Due: Monday, October 23, 2017, 8:00 AM (submit to BlackBoard, under Assignments)

File Type: Microsoft Word

Highlighting indicates changes

Team Name: The Effervescent Violators of International Legislation (E.V.I.L.)

Members/Contact:

- Brandon Jensen, brandon.jensen@ku.edu
- Jonah Mooradian, j211m087@ku.edu
- John Russell, j362r647@ku.edu
- Sierra Seacat, s831s804@ku.edu
- Nicholas Shaheed, nicholasshaheed@gmail.com

Nicholas von Beethoven, nicholasshaheed@gmail.com

J. Jonah Jameson, j211m087@ku.edu

John Ron Swanson, j362r647@ku.edu

Sierra Seakitty, s831s804@ku.edu

B-man, brandon.jensen@ku.edu

Team Meeting time:

T 2:30-4:30 pm

Lab Meeting time:

W 1 pm

Project Sponsor (if any): n/a

Project Description

We are currently developing a GPU-accelerated synthesizer. Many synthesizers (both digital and analog) create sound using simple oscillators: sine waves, square waves, sawtooth waves, etc. Generally, these synthesizers use somewhere between 3-15 of these oscillators concurrently to create different sounds, with a focus on fine tuning the parameters of these oscillators to achieve particular sounds. The goal of this project is to scale up the standard synthesizer, and use the GPU to create a synth that plays thousands of oscillators at once. This project will be an opportunity for musicians/programmers/etc. to create sounds, textures, and music that are not possible with conventional tools, hopefully allowing for new approaches to music making.

The end result will include one or two interfaces to this synth: the first will be a domain specific language that will allow the user the maximal flexibility in controlling the synthesizer. Scripts in this language will be commands that control the oscillators at a higher level than fiddling with the parameters of a single oscillator. For example: have all oscillators slide to a single frequency over 3 seconds, or assign the frequency of each oscillator based off of its index.

The second interface will be a GUI that allows for interactive controlling of certain subsets of the DSL, expanding the possible users of the synthesizer beyond people with experience programming. If we run out of time, we may drop the second interface from our final design. The end result would still work fine, just be less user-friendly.

Both interfaces will be plugins to existing digital audio software--in our case, Max.

Project Milestones

- 3-5 specific and measurable objectives per semester for first & second semester
    - Jobs Breakdown:
        - DSL stuff:
            - Nick, Brandon
        - GPU stuff:
            - John, Jonah, Sierra
        - GUI/Max stuff
            - Jonah, Nick
        - Documentation
            - Everyone
    - Spring:
        - Create DSL parser/interpreter
            - parser
                - Dec 1 - March 15
            - interpreter
                - Dec 1 - March 15
        - Create virtual instrument plugin/GUI
            - Beginning of Spring semester - April 1
        - Create GPU backend
            - Interface between max and DSL commands
                - Jan. 1 - March 15.
            - Implement DSL commands
                - Jan. 1 - April 1
        - Create demo video
            - Script writing
                - April 15 - 25
            - Recording
                - April 25 - 27
            - Editing
                - April 27 - May 1
        - Documentation
            - March 15 - May 4

Project Budget

- Hardware, software, and/or computing resources
    - Max/MSP software: 1 license (acquired).
    - GPU
        - GPUs are standard in most PCs, and the lab has a couple computers with hefty GPUs already. We will likely not need to purchase any additional hardware.
    - Ableton Live
        - Professional digital audio workstation (DAW) software, has built in Max/MSP integration making it ideal to create small GUI plugins that use our DSL (to reach a wider audience).
- Estimated cost
    - $59 yearly license for Max/MSP subscription
    - $449 student license for Ableton Live
    - Total: $508
- Vendor
- When they will be required?
    - Mid October to early November

Work Plan

- Who will do what?
    - Create DSL parser/interpreter
        - Nick, Brandon
    - Create virtual instrument plugin/GUI
        - Jonah
    - Create GPU backend
        - Sierra, John
    - Create demo video
        - Everyone
    - Documentation
        - Everyone

Github Link

- https://github.com/nshaheed/gigasynth (It is currently private)

