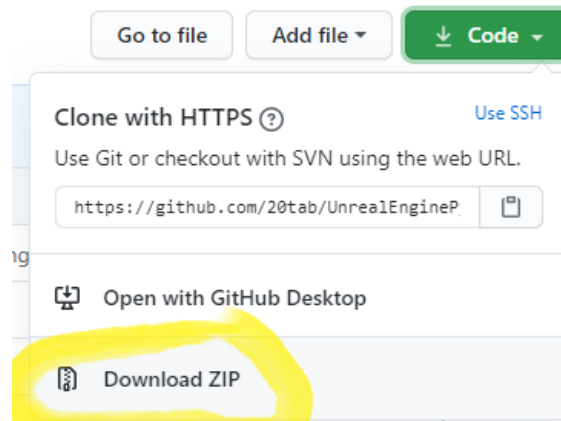
את הפרויקט ניתן להשיג ממעבדת SIPL בפקולטה להנדסת חשמל בטכניון או מעמוד הפרויקט ב-GitHub

בלינק <https://github.com/avivazran/SOUNDSIMULATOR>

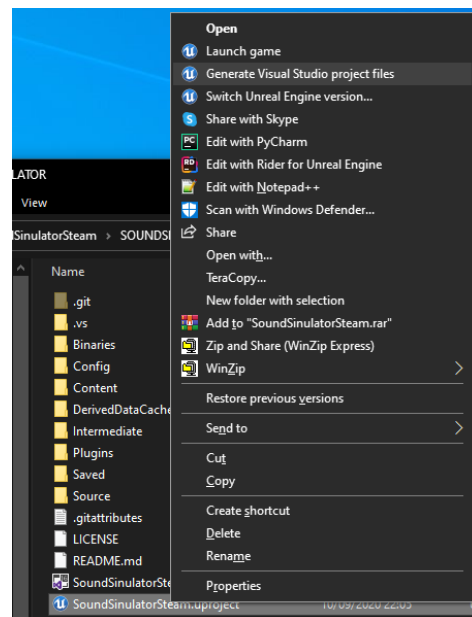
אתר הפרויקט: [/https://avivazran.github.io/SOUNDSIMULATOR](https://avivazran.github.io/SOUNDSIMULATOR)

כדי להוריד את הפרויקט מ-GitHub:

- יש להיכנס לקישור לעיל וללחוץ על Code->Download Zip. יש לשים לב שלאחר ביצוע Build הפרויקט שוקל כ- 15Gb לכן יש לשמור את הקובץ לזכרון בו יש מספיק מקום.

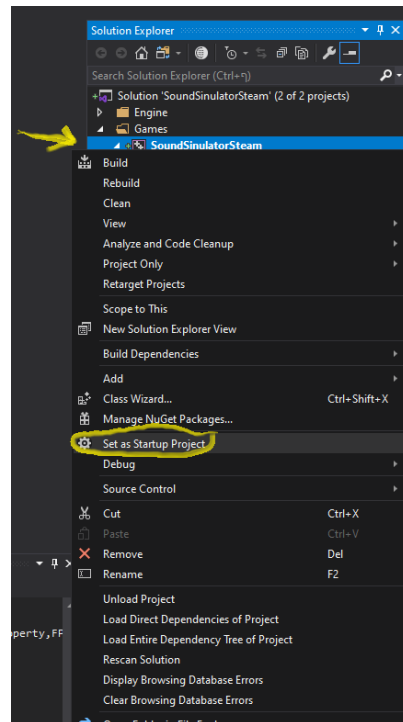


- בשלב זה יש לוודא שהתקנת Intel Embree, Unreal Engine 4.25.3, Visual Studio. לאחר הודוא יש לוודא שכל התוספים המפורטים בטבלה 4: תוספים למנוע מותקנים אחרת תהליך ה-Build ייכשל.
- לאחר ההורדת וביצוע Extract לקובץ Zip, קליק ימני על SoundSimulatorSteam.uproject ובתפריט שנפתח יש ללחוץ על Generate Visual Studio Project Files ולהמתין לסיום התהליך.



