Video Game Music Generation with CUDA-trained Markov Chains and Multithreading Optimizations

Annie Xu (jingjinx@andrew.cmu.edu) Michael You (myou@andrew.cmu.edu)

15-418 Final Project

Abstract

Most video games use static soundtracks, which although can be impressive in their own right, can quickly become repetitive to the gamer. In this project, we build a complete system that can process, learn, and generate new video game music. Our system takes in MIDI files, which are processed and trained as a Markov Chain. We then use our Markov Chain to generate a multi-voice music excerpt. We optimize our algorithms by using CUDA techniques in our training algorithm, and multi-threading in the music generation. Speedup analysis shows that our system performs significantly better than sequential algorithms. In fact, because our system is fast, we explore the possibility of integrating our music generation method into video games, so that games can have real-time generated music for a more interactive and unique gaming experience.

Background

- Music can be treated as a set of rules - ideal for an algorithmic implementation
- Video game music is characterized by the overlapping of multiple melodic lines and a repetitive and structured format
  - Old 8-bit music constrained composers
  - Newer video game music is often symphonic
- Many melodic lines offer ample avenues for parallelization - parallel synthesis of different parts
- Rigid structure makes music easier to predict and generate with a Markov Model

1st – Order Matrix

Note	A	D	G
A	0.1	0.6	0.3
D	0.25	0.05	0.7
G	0.7	0.3	0

2nd – Order Matrix

Note	A	C	G
AA	0.18	0.6	0.22
AD	0.5	0.5	0
AG	0.15	0.75	0.1
DD	0	0	1
DA	0.25	0	0.75
DG	0.9	0.1	0
GG	0.4	0.4	0.2
GA	0.5	0.25	0.25
GD	1	0	0

- Matrices built from training data which fit a similar genre
- During generation, probabilities from transition matrix are used to generate next note based on current notes
- Generated music uses structure and patterns from training music
- Training requires looking at millions of notes and updating a very large matrix
- Generation has ~10 music lines generated at once, where each line requires complex computation and synchronization

Process

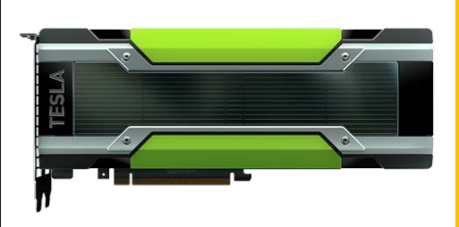
Preprocessing

- Convert MIDI to text
- Transpose to C Major
- Specific encoding

music21

Markov Model Training

- Counts note transitions
- Parallelized over multiple GPUs and threads
- Split between matrix type (major/minor, soprano/bass/chord) on GPUs
- Split notes into sections for many threads to read



Music Generation

- Creates music based on client request and global settings
- Uses trained Markov Chain to make musical choices
- Multithreading to generate many voices at once
- Synchronization of voices via the conductor