Preliminary Project Design

Overview

The design of the Gigasynth is based around using a GPU to perform the calculations for hundreds/ thousands of oscillators. Synthesizers are simply wave oscillators. While there are programs available that allow multiple oscillators to run at once, they are limited by computing power, thus the number of concurrently running oscillators is smaller than what we will be able to accomplish using a GPU. This GPU backend will be integrated with the existing software, Max/MSP as a plugin, and Ableton Live, as a Virtual Instrument. A Domain Specific Language

(DSL) will also be created to control the Gigasynth independently and through Ableton Live. Ableton Live is a Digital Audio Workstation commonly used by musicians. Thus, the project can be considered to have three primary components: the GPU backend integrated with Max/MSP using the Max SDK, the Domain Specific Language, and the Ableton Live Virtual Instrument. Fig. 1 gives an overview of the design.

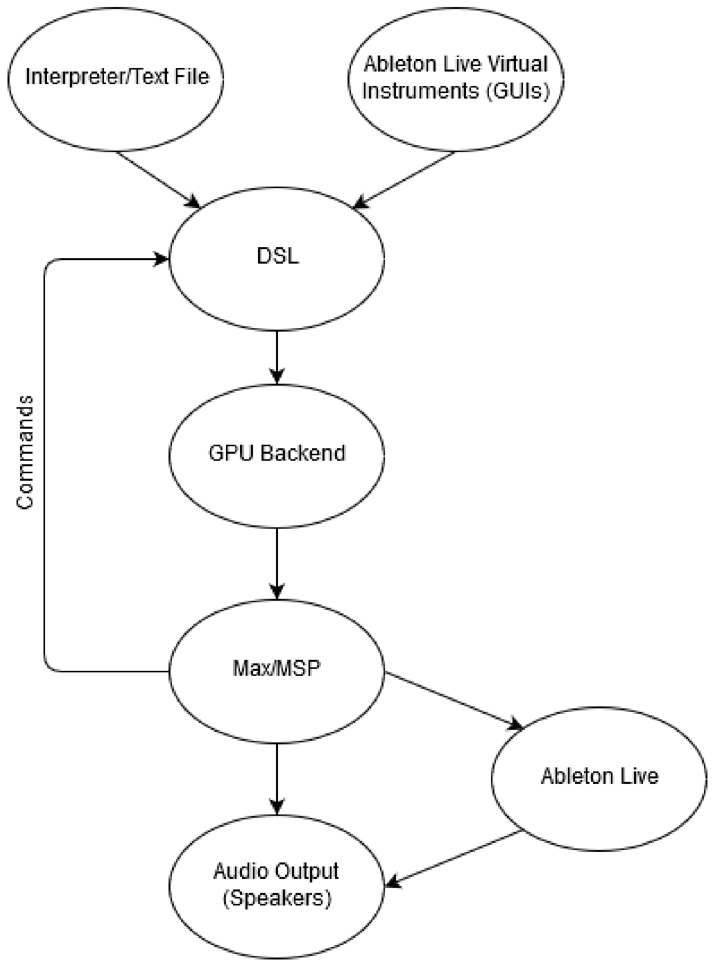


Fig. 1. Design overview of the Gigasynth. DSL commands will be issued either by text files, through an interpreter, the Ableton Live Virtual Instrument, through a GUI, or directly from Max/MSP. The DSL will control the GPU backend, which will integrate with Max/MSP as a plugin. This will either be sent directly to the speakers for Audio Output or sent through Ableton Live for additional processing.

II. GPU backend

The GPU backend will control the oscillators running on the GPU. The GPUs will be used for the computations that control the oscillators. The oscillations are just amplitudes than change as a function of time. The GPUs will perform the calculations that update the amplitudes in time. Essentially, multiple oscillators will be declared and run on the GPU. These signals will then be combined and normalized and output as a single waveform.This output will be sent to an audio driver and output as sound. Fig. 2 illustrates the process the waveforms will undergo.

This will integrate with the existing program Max/MSP as a plugin. To create the plugin, we will use the Max SDK. This will allow for increased functionality of our program as Max/MSP is currently used by many electronic musicians and composers, the target audience for our software.

The GPU communication will be handled using ArrayFire, a C++ library that streamlines GPU computations. All backend code will be written in C/C++. The Max SDK is also written in C++.

