UNIVERSITY OF MIAMI

3D MESH RECOGNITION AS AN INPUT TO A PROCEDURAL AUDIO ENGINE IN GAMES

By

Ashay Dave

A PROJECT

Submitted to the Frost School
of Music, University of Miami
in partial fulfillment of the requirements
for the degree of Master of Science
Coral Gables, Florida

May 2024

UNIVERSITY OF MIAMI

A project submitted in partial fulfillment of
the requirements for the degree of
Master of Science

3D MESH RECOGNITION AS AN INPUT TO A PROCEDURAL AUDIO ENGINE IN GAMES

Ashay Dave

Approved:

DAVE, ASHAY (M.S., Music Engineering Technology)

Use This One
to Indicate a Line-broken
Version of Your Title (May 2024)

Abstract of a dissertation at the University of Miami.

Dissertation supervised by Professor Tom Collins.
No. of pages in text. (26)

The dissertation abstract goes here.

Dedication

Acknowledgments

Acknowledgments go here.

Ashay Dave

University of Miami

May 2024

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Ever since the first video game was developed, the video game industry has witnessed a remarkable surge in the pursuit of graphical fidelity. Advancements in hardware capabilities have allowed game developers to create visually stunning and realistic virtual environments. These improvements are evident in the intricate details of textures, realistic lighting effects, and lifelike character animations. Audio, until relatively recently, did not receive a proportionate level of attention.

Game audio is most commonly classified into the categories of speech, sound effect, and music [2]. Speech refers to the vocal from any character in the game, including the player and non-playable characters (NPCs). Sound effects are the non-periodic, non-musical elements that usually represent the sonic characteristics of real-world objects sounds, like wind, thunder, rain, etcetera. Sound effects can also be abstract and artificial sounding, like the sounds of the resident robots in *Stray (2022)*, the sound of a character jumping, zombies in *Dying Light (2015)*, or even the beep-bop sounds of a button being pressed. Lastly, game audio music can be categorised as soundtrack and music composition that often accompanies the narrative or theme of the video game.

Traditionally, audio in video games has been approached with a more static mindset, where pre-recorded soundtracks and sound effects were used to enhance the gaming experience. This approach, although effective in many cases, lacked the dynamic adaptability needed to match the increasingly sophisticated visual elements in modern games. As games became more complex and interactive, the static nature of traditional audio implementation started to reveal its limitations. The adoption of procedural sound synthesis presents an appealing alternative to enhance the realism of interactive scenes. [3].

The prevalent approach to incorporating music into games, as mentioned above, involves playing a pre-composed, linear piece of music during gameplay. In numerous games, the music is linked to the current level or game state. In the case of linearly composed music, the musical piece starts playing when the corresponding level is loaded. The music is often looped so it continues as long as the player is in the current active game state or level. Transitioning to a new level prompts an abrupt change or a swift crossfade into the music associated with the new level.

The evolution of adaptive music, a sophisticated system that tailors the musical score based on player interactions and in-game events, has been studied by Plut and Pasquier [4]. The orchestration dynamically adjusts to match mood, intensity, and pace, enhancing the emotional impact of key moments. Adaptive music transforms a static soundtrack into a responsive entity that mirrors the player's exploration of the virtual realm a video game offers.

Plut and Pasquier again discuss the state of the art in Generative Audio for video games in [5]. Generative audio using machine learning, neural networks has been a primary focus of research in the growing field of audio and AI.

As technology advances, the exploration of 3D objects within games attains prominence. These objects, representing characters, environments, and artifacts, are defined by their geometry, with the mesh— a network of interconnected vertices and faces forming the fundamental structure. In this context, MeshCNN [6], a significant 3D Object Mesh recognition model, that uses Convolutional Neural Networks (CNN) in PyTorch, is utilized. The network employs convolutions on edges and their incident triangles, with pooling facilitated by an edge collapse operation preserving surface topology. This task-driven pooling enhances feature extraction by learning to retain crucial edges and discard redundant ones. The effectiveness of this approach is demonstrated across various learning tasks applied to 3D meshes.

In this thesis, the integration of procedural audio and 3D object recognition is explored. Proposing a system that dynamically assigns sonic characteristics based on 3D Object Mesh classifications using a convolutional neural network - MeshCNN, integrated into the Unity Game Engine through Barracuda/Sentis and through the Open Neural Network Exchange file format. Through this exploration, we contribute to the theoretical foundations of procedural audio and generative music, utilizing a rather unexplored fusion of technologies to enhance user immersion in the dynamic world of video games.

1.1 Motivation

1.2 Research Objectives

CHAPTER 2

Background[*]

This section explores the history of audio and music in interactive games, and the fields of procedural, generative, and adaptive audio within interactive mediums. Additionally, some of the problems with current implementations of game audio systems is explored.

2.1 Procedural Audio In Games

Procedural audio is a dynamic approach to sound generation in which audio content is generated algorithmically in real-time, often based on parameters, rules, or algorithms, rather than relying on pre-recorded samples or fixed sound files. This technique allows for a high degree of flexibility and adaptability, making it well-suited for interactive and dynamic environments such as video games, simulations, and virtual reality experiences.