4. יש לפתוח את הקובץ SoundSimulatorSteam.sln ולסמן את הפרויקט כ- Startup Project כמתואר

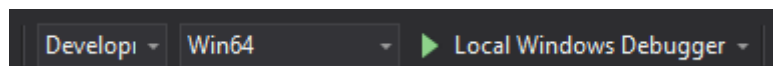
בתמונה להלן



5. יש לבנות את הפרויקט ע"י לחיצה על Ctrl+B ולהמתין לסיום.

6. יש לוודא את ההגדרות בתפריט העליון ב- Visual Studio כפי שמופיעות בתמונה להלן (ההגדרה

השמאלית היא (Development Editor)

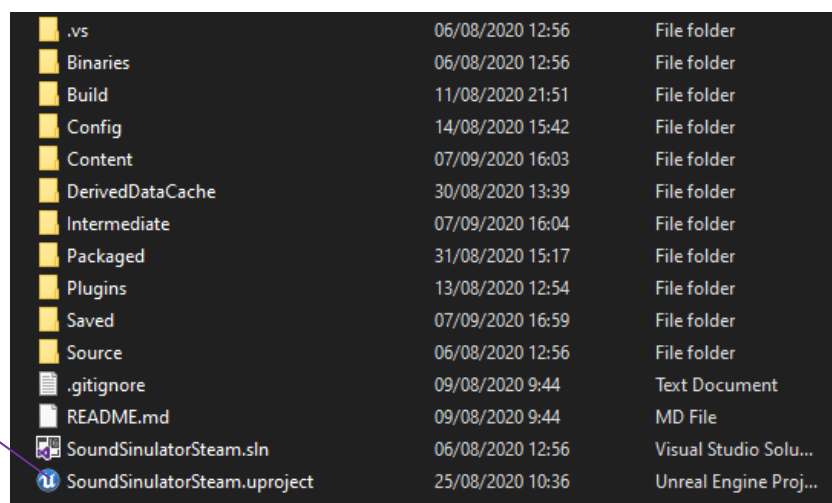


7. ניתן לפתוח את הפרויקט ע"י לחיצה על Local Windows Debugger או ע"י סגירת Visual Studio,

כניסה לתיקיה אליה הורדתם את הפרויקט ופתיחת את הקובץ כמתואר באיור 20: פתיחת

הפרויקט לעריכה.

פתיחת הפרויקט
לעריכה



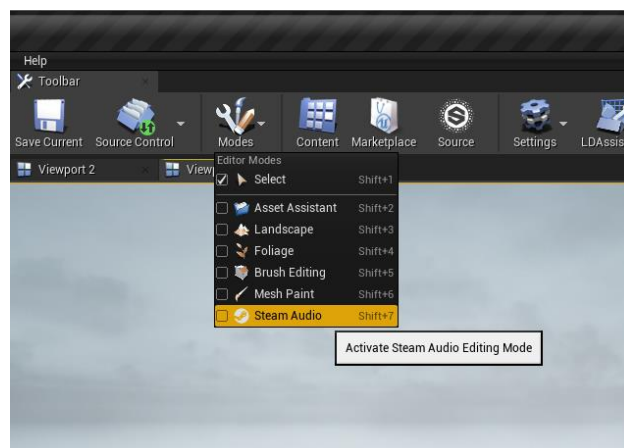
איור 20: פתיחת הפרויקט לעריכה

8. בכל פעם שפותחים את הפרויקט, יש להיכנס לUI->content browser ולפתוח את BP_PlayerController ולבצע קומפילציה, שמירה, קומפילציה, שמירה.

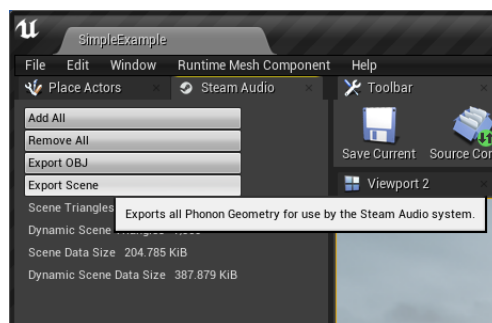
Packaging

Packaging הוא התהליך שבו אנו יוצרים את הסימולטור כ-standalone, כלומר ניתן להפעלה ללא צורך במנוע. נרצה לבצע packaging לאחר שהרחבנו את הפרויקט, הוספנו חומר אקוסטי חדש או שינינו הגדרות סאונד פנימיות.