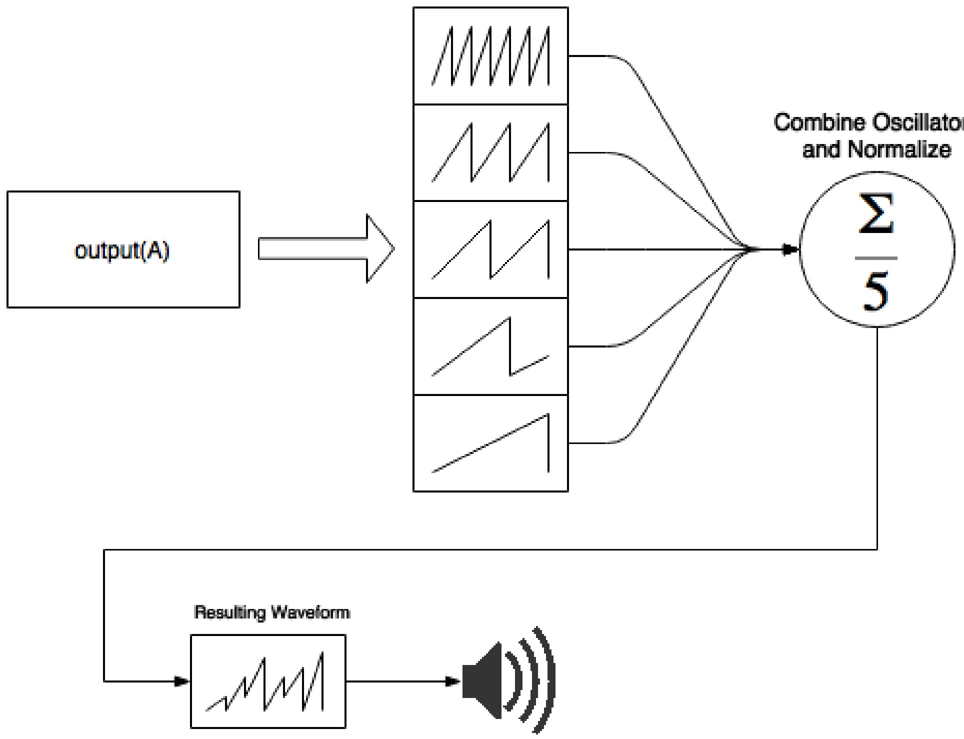


Fig. 2. Five oscillators have been declared and are active on the GPU, the results of the GPU calculations are then summed and normalized to avoid clipping. This final result is sent as output to the speaker/ Ableton Live.

III. Domain Specific Language

A Domain Specific Language will be created to control the synthesizer. It will provide a way to create instances of waveforms and to apply specific functions to them. The language is intended to be versatile enough that the necessary functions can be accomplished, while still operating on the waveforms at a high level. Table 1 shows functions that will be implemented in the DSL.

The primary idea of the DSL, is to allow for blocks of multiple oscillators to be declared at the same time. For instance, sin(5) would create five sine oscillators.The declaration of a block of waves is shown in Fig. 3. Other functions that transform (such as slide, setfreq, setamplitude) the waveforms will also be implemented. The language will allow for the control and manipulation of the oscillators at both a block and individual level. There will also be functions that create a block of waves with random values for the parameters (frequency, phase) assigned to each oscillator. There will also be commands related to timing such as wait.

The Domain Specific Language will likely be implemented using C++, using parsing libraries for C++ to aid in the interpretation of the language.

Functionality	Code	Example
Create a block of waves (synth blocks)	<block name> = <wave type>(<number of waves>,{<optional parameters>}),<additional waves>	synth A = sine(5), square(10) A = sine(5,freq=100,phase=0.33)
Raw blocks	These are like blocks, but instead of information about the oscillators, they store information about the raw amplitude	synth B = sine(5) synth C = square(5) raw A = B * C
Create singleton waves	<wave_name> = <wave type>({<optional parameters>})	wave a = sine(freq=5)
Other variables (strings, floats, ints)	<variable name> = <value>	num val = 30 string name = “name”
Delete a block of waves or a variable		delete A

Output the summation of the block to Max	output(<block>,<channel_num>)	output(A,1)
		output(A,false) (no normalization)

Concatenate two (or more) blocks		A = concat(A,B)
		or
		A = concat(A,B,...)