The key principle behind procedural audio is to create sound on-the-fly, responding to various parameters and events within the system. Unlike traditional methods of sound design that involve using pre-recorded samples, procedural audio allows for the generation of an infinite variety of sounds based on mathematical algorithms

[*]Footnote here.

or rules. This approach offers several advantages, including reduced storage requirements, increased adaptability, and the ability to generate unique, non-repetitive audio experiences.

The process of procedural audio involves the manipulation of audio synthesis parameters to achieve desired sonic characteristics. These parameters can include pitch, volume, frequency, modulation, filtering, and spatialization, among others. By dynamically adjusting these parameters in response to specific triggers or environmental changes, procedural audio systems can create a rich and immersive auditory experience that complements the visual elements of interactive media.

One common application of procedural audio is in the generation of environmental sounds. For example, instead of relying on a library of pre-recorded footsteps for a character in a game, procedural audio could simulate footsteps in real-time based on factors such as character movement speed, surface type, and environmental conditions. This not only reduces the need for a vast library of sound files but also ensures a more responsive and contextually relevant auditory experience for the user.

Procedural audio techniques can encompass various synthesis methods, including subtractive synthesis, granular synthesis, wave shaping, and physical modeling. These methods enable the creation of a wide range of sounds, from realistic simulations of natural phenomena to abstract and otherworldly sonic textures.

In summary, procedural audio is a powerful and versatile approach to sound generation that emphasizes real-time, algorithmic creation of audio content. By dynamically adapting to changing conditions and parameters, procedural audio enhances the interactivity and immersion of digital experiences, making it a valuable tool in the realm of interactive media and beyond.

2.2 Generative and Adaptive Music Systems

Generative audio and music represent a departure from the traditional method of using static audio assets. These systems leverage algorithms to dynamically generate audio content based on real-time inputs, allowing for a more adaptive and responsive auditory experience. As players navigate through virtual environments, interact with elements, or trigger specific events, generative audio systems respond dynamically, ensuring that the sonic landscape is as immersive and evolving as the visual components.

Research in generative audio and music for video games aims to go beyond the mere augmentation of graphics. It seeks to establish audio as a dynamic and integral part of the interactive experience, enhancing player engagement and emotional resonance. This exploration aligns with the industry's realization that true immersion extends beyond the visual domain and encompasses a multisensory engagement that includes audio.

As a result, there is a growing synergy between advancements in graphical fidelity and the exploration of generative audio and music. The goal is to create a harmonious blend where both visual and auditory elements contribute equally to the overall immersive experience in modern video games. This research trend is not just about catching up with graphical advancements but is an acknowledgment of the unique role that audio plays in shaping the way players perceive and engage with virtual worlds.

An Adaptive Music System (AMS) is a sophisticated audio design approach employed in interactive media, particularly in the context of video games, to dynamically tailor the musical experience based on changing parameters within the game environment [2]. Unlike traditional static music tracks, which follow a linear progression, adaptive music responds in real-time to various factors, creating a personalized and responsive auditory backdrop for the player [7]. This intricate system is designed to enhance player immersion, emotional engagement, and overall gaming experience.

At its core, an AMS is built on the principle of dynamic variability, allowing the music to adapt to the evolving conditions and events within the game. This adaptability is achieved through a combination of design principles, implementation strategies, and interactive triggers.

As noted by C. Plut and P. Pasquier in their analysis of the current state in Generative and Adaptive Music [5], 2017's Hellblade: Senua's Sacrifice [reference game here] is an example of a game state change. In Hellblade, there is a clear change between ”combat” and ”non-combat” – Senua draws her sword, the environment changes to create a small inescapable arena, the camera slightly changes its angle, and the players controls change to allow new actions. The music in Hellblade also changes with this state change. The “non-combat” music quickly fades out and the “combat” music quickly fades in. Games such as 1998's Baldur's Gate [15], 2009's Batman: Arkham Asylum [reference game here], and 2016's XCOM 2 [reference game here] use this technique as well.

An adaptive music engine that combines real-time music generation and affective musical mood transitions is proposed in “barelyMusician” [8]. This middleware tool focuses on practical composition transformation techniques, offering a comprehensive framework for real-time audio sample generation and manipulation.

Rodrigues and Coutinho (2021) introduce the “Adaptive Audio Manager” tool for Unity games, simplifying the implementation of adaptive audio techniques. While it has limitations, such as allowing only one layer to play simultaneously, it is a tool for developers seeking to create dynamic audio systems that do not need excessive complicated systems for adaptive music [9].