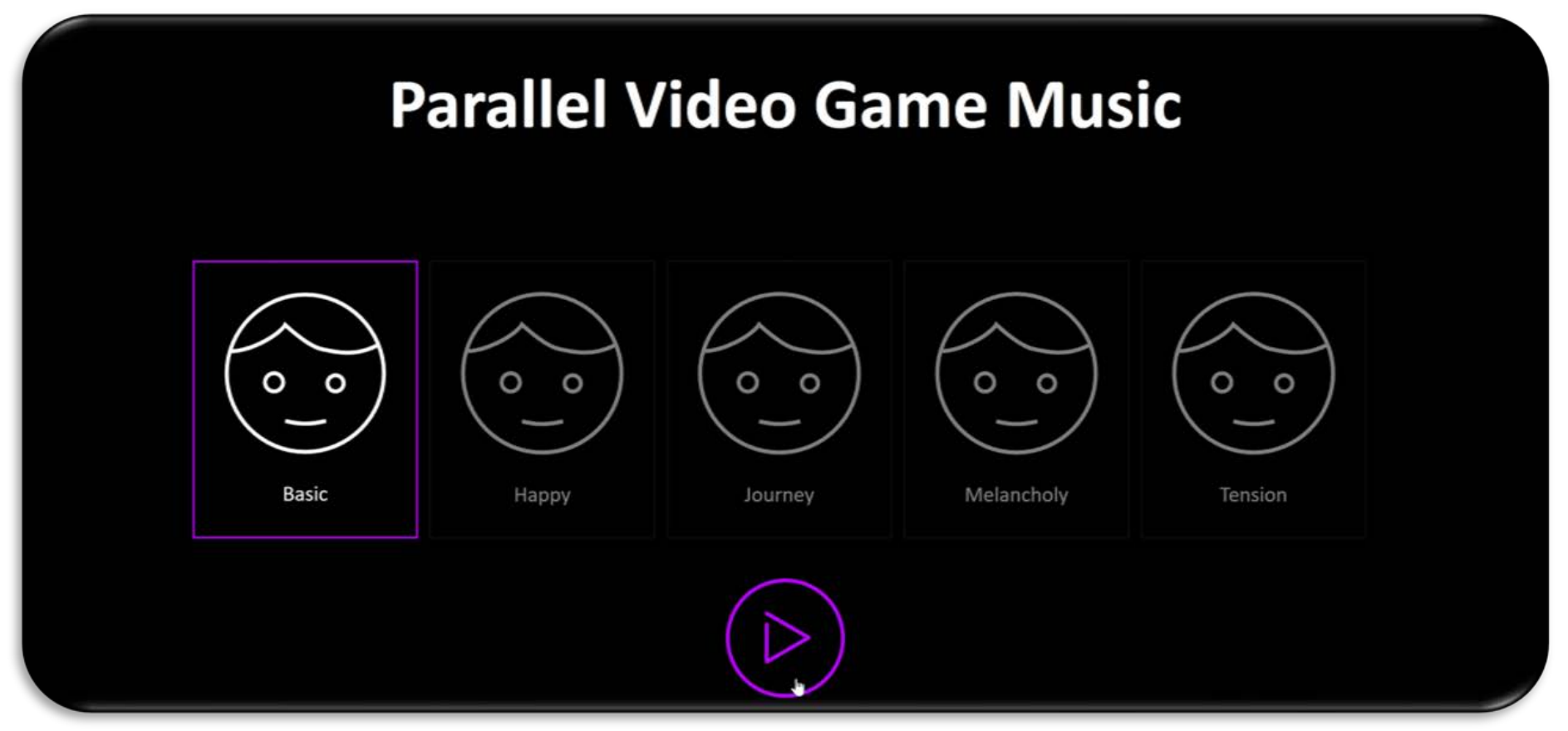
websockets

multi-processing

$$\begin{bmatrix}0.5 & \cdots & 0.4\\ \vdots & \ddots & 0.5\\ 0.1 & \cdots & 0.1\end{bmatrix}$$

Client

- Requests music
- Plays notes from server



Optimizations and Results

Challenges Faced

- MIDI files are hard to parse and missing important information – key signature, rhythm, etc.
- Required pre-processing and re-formatting of input MIDI files into a text file
- Pre-processing and server algorithms use different languages and parallelism techniques
- Balance of size in Markov Model - larger matrices mean more detailed probabilities and likely better sounding music, but take up a lot of storage
- Client-Server communication of generation settings and live music generation results
- Multithreading vs. GPUs balance in Markov training (much more material to parallelize) and generation (fewer parts and heavier computation)
- Gave user the ability to customize the music they wanted to generate, including live-mixing

Conclusion

- Created a real-time video game music generator!
- With parallelism, we see speedup in both model training and music generation
- Depending on the structure of the issue, multiple forms of parallelism (GPUs vs. multithreading) are better for different problems
- Even with communication costs, especially between voices, parallelism is still able to achieve both speedup and maintain musicality

Future Work

- Explore feasibility of using parallel computation for video game music in industry
- Generating video game music in real time along with gameplay
- Further and more complicated setting manipulation on the client side
- Expansion into other genres/types of music

Acknowledgments

- Professor Mowry and Professor Railing
- 15-418 TA Staff
- Carnegie Mellon University

References

- "Algorithmic Composer." OpenMusic Markov Chains and Omlea, 23 Apr. 2010, www.algorithmiccomposer.com/2010/04/openmusic-markov-chains-and-omlea.html.
- Elowsson, A. and Friberg, A. "Algorithmic Composition of Popular Music," in Proceedings of the 12th International Conference on Music Perception and Cognition and the 8th Triennial Conference of the European Society for the Cognitive Sciences of Music, July 2012, pp 276-285
- Dannenberg, R. (2018). Music Generation and Algorithmic Composition [Powerpoint slides]. Retrieved from http://www.cs.cmu.edu/~./music/cmsip/slides/05-algo-comp.pdf
- Collins, Karen. "In the Loop: Creativity and Constraint in 8-Bit Video Game Audio." Twentieth-Century Music, vol. 4, no. 2, 2007, pp. 209–227., doi:10.1017/S1478572208000510.
- Whalen, Zach. "Case Study: Film Music vs. Video game Music: The case of Silent Hill." 2007.
- Seabrook, Andrea (Host). 13 April, 2008. The Evolution of Video Game Music [All Things Considered]. https://www.npr.org/templates/story/story.php?storyId=89565567