Manipulate entire block	<block>[]	A[]
Manipulate some waves in a block	<block>[<first>:<last>]	A[0:10]

Manipulate single wave in a block	<block>[<wave number>]	A[5]
Define a function (must define types for arguments)	function <function name>(<parameters>) = {}	function slide(A, -2, 2, 10) function slide(block,start,end,time)
Perform operations on blocks/scalars	<block/scalar> <operator> <block/scalar>	block + 10 10 * block 10 / 10 block * block
Wait for a period of time (in milliseconds) (possibly include this as a flag in functions like slide: wait until it's completed or don't)	wait <num>	wait 1000

Table 1. List of functionality and syntax for the DSL.

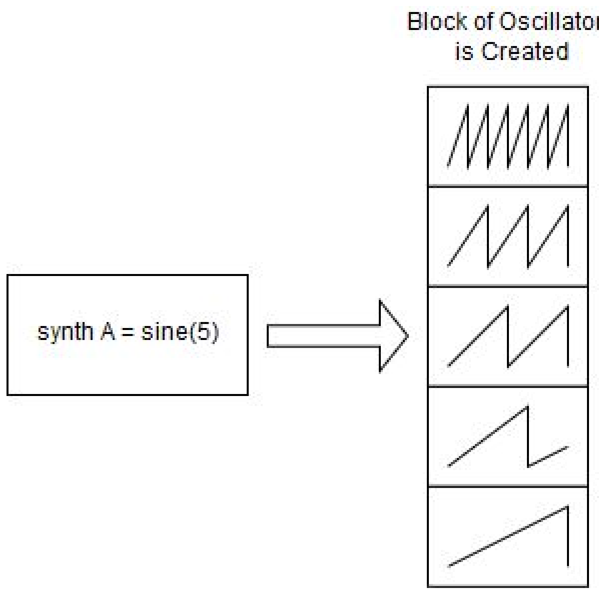


Fig. 3. sine(5) declares five oscillators which will be treated as a synth block.

The DSL will get commands either from a text file, which will run through the interpreter or through the Ableton Live Virtual Instrument and a GUI. It will also be able to be controlled directly from Max/MSP

IV. Ableton Live Virtual Instrument (stretch goal).

The Ableton Live Virtual Instrument will allow the synthesizer to be controlled through a GUI in Ableton Live. This will allow for users without programming backgrounds to use the Gigasynth. This will also allow the effects from the Gigasynth to be applied to a user's pre recorded sounds, as sound can be input to it through Ableton.

Ethical and Intellectual Property Issues

Ethical Concerns:

For the GigaSynth team, we are lucky to find ourselves in the enviable position where there are very few ethical concerns to be considered. In the realm of music production, not much weighs on our shoulders. That does not, however, render it completely safe. Aspects to be considered include the safe handling of user information, and various tenets of the Software Engineer Code of Ethics that apply. For the handling of user information, ethical treatment thereof is simply ensuring that it is kept safe from misappropriation by unauthorized individuals and not using it in an unethical manner (for all that entails), and generally upholding the practices of C.I.A. (Confidentiality, Integrity, and Authorized Access). As for the realm of the Software Engineering Code of Ethics, the passages therein still apply, especially concerning the ones on creating a product we know works without malicious intent or application, etc. etc.

Intellectual Property Concerns:

Facets to consider of the realm of intellectual property include licensing information, and the fruits of the labor of both the GigaSynth team and individuals who will utilize our product (i.e. who owns what was made). For licensing information, the team will be building our product on top of Max 7, Ableton Live, and Max for Live while utilizing ArrayFire for the GPU backend. In the usage of Max 7, the licensing agreement from their wraps up our concerns very succinctly. To quote their FAQ:

"I would like to sell my standalone application. May I do that?

Yes."

Thusly our concerns are assuaged. However, the EULA for Ableton Live and Max for Live, it's much more vague. As I understand the information presented in Licensing Agreement section 5a, the phrase "You may not translate, reverse engineer, decompile, disassemble, or create derivative works from the Software" refers to attempts to make a duplicate of Ableton, a program that is functionally identical made in an attempt to undercut their business. Fortunately for us, we plan to extend the functionality of their program by piping information out of it, into the GPU for mass-processing, and back into the program for playback.