Adaptive Music often employs a layered structure, breaking down compositions into stems or individual components such as melody, harmony, rhythm, and instrumentation. These layers can be dynamically combined or adjusted based on in-game events. Vertical and Horizontal Re-Sequencing:

Vertical re-sequencing involves adjusting the intensity of individual musical elements, while horizontal re-sequencing involves changing the order of musical sections. These techniques enable nuanced responses to player actions, maintaining a sense of coherence. Interactive Triggers:

Adaptive Music Systems respond to interactive triggers within the game, such as player choices, progress, or specific events. These triggers act as dynamic cues, prompting transitions between different musical states.

Middleware solutions like Wwise red reference and FMOD red reference are often used for the implementation of Adaptive Music Systems. These tools provide developers with specialized features and APIs for interactive audio, facilitating the integration of adaptive music into games.

Adaptive Music often relies on data-driven methodologies, where game events, states, or parameters influence the music in real-time. This data-driven approach allows for a dynamic and flexible system that can adapt to diverse gameplay scenarios. Player Experience and Immersion:

Bossalini, Raffe, and Garcia (2020) [10] investigate the potential of dynamically rendered soundtracks in gaming environments. Their work highlights the need for combining experimental technologies with existing frameworks to create immersive and adaptive audio experiences. The paper presents a system that allows for the adjustment of individual audio track parameters and implements a region-based system to change global music parameters such as reverb, tempo, key, and pattern. Custom collider regions are used to create parameter control regions.

Lopez Ibanez, Alvarez, and Peinado (2018) [11] present an empirical study assessing the impact of adaptive music on player navigation in virtual environments. In this research study, the researchers aim to explore the impact of adaptive music in guiding players through virtual environments. They developed a video game featuring a 3D labyrinth and organized two groups of participants tasked with efficiently retrieving objects. Each group experienced a distinct audio version: one group had the ability to influence spatialized music by stating their musical preferences, while the other group encountered a default, non-adaptive spatialized soundtrack. The researchers measured task completion time as a metric for evaluating performance. The results revealed a statistically significant correlation between player performance and the inclusion of a personalized soundtrack. These findings suggest the absence of a definitive musical criteria for optimizing user engagement, highlighting the utility of the proposed adaptive system, particularly in addressing the needs of a diverse user base.

Plut and Pasquier (2020) [5] provide a comprehensive overview of generative music in video games. They discuss adaptive and generative music, offering a taxonomy of these approaches and suggesting the potential for combining both techniques to create unique and adaptive audio experiences. It is mentioned that music can be considered generative within a video game if the music is produced by a systemic automation that is partially or completely independent of the gameplay. This independence can have a large range of possibilities. For instance, a piece of music can be requested and then played linearly, without any connection to or influence from the ongoing gameplay. A highly adaptive generative music system may use a large array of game variables to inform the generation of musical content. It is worth noting that all but one of the generative music systems were using Markov models. Markov models are indeed simpler than deep learning/neural networks models, but they face the risk of plagiarism, by recopying too long sequences from the corpus.

Plut and Pasquier (2019) [4] also previously investigated the effects of adaptive music on player emotions and enjoyment. Their study emphasizes the positive influence of adaptive music on player experiences, particularly when the music aligns with gameplay tension, and conclude that adaptive music provides a better emotional response than linear music in games by using the Valence, Arousal and Tension graph to determine the emotional state of the player during certain scenarios.

Hutchings and McCormack (2020) [12] propose an adaptive music system based on cognitive models and multi-agent algorithms and implementing RNNs for these multi-agent algorithms that denote the harmony, melody and percussive layers of the music. The cognitive model in this context is the spreading activation model, adopted as a generalized model that combines explicit and inferred relationships between emotion, objects, and environments in the game world. It is used to generate context descriptions that are fed into a multi-agent music composition system.

The Multi-agent music composition agents a software agents that generate musical arrangements. These agents do not represent performers but instead represent roles that a player or multiple players in an improvising group may take. The agents are designed to have specific roles such as Harmony, Melody, or Percussive Rhythm for developing polyphonic compositions with a mixture of percussive and pitched instruments. The Harmony agent utilizes a recurrent neural network (RNN) model with two hidden layers and gated recurrent units to generate harmonies. The melody agents, on the other hand, generate melody lines with varying pitches, following common harmonization and counterpoint composition techniques. The number of melody agents and the style of music determine the range between the highest and lowest notes produced by the melody agents The melody agents also consider melodic shape, harmony, and rhythmic qualities when adding to the composition The model is used to determine the activation of concepts and affect categories from gameplay. Edges are formed between vertices based on coactivation of concept vertices and can also be assigned from game messages, allowing game creators to explicitly define associations.

This research is primarily concerned with “experience-driven procedural content generation,” a term coined by Yannakakis and Togelius [13] and applied to many aspects of game design.

Dos Santos et al. (2022) [14] introduce an architecture for generating music in games using the Transformer deep learning model, by building upon the approach by Hutchings and McCormack [12]. Their approach emphasizes customization based on player preferences and employs emotion models and layering to adapt music to gameplay, representing a promising direction in adaptive music generation. It uses the technique of layering, with the activation of layers controlled by an emotion model, in order to adapt it to the gameplay.