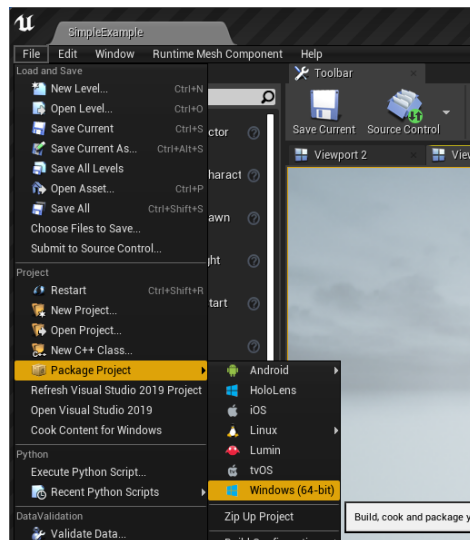
1. בסרגל הviewport, נלחץ על Steam Audio Modes->



2. בסרגל הנפתח בצד שמאל נלחץ על Export Scene ולאחר מכן נשמור.



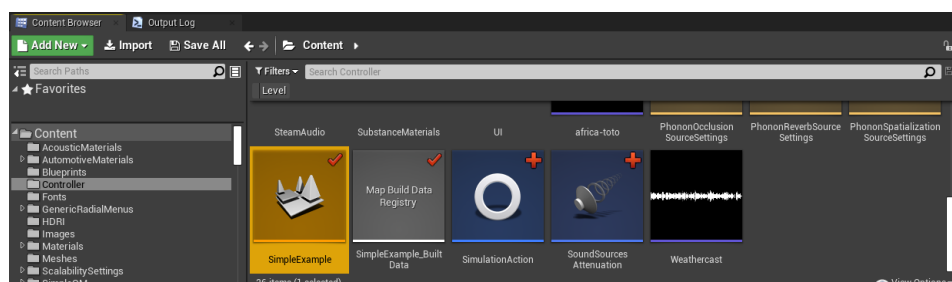
3. נלחץ על Windows(64bit) Project->Package File->



4. בחלון שיפתח נבחר את המיקום בו תיווצר הגרסה הארוזה ונלחץ Select Folder.

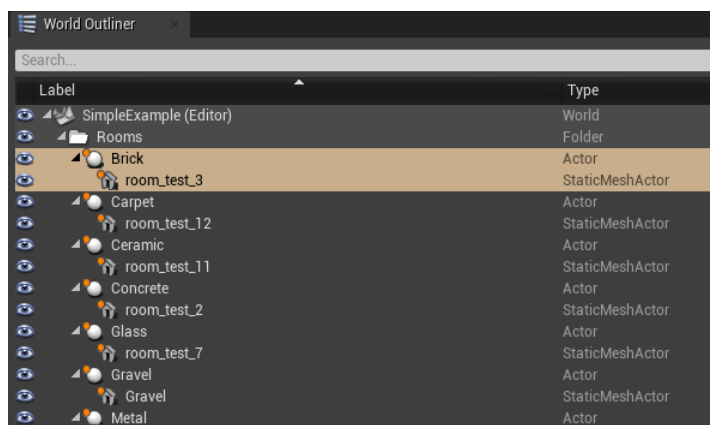
הוספת חומר אקוסטי חדש

1. בפרוייקט, נפתח את המפה הראשית ע"י לחיצה כפולה על SimpleExample



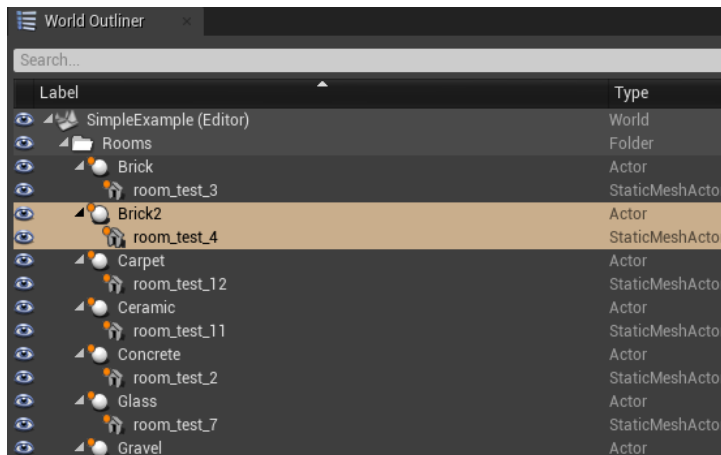
2. מצד ימין, בתפריט World Outliner נחזיק Ctrl ונסמן את אחד האובייקטים מסוג Actor, יחד עם

האובייקט המקושר אליו.