Lastly, for the licensing of ArrayFire, it says in the link here:

https://github.com/arrayfire/arrayfire/blob/master/LICENSE

That so long as we do not claim the ArrayFire logo or brand as our own, claim their code base as our own, or use their name or brand in or as an endorsement for our product, we are fine to use it as long as their copyright information is included with their code.

As for the ownership rights of production, it's easy to just say "We own what we made, and music made using this product belongs to those who made it" and leave it at that.

Ableton Live EULA: https://www.ableton.com/en/eula/

MAX 7 FAQ concerning licensing: https://cycling74.com/support/faq-max7 under "I would like to sell my standalone application".

Max for Live was purchased by Ableton, and is now under their umbrella EULA.

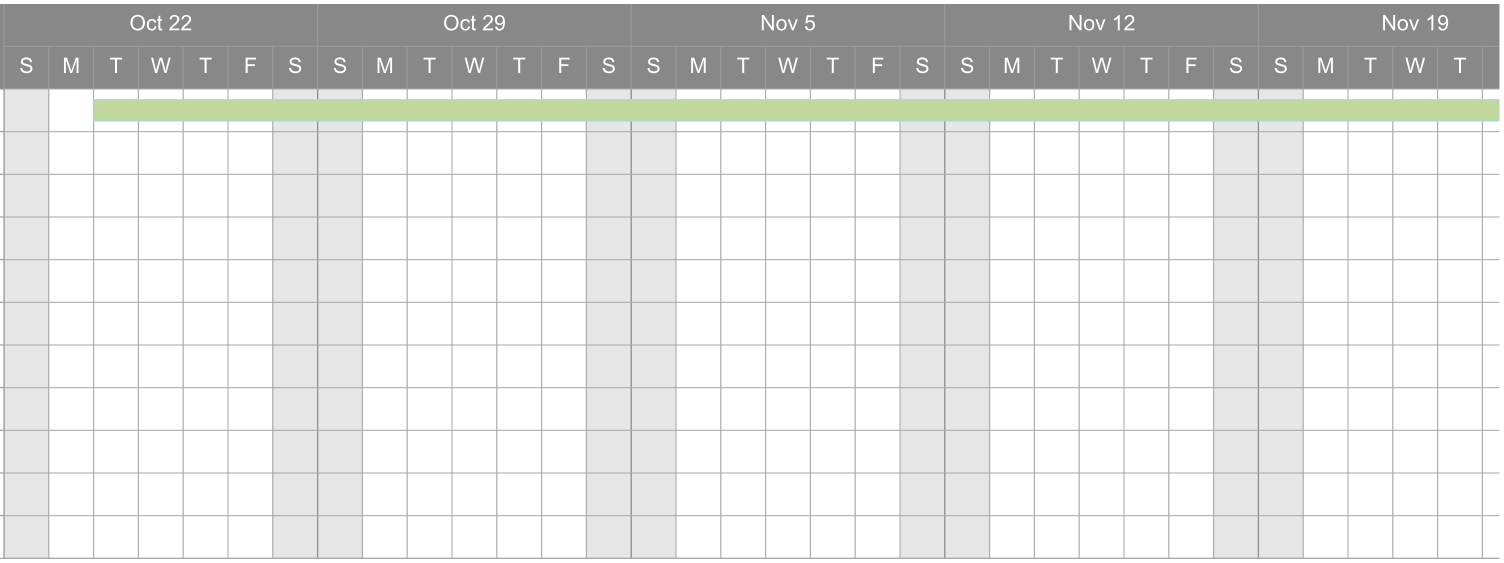
https://docs.google.com/document/d/1AAYjI8uHQoOxeFwMyL6KfLYzvaG8WgRkCYwx7ftXVqI/edit