Coordinating adaptive music with other audio elements, such as procedural sound effects or voiceovers, presents ongoing challenges. Ensuring that all audio components work harmoniously within the interactive space is a crucial aspect of Adaptive Music System design.

2.3 Machine Learning in Audio For Games

Research has been conducted on utilizing machine learning for implementation in audio for games. A handful of research explores the use of machine learning and neural network techniques in extracting

Audio-based video game genre recognition is explored by Amiriparian et. al [15], introducing the idea of classifying video game genres based on acoustic features, opening the door for genre-specific soundscapes in adaptive audio systems. The extraction of three different feature representations from game audio files is performed: knowledge-based acoustic features, DEEP SPECTRUM features, and quantized DEEP SPECTRUM features using a ”Bag-of-Audio-Words”. If a genre can be classified using audio, the reverse is also possible that audio can be classified based on a genre. Features could include aspects like tempo, rhythm patterns, pitch, harmonies, and timbre. For example, fast and upbeat rhythms might be associated with genres like techno or pop, while slower tempos with distinct instrumental patterns could be linked to jazz or blues. Machine learning algorithms can be trained on datasets containing examples of different genres, learning to recognize patterns and associations between audio features and specific genres. Once trained, these algorithms can classify new, unseen audio tracks into predefined genres. This particular piece of research may very well help in setting up a layout for semantic input based on the game genre that the developer can define during initial developments of the game. This would aid in certain aspects of the generation of audio/music for the particular game. As this paper has clearly concluded that their proposed method has a statistically significant difference from other proposed methods, perhaps it is feasible to build upon this method, to have it implemented in conjunction with a generative audio system.

Ringer et al. (2023) [16] delve into the world of esports with their paper on predicting in-game deaths in Dota 2, using telemetry data and recurrent neural networks. An enhanced dataset is used, covering 9,822 Dota 2 matches and employing techniques such as learned embeddings for categorical features and modeling the temporal element of the telemetry data using recurrent neural networks. Although their primary focus is on esports analytics, their approach could be used for predicting upcoming audio cues or soundscapes in response to in-game events.

2.4 3D Objects and Mesh Recognition

In the context of gaming, 3D objects refer to digital entities that occupy three-dimensional space within a virtual environment. These objects can represent characters, props, structures, or any other element contributing to the visual composition of the game world. Unlike 2D objects that only have width and height, 3D objects have depth, allowing for a more realistic and immersive representation. The position, rotation, and scale of these objects contribute to their spatial relationships within the game environment, providing depth and perspective.

A mesh, often referred to as a polygon mesh, is a fundamental structure used to represent 3D objects in computer graphics. It consists of vertices, edges, and faces that define the shape and surface of the object. Vertices are points in 3D space, edges connect these points, and faces are polygons formed by connecting the edges. Triangles and quadrilaterals are common face types in polygon meshes. The arrangement and connectivity of vertices, edges, and faces collectively define the geometry of the 3D object. Meshes serve as the foundation for rendering visual elements in games, providing a framework for defining complex shapes and surfaces.

Object recognition and mesh recognition are areas of continuing research in the computer vision sector. Notable object detection algorithms and networks such as YOLOv7 [17], are implemented in real-world applications and games alike. One example of real-world implementation of object detection that in turn drives audio output is presented in [18], a project aimed at transforming the visual world into the auditory realm in real-time to assist blind persons. Using a combination of portable cameras, object detection models (specifically YOLO), and 3D binaural sound simulation, the system identifies objects and conveys their spatial locations through audio cues, which are generated using a program the authors developed and a Unity audio plugin called 3DCeption that renders the audio clips in spatial format. The prototype is tested in simulated scenarios where blind users successfully navigate and interact with their environment, demonstrating the potential of the proposed visual-to-audio conversion system. This approach in using ”objects” and giving them sonic characteristics stands relevant in our proposed methodology, using the more detailed characteristics of a 3D object - the mesh.

In games, Jung et. al [19], discuss about using object detection in analyzing computer games, as they propose a new model for improving object detection algorithms. While the research in object detection focuses primarily on the object itself, mesh recognition may offer a new outlook on applications pertaining to analyzing game scenes and even creative uses with the information received through these methods.

Mesh recognition models are computational systems designed to analyze and interpret the features of 3D objects represented by meshes. These models leverage various techniques, including machine learning and computer vision, to classify, recognize, or extract information from 3D meshes. One notable example is MeshCNN, a 3D Object Mesh recognition model that utilizes Convolutional Neural Networks (CNN) and is implemented using PyTorch. The paper presents MeshCNN, a special neural network for triangular meshes. It uses specific convolution and pooling layers that work on mesh edges. MeshCNN takes advantage of the unique characteristics of mesh analysis by using the intrinsic geodesic connections of mesh edges, enabling a direct analysis of 3D shapes. In the context of 3D meshes, intrinsic geodesic connections refer to the intrinsic geometric relationships and distances between vertices connected by edges. Geodesic connections are paths that follow the intrinsic curvature of the mesh surface, providing a measure of the shortest distance between two points along the surface.