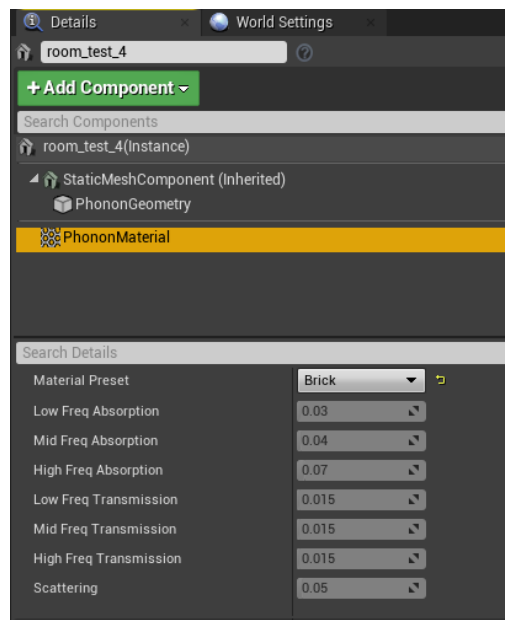


3. נלחץ Ctrl+C ו-Ctrl+V. ייווצר עותק של האובייקטים המסומנים. נשנה את השם של אובייקט האב

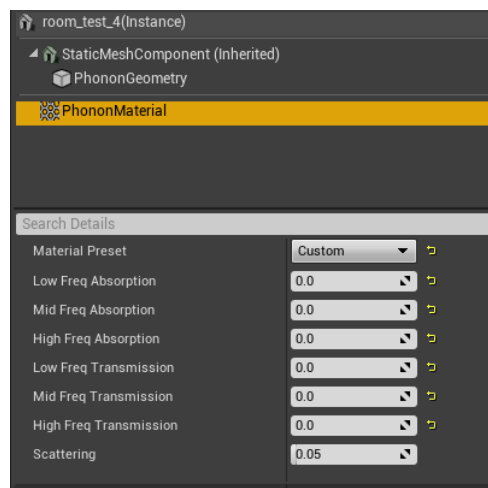
לשם משמעותי (עדיף שם החומר).



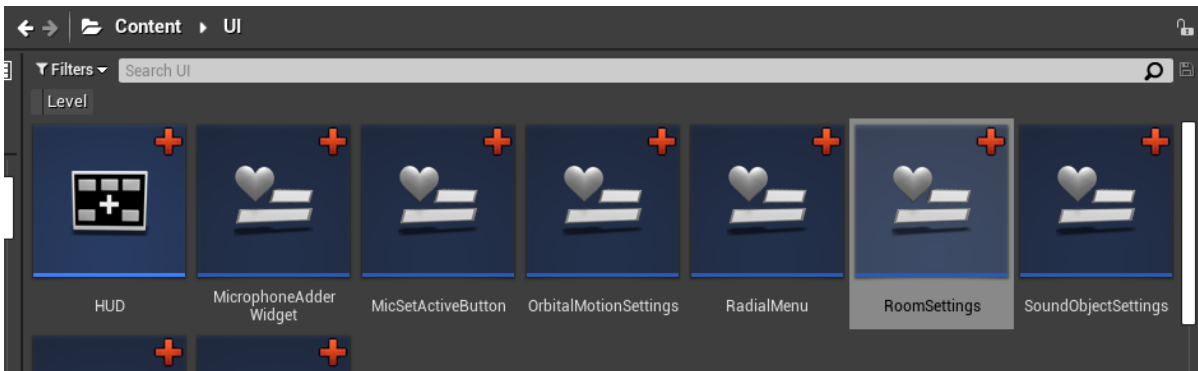
4. נסמן את אובייקט הילד החדש שיצרנו. בתפריט למטה, מתחת ל-World Outliner, בטאב Details, נסמן את הקומפוננט PhononMaterial.



