# Midi Machina

Chern Yang and Griffin Schulman

## Abstract

This project is an *Animusic*-inspired animation that uses an automation program to map MIDI input to Blender keyframes. Emphasis in this project is twofold: on a visually appealing final result, and on establishing a foundation for future projects. The project was generally successful, albeit with some limitations due to time constraints, such as the level of polish on the animations and the rendering quality.

## Introduction

Lying at the intersection between computer graphics and music composition, this project is inspired by the 2001 and 2005 animations, *Animusic* and *Animusic 2*. Our objective is to recreate the whimsical, self-playing instruments and fantastical charm of the original videos, but modernized through Blender in a brand new animation. By doing so, not only do we unlock more powerful rendering capabilities (photorealistic textures, ray tracing), but by the open source nature of Blender, we establish a framework upon which any design can be brought to life by anyone.

Aside from the design of the animation itself, the central contribution of this project is the automation achieved via Blender's native scripting API, which takes an arbitrary MIDI file and automatically animates the scene. To be clear, such automation is not a novel concept – in fact, the original *Animusic* videos were created in this fashion. However, the original program is proprietary and confidential, and fan-made videos have either forgone this infrastructure and animated everything manually, or not disclosed the automation they used. Regardless, by constructing our automation program in Blender and modularizing it as cleanly as possible, we hope that this project yields not only a neat animation but also useful scaffolding for any similar future projects.

## Methodology

Our implementation revolves around the Mido python library, which allows for seamless interpretation of MIDI files in a python environment. On a broad level, the program takes in the MIDI files, parses the file with Mido, matches the data to the animation – mapping each track to the correct instrument, translating MIDI "ticks" to frame numbers – and then animates the scene by setting keyframes. What is difficult is the difference between instrument animations. Some, like hammer- or stick-based instruments, might require a simple rotation. Others might require a linear translation, such as piston-based contraptions. More complex ones might require interpreting transitions between notes, the most complicated of which are the mechanical limbs commonly featured in the original *Animusic*. By modularizing the code and setting a precedent

for each of these types of animations, our goal is for the program to be applicable not only to the specific instruments of this project, but also to any such fictional instrument in Blender.

In terms of making the animation itself, the process is fairly straightforward, albeit tedious. Traditional 3D modeling techniques are employed to construct the instruments and the scene. Almost all the models in the animation are made from scratch, with the exception of the plant life, the benches, and the HDRI background environment. Then, shaders and textures are applied to each. Materials are a mix of simple BSDF shaders, BSDF shaders with procedural effects (fractal Perlin noise, Voronoi textures), and photorealistic textures obtained from free online resources. Everything from start to finish is accomplished in Blender, in keeping with the open-source objective of this project.

## Challenges

Installing and importing Mido proved to be a challenge when we had to start testing inside of Blender's bundled python environment. Mido (and then Packaging) wouldn't import in Blender, so we had to vendor dependencies into the project folder. After getting results in Blender, we realized we needed a quick way to test our changes, so we incorporated a "clear all keyframes" function into our main.py as we started having issues with unexpected axis rotations. Another challenge was matching the motion to hit points; we found that the inserted keyframes matched almost perfectly with the audio pairing, but each animation required a little extra motion to achieve a more realistic effect. We added windup, hit, rebound, and settle keyframes for every instrument. Vibrating strings with shape keys had their own tuning problem (cycles, step size, decay curve). Not to mention, it took some investigating to figure out how to access some of these parameters, like object mesh material BSDF shader emission value (for glowing in the organ). Translating MIDI structure into animation meant we needed a lot of hard mappings from Blender object names to pitches, which was a tedious process. In general, the testing process was a tedious one, with lots of reruns, reloads, and checking object names inside of Blender.

## Results and Discussion

We built an end-to-end pipeline that takes MIDI file input and generates a synchronized, music animation driven by mechanical instruments. The program successfully parses note events (pitch, velocity, note-on/-off) into a structured representation, translates MIDI timing to Blender frame timing, and performs some procedural animation by inserting keyframes for note-mapped objects in the scene. This program achieves the core objective of mimicking an Animusic-style animation with original music and animated instruments that interact with the music.
Not all of the "physics" is quite perfect, especially on the drumset and the trumpet-lasers. We tried mapping values within a range to direct the lasers in interesting patterns, but found it difficult to adjust keyframes to achieve a more natural look. However, the harp looks really nice with the level of bounce in the hammer and vibration decay in the string. Splitting the main functions of the pipeline into its main file components (MIDI parsing, note data structure, per-instrument animation modules) made it easier to debug and tune parameters.