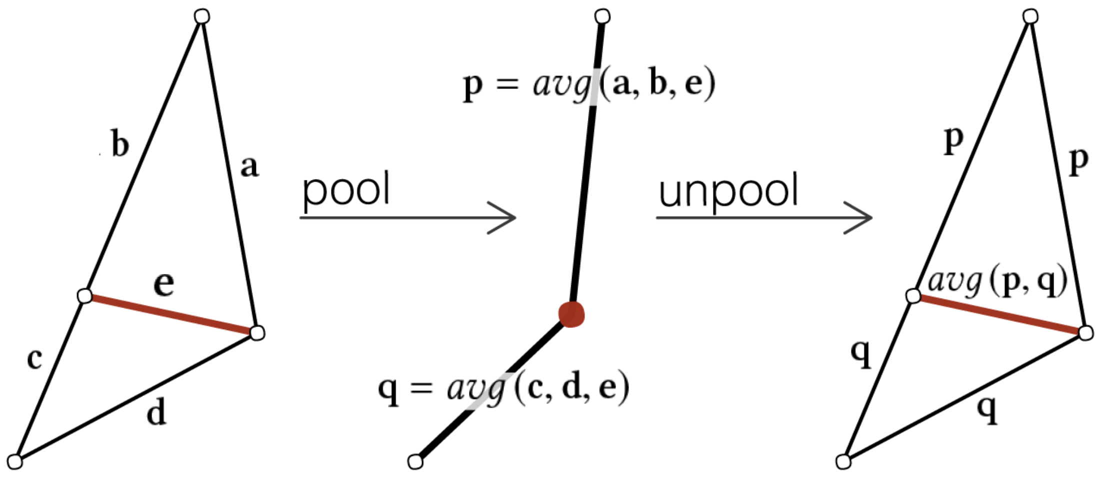

Figure 2.1: MeshCNN pooling and unpooling.

The network combines specialized convolution and pooling layers that operate on the mesh edges, leveraging their intrinsic geodesic connections. Convolutions are applied on edges and the four edges of their incident triangles, while pooling is applied via an edge collapse operation that retains surface topology. MeshCNN learns which edges to collapse, forming a task-driven process where the network exposes and expands important features while discarding redundant ones. The mesh provides connectivity for convolutional neighbors and initial geometric input features, with edge features evolving as they progress through the network layers. Classic mesh processing techniques, such as edge-collapse, are used for task-driven pooling, reducing the resolution of feature maps within the network.

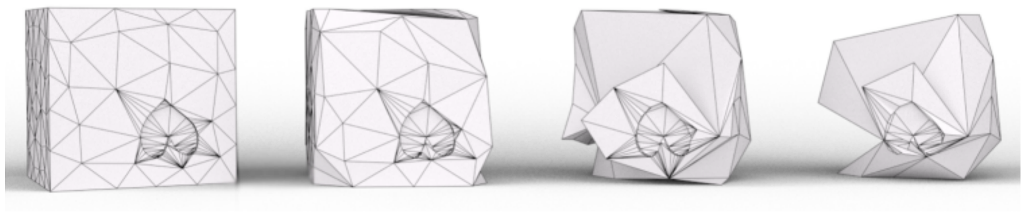
Figure 2.2: Learned Simplifications of MeshCNN on Cube Dataset.

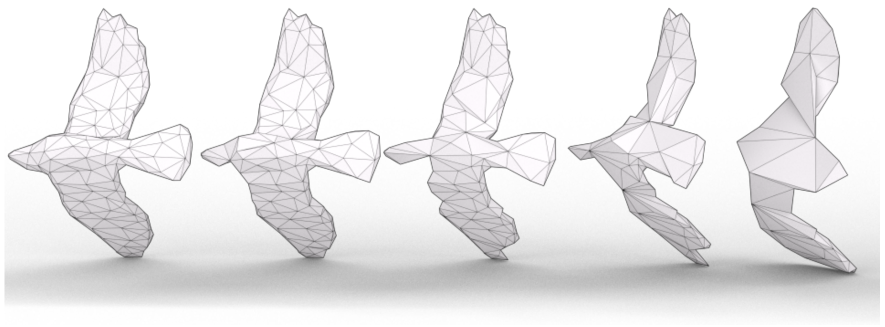
Figure 2.3: Learned Simplifications of MeshCNN on Shrec Dataset

In [20], a method of recognizing 3D objects using a machine learning algorithm is described, specifically by analyzing 3D object data sets consisting of geometric polygons and applying composition rules to the objects. The deep learning API Keras is used to learn the composition rules of the data sets. The paper experiments with various types of 3D objects and compares evaluation results for different numbers of objects. The process involves analyzing 3D object data sets consisting of geometric polygons using the Keras deep learning API. Data pre-processing is performed to match data formats for machine learning. A model for training is defined, and the 3D data set is trained using the model for machine learning. The DNN (Deep Neural Network) model algorithm is used to estimate the shape of a 3D object based on a sequence of input 3D values.The Adam optimizer, a combination of Momentum and RMSProp algorithms, is used to minimize the loss function.