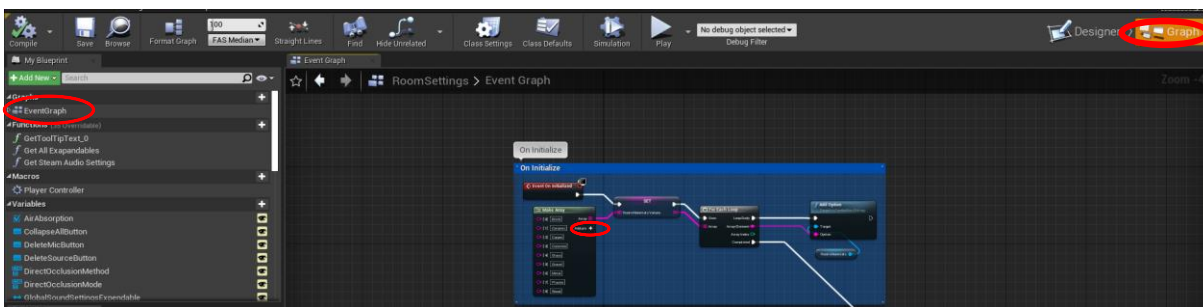
5. נפתח את Material Preset ונבחר באפשרות Custom, ונשנה את הפרמטרים של החומר כפי שנרצה אותם. לשמירה – Ctrl+S



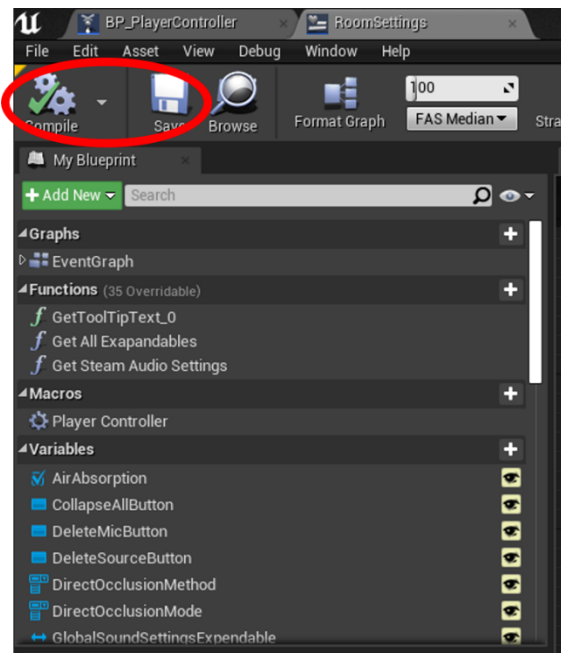
6. נפתח את RoomSettings



7. מצד ימין למעלה, נלחץ Graph. נפתח את הEvent Graph ונלחץ על Add Pin בבולוק MakeArray.
בכניסה שנוספה, נכתוב במדויק את השם שנתנו לאובייקט בסעיף 3.