## Future Steps

One possible future step is trying instrument mappings with JSON instead of hard-coding every instrument in; this could be useful for inputting other MIDI files in the same scene. From that, the animation scripts assume specific object names and available shape keys/materials, so perhaps more portability in the future could make it more transferable to new scenes. In terms of aesthetics, we could add velocity-aware motion and glow to swings, hits, and vibrations. Better easing and constraints would allow for smoother transitions with Bezier curves. Automated camera cuts with beat-synced lighting changes would turn this scene into the ultimate Animusic-style music video.

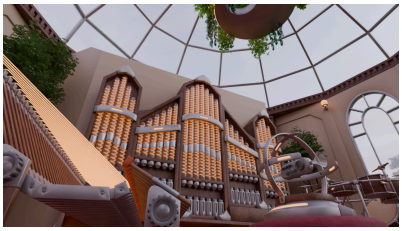
## Ethics Note

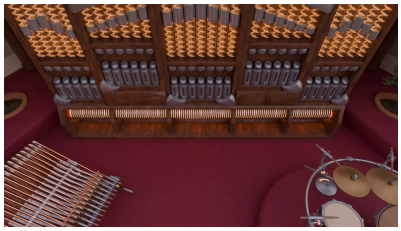
In accordance with ACM's Code of Ethics and Professional Conduct, section 1.5, we reiterate that this project is inspired by *Animusic*, the creation of Wayne Lytle and David Crognale. We acknowledge their contributions, and use their intellectual property in the spirit of pastiche. The importance of enshrining ideas is ever more salient in this age of AI, and *Animusic* is, if anything, a celebration of human creativity.

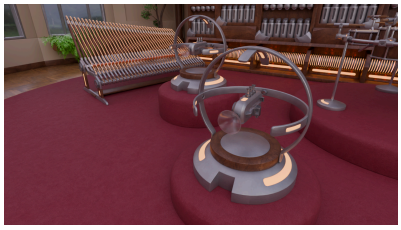
To that end, we would also like to acknowledge Lytle's personal struggles with respect to *Animusic* – since this project is intended as a kind of revival or continuation, it is important to recognize why it stopped production in the first place. During the development of *Animusic 3*, Lytle was dealing with physical and mental health issues, including ADHD, depression, Repetitive Strain Injury (RSI), and Bell's Palsy. Unfortunately, as a result, *Animusic 3* was never released. We would like to bring attention not only to those circumstances, but Lytle's immense personal disappointment in not finishing the album and continuing the project/business. As such, we present our work in the spirit of honoring Lytle's efforts, and, as with many ardent *Animusic* fans, continuing the journey that he could not.

# Media

Some stills from the scene, to showcase all the details of the animation.







# References

The programming portion of the project was primarily handled by Griffin, and the modeling portion of the project was primarily handled by Chern.


*Animusic* (2001), *Animusic 2* (2005). Created by Wayne Lytle and David Crognale.
Blender API: https://docs.blender.org/api/current/index.html
Mido API: https://mido.readthedocs.io/en/stable/index.html
Mido implementation reference: http://course.ece.cmu.edu
HDRI background: https://polyhaven.com/a/bryanston_park_sunrise

Plants hanging above scene: https://www.cgtrader.com/items/6198822/download-page
Ferns on windowsills: https://www.cgtrader.com/items/3797654/download-page
Trees to left and right of organ: https://www.cgtrader.com/items/4840562/download-page
Benches along walls: https://www.cgtrader.com/items/3177481/download-page
Smaller potted trees: https://www.cgtrader.com/items/6539844/download-page

Herringbone flooring: https://polyhaven.com/a/herringbone_parquet
Velvet carpet: https://polyhaven.com/a/velour_velvet
Lighter wood: https://polyhaven.com/a/plywood
Darker wood: https://polyhaven.com/a/wood_table_worn
Bark: https://polyhaven.com/a/chinese_cedar_bark
Clay pot: https://polyhaven.com/a/concrete_floor
Dirt: https://polyhaven.com/a/brown_mud_02

Knowledge sources:
https://en.wikipedia.org/wiki/Animusic
https://midi.org/animusic-midi-driven-computer-animation

Fan-made *Animusic* examples:
https://youtu.be/MtcDuqGMvGo?si=t8elYUchwQXQkScK
https://youtu.be/x3h40yV3C7w?si=OTo32-NW5f51NdR4
https://youtu.be/ZV84tO8r_zM?si=QqwWGVn1RtBDTmRQ