Deep Neural Networks have been further explored in [1], proposing a network architecture that combines triangular mesh and graph convolutional neural networks to process mesh data and extract useful information.The 3D model is represented as a graph and leverages graph convolutional networks to enhance local feature extraction. The network utilizes an adjacency matrix obtained from the mesh data to apply graph convolutional neural networks for processing mesh data.

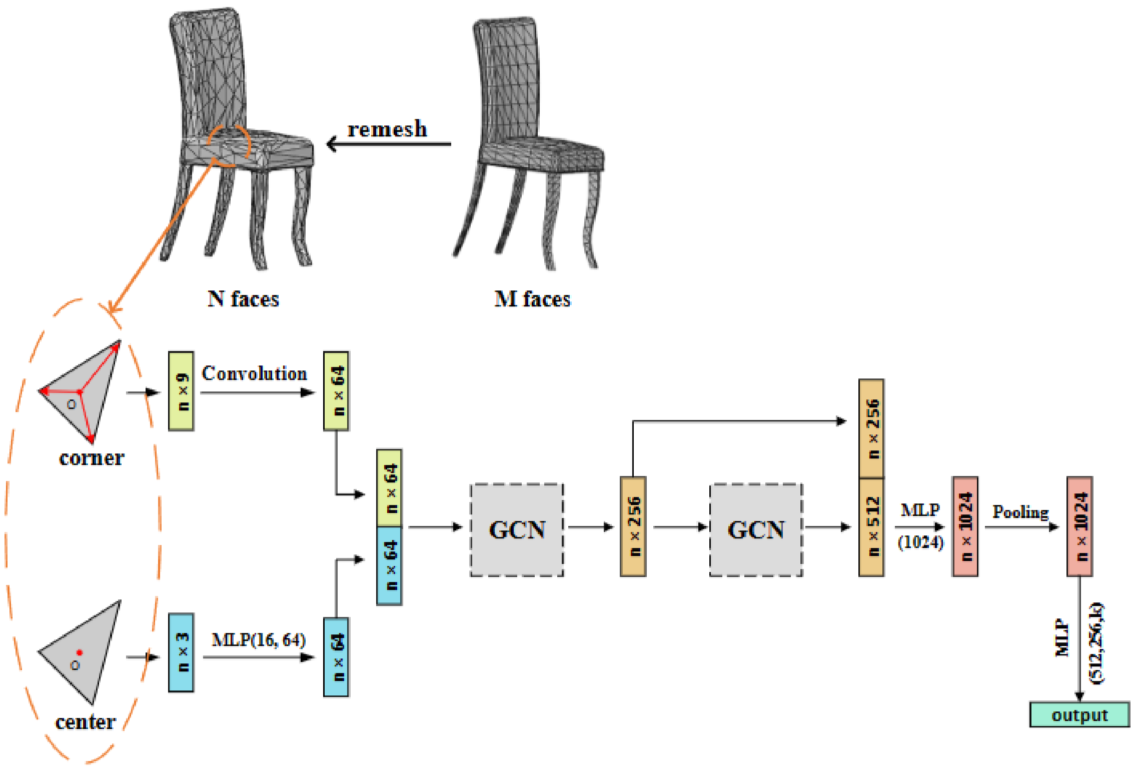

Figure 2.4: Algorithm flow chart by Gao, M. et Al [1]

An adjacency matrix is a mathematical representation of connections between vertices in a graph. In the context of mesh data, which represents 3D objects using vertices, edges, and faces, the adjacency matrix encapsulates the relationships between vertices. In this matrix, each entry corresponds to a pair of vertices, indicating whether there is an edge (connection) between them.

Graph convolutional neural networks (GCNNs) are a type of neural network architecture designed to operate on graph-structured data, like the aforementioned mesh data. In the case of meshes, the vertices and edges form a graph, and GCNNs leverage the adjacency matrix to perform convolution operations. Unlike traditional image-based convolutional neural networks, which operate on regular grid-like structures, GCNNs adapt to the irregular and interconnected nature of graph data.

Mesh recognition models play a crucial role in tasks such as object recognition, shape classification, and semantic segmentation, contributing to applications ranging from computer-aided design to virtual reality and gaming. These models enable machines to understand and categorize the intricate details of 3D objects, facilitating their integration into interactive digital environments.

CHAPTER 3

Methodology

In this methodology, we blend 3D object recognition, procedural audio design, and Unity game development. We start by preparing 3D object data and implementing MeshCNN in Unity for real-time recognition. Barracuda helps integrate MeshCNN into Unity, and we use ONNX [21] for compatibility.

Simultaneously, we craft a procedural audio engine in PureData, incorporating PureData into Unity through LibPd [22] and [23], overhauling Unity's audio engine. The procedural audio engine design draws inspiration from Andy Farnell's PureData procedural audio structure in [3], such as the bouncing ball algorithm showin in Figure 7.