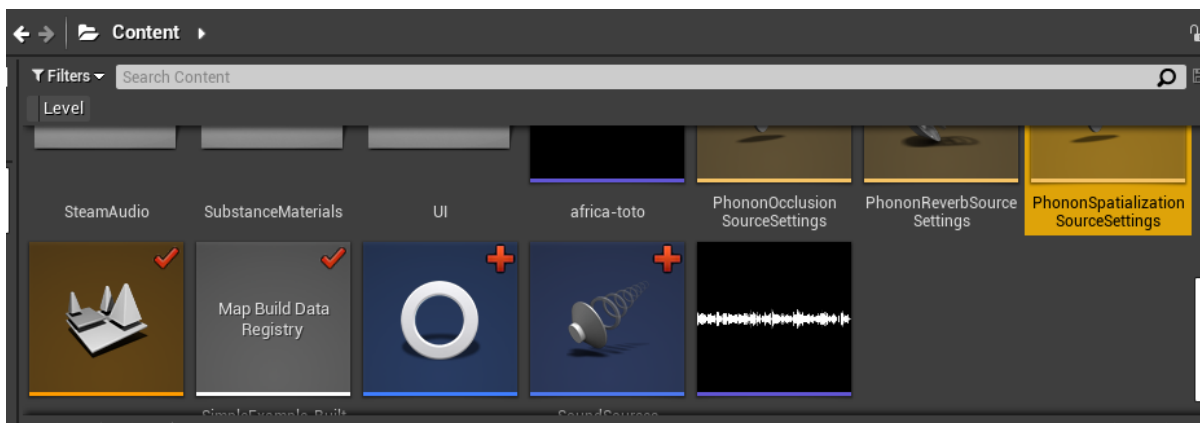


8. נבצע קומפילציה ונשמור.

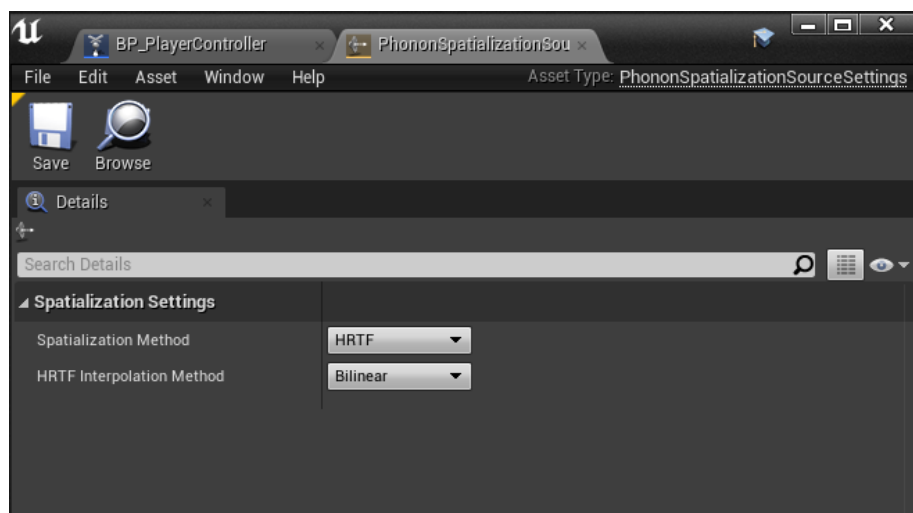


שינוי Spatialization Method ו-HRTF Interpolation method

1. נפתח את PhononSpatializationSourceSettings

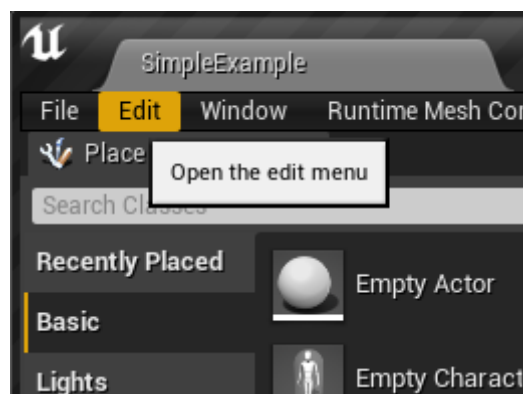


2. בחלון שנפתח נבחר את הערכים הרצויים ובסיום נשמור ונסגור את החלון.

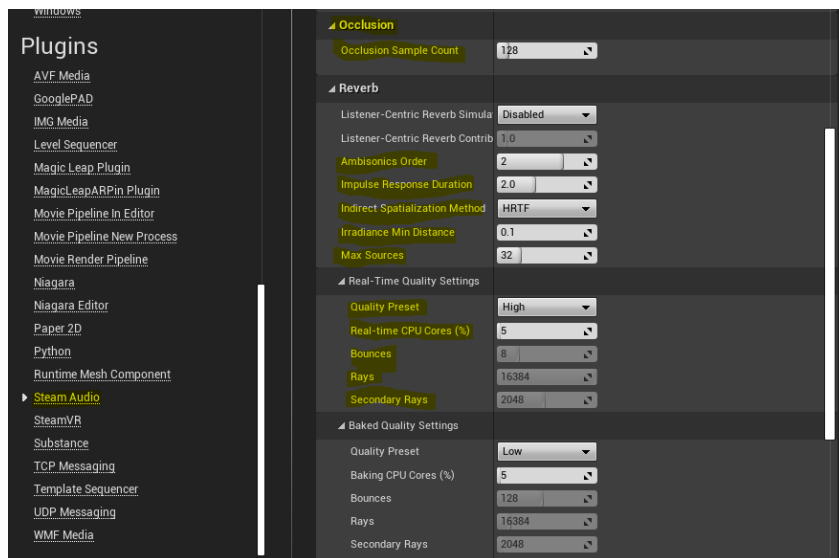


שינוי ההגדרות של תוסף Steam.