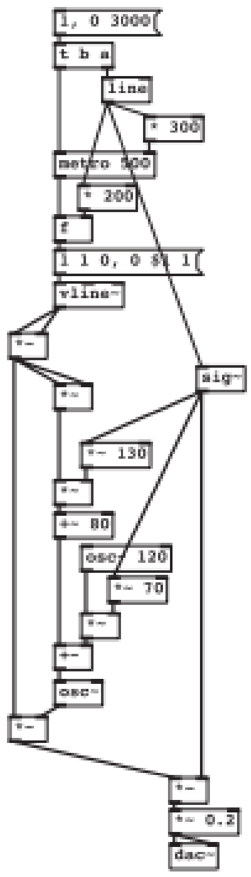

Figure 3.1: Bouncing Ball Model in PureData based on Andy Farnell's implementation

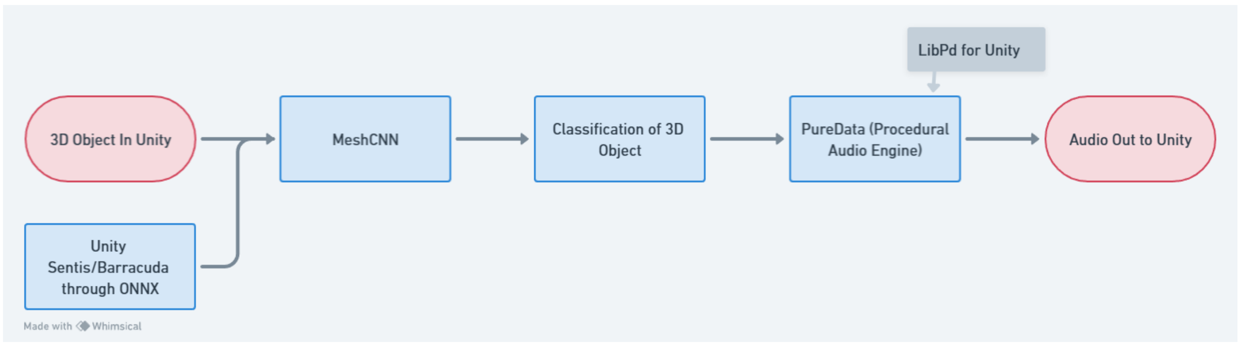
3D Object In Unity
Unity Sentis/Barracuda through ONNX
MeshCNN
Classification of 3D Object
LibPd for Unity
PureData (Procedural Audio Engine)
Audio Out to Unity
Made with Whimsical

Figure 3.2: Simplified system flow for utilizing MeshCNN as an input to a procedural audio engine

Communication in Unity relays classification info to PureData during runtime. Rigorous testing validates 3D object recognition and system integration.

A simplified system of the approach proposed in given in Figure 8.

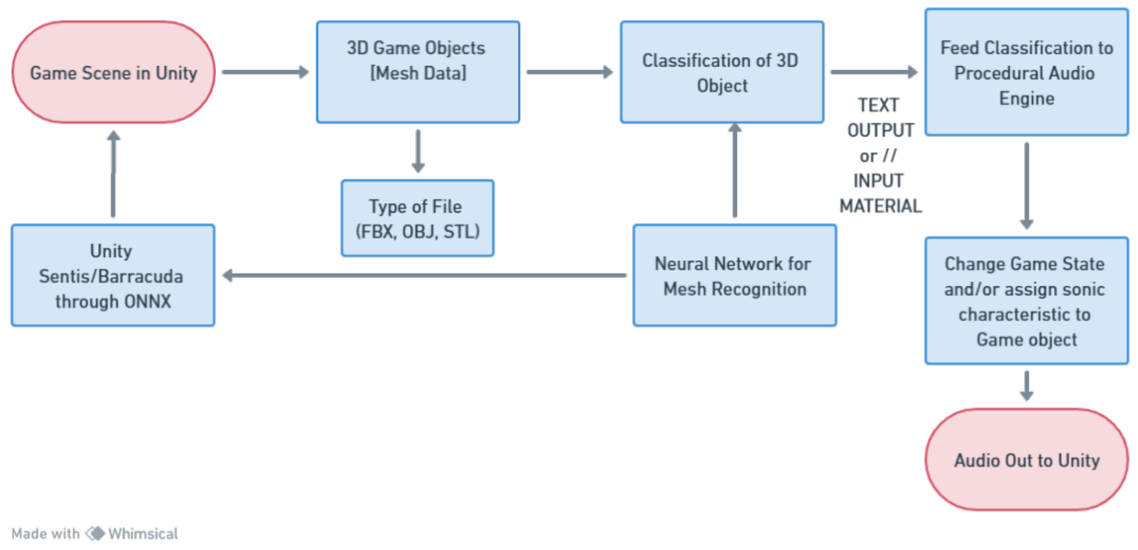
Game Scene in Unity
3D Game Objects [Mesh Data]
Classification of 3D Object
Feed Classification to Procedural Audio Engine
TEXT OUTPUT or // INPUT MATERIAL
Type of File (FBX, OBJ, STL)
Unity Sentis/Barracuda through ONNX
Neural Network for Mesh Recognition
Change Game State and/or assign sonic characteristic to Game object
Audio Out to Unity
Made with Whimsical

Figure 3.3: Comprehensive outlook of the proposed architecture

Bibliography

[1] Mengran Gao, Ningjun Ruan, Junpeng Shi, and Wanli Zhou. Deep Neural Network for 3D Shape Classification Based on Mesh Feature. *Sensors*, 22(18):7040, September 2022.

[2] Michael Sweet. *Writing Interactive Music For Video Games*. Addison-Wesley, 2015.

[3] Andy Farnell. *Designing Sound*. MIT Press, 2010.

[4] Cale Plut and Philippe Pasquier. Music Matters: An empirical study on the effects of adaptive music on experienced and perceived player affect. In *2019 IEEE Conference on Games (CoG)*, pages 1–8, London, United Kingdom, August 2019. IEEE.

[5] Cale Plut and Philippe Pasquier. Generative music in video games: State of the art, challenges, and prospects. *Entertainment Computing*, 33, March 2020.

[6] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. MeshCNN: A Network with an Edge. *ACM Trans. Graph.*, 38(4):1–12, August 2019. arXiv:1809.05910 [cs, stat].

[7] Anthony Prechtl. Adaptive Music Generation for Computer Games. 2016. Publisher: The Open University.

[8] Alper Gungormusler, Natasa Paterson-Paulberg, and Mads Haahr. An Adaptive Music Engine For Video Games. 2015.

[9] Miguel F. Rodrigues and Flavio R. S. Coutinho. A Tool to Implement Adaptive Audio Techniques on Unity Games. In *2021 20th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, pages 1–8, Gramado, Brazil, October 2021. IEEE.

[10] Cameron Bossalini, William Raffe, and Jaime Andres Garcia. Generative Audio and Real-Time Soundtrack Synthesis in Gaming Environments: An exploration of how dynamically rendered soundtracks can introduce new artistic sound design opportunities and enhance the immersion of interactive audio spaces. In *32nd Australian Conference on Human-Computer Interaction*, pages 281–292, Sydney NSW Australia, December 2020. ACM.

[11] Manuel Lopez Ibanez, Nahum Alvarez, and Federico Peinado. Assessing The Effect of Adaptive Music On Player Navigation In Virtual Environments. *Proceedings of the 21st International Conference on Digital Audio Effects (DAFx-18), Aveiro, Portugal, September4-8, 2018*, 2018.

[12] Patrick Edward Hutchings and Jon McCormack. Adaptive Music Composition for Games. *IEEE Trans. Games*, 12(3):270–280, September 2020.

[13] G. N. Yannakakis and J. Togelius. Experience-Driven Procedural Content Generation. *IEEE Trans. Affective Comput.*, 2(3):147–161, July 2011.

[14] Gustavo Amaral Costa dos Santos, Augusto Baffa, Jean-Pierre Briot, Bruno Feijó, and Antonio Luz Furtado. An adaptive music generation architecture for games based on the deep learning Transformer mode, September 2022.

[15] Shahin Amiriparian, Nicholas Cummins, Maurice Gerczuk, Sergey Pugachevskiy, Sandra Ottl, and Bjorn Schuller. "Are You Playing a Shooter Again?!" Deep Representation Learning for Audio-Based Video Game Genre Recognition. *IEEE Trans. Games*, 12(2):145–154, June 2020.

[16] Charles Ringer, Sondess Missaoui, Victoria J. Hodge, Alan Pedrassoli Chitayat, Athanasios Kokkinakis, Sagarika Patra, Simon Demediuk, Alvaro Caceres Munoz, Oluseji Olarewaju, Marian Ursu, Ben Kirman, Jonathan Hook, Florian Block, Anders Drachen, and James Alfred Walker. Time to Die 2: Improved in-game death prediction in Dota 2. *Machine Learning with Applications*, 12:100466, June 2023.

[17] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, July 2022. arXiv:2207.02696 [cs].

[18] Rui Jiang, Qian Lin, and Shuhui Qu. Let blind people see: real-time visual recognition with results converted to 3d audio. *Report No. 218, Standord University, Stanford, USA*, 2016.

[19] Minji Jung, Heekyung Yang, and Kyungha Min. Improving Deep Object Detection Algorithms for Game Scenes. *Electronics*, 10(20), October 2021.

[20] Ha-Seong Kim and Myeong Won Lee. 3D Object Recognition Using X3D and Deep Learning. In *The 25th International Conference on 3D Web Technology*, pages 1–8, Virtual Event Republic of Korea, November 2020. ACM.

[21] Open neural network exchange. The open standard for machine learning interoperability.

[22] Libpd. Pure Data embeddable audio synthesis library.

[23] upd. A relatively complete alternative to Unity's audio engine using Pure Data and LibPD.

APPENDIX