1. בסרגל הכלים העליון, נפתח את Edit ונלחץ על Project Settings.



2. נגלול בתפריט השמאלי למטה ל-Plugins ובנבחר ב-Steam Audio. האפשרויות ההמורקות הן האפשרויות אותן מותר לשנות.



3. בסיום, נלחץ על X ונשמור את הפרויקט.

8.2. נספח – קישורים שימושיים

בטבלה 5: קישורים שימושיים מופיעים קישורים שימושיים לצרכי שימוש בפרויקט ועריכתו.

טבלה 5: קישורים שימושיים

קישור	תיאור
https://github.com/avivazran/SOUNDSIMULATOR	עמוד Github של הפרויקט
https://avivazran.github.io/SOUNDSIMULATOR	אתר הפרויקט
https://www.unrealengine.com/en-US/get-now/agnostic	דף הבית של מנוע Unreal
https://docs.unrealengine.com/en-US/index.html	דוקומנטציה של מנוע Unreal
https://valvesoftware.github.io/steam-audio	דף הבית של Steam Audio
https://valvesoftware.github.io/steam-audio/doc/phonon_unreal.html	דוקומנטציה של Steam Audio ל- Unreal Engine

8.3. נספח - רשימת פקודות Python API

```
class UnrealSoundSimulator:

    @property
    room()

    # Sound Sources
    add_sound_source( source_name: str, x=0, y=0, z=0)
    destroy_sound_source( source_name: str)
    destroy_sound_source( sound_source_object: SoundSource)
    play_all_sounds()

    # Microphones
    add_microphone( microphone_name: str, x=0, y=0, z=0)
    destroy_microphone( mic_name: str)
    destroy_microphone( mic_object: Microphone)
    set_mic_as_current_listener( mic_name: str)

    # Simulation
    start_simulation( save_path: str)
    cancel_simulation()
    set_path_to_export( path: str)
    set_saved_wav_file_name( filename: str)

    # Global Sound Settings
    set_default_sound_settings()
    set_speed_of_sound( speed_of_sound: float)
    set_physics_based_attenuation( bValue: bool)
    set_air_absorption( bValue: bool)
    set_sound_occlusion_source_radius( radius: float)
    set_direct_occlusion_mode_no_transmission()
    set_direct_occlusion_mode_none()
    set_direct_occlusion_mode_frequency_independent_transmission()
    set_direct_occlusion_mode_frequency_dependent_transmission()
    set_direct_occlusion_method_partial()
    set_direct_occlusion_method_raycast()
    get_global_sound_settings()

    # Reset and Exit
    reset()
    exit_simulator()
```

```

class UnrealSoundSimulator.room()

set_material( new_material: str)

@property
current_material()

available_materials()

get_dimensions()

set_dimensions(x, y, z)

```

```

class SoundObject:

@property
name()

set_location( x, y, z)

get_location()

init_orbital_movement()

set_orbit_rotation( x, y, z)

set_orbit_center(x, y, z)

set_orbit_radius(x, y)

set_orbit_start_angle(angle: float)

set_movement_speed(speed: float)

toggle_show_orbit(bValue: bool)

set_default_rotation(x, y, z)

set_rotation_speed(vx, vy, vz)

set_rotate_relative_to_orbit(bValue: bool)

enable_orbital_motion(bValue: bool)

get_orbital_settings()

```

```

class SoundSource(SoundObject):

set_wav_path( path: str)

play_sound()

set_delay( delay: float)

get_soundwave_details()

set_volume_multiplier( mult: float)

```

```

class Microphone(SoundObject):

set_rotation(x, y, z)

get_rotation()

```