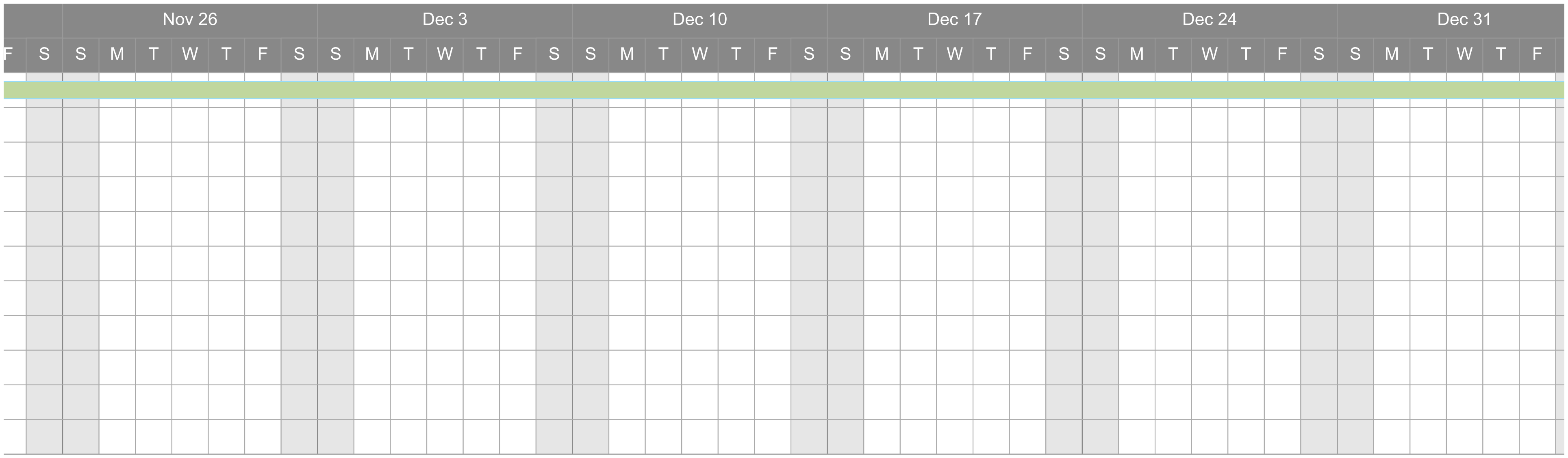
Change Log

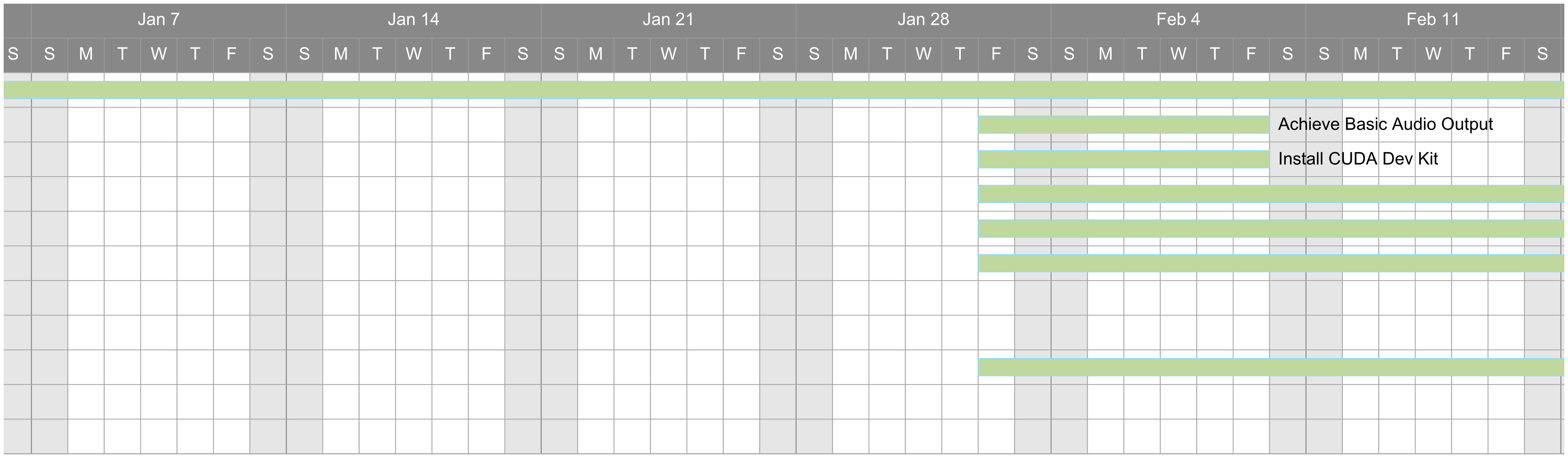
- Change GPU backend development dates
- Updated number of Max/MSP licenses required.
- Updated Ableton Live prices
- Updated Gantt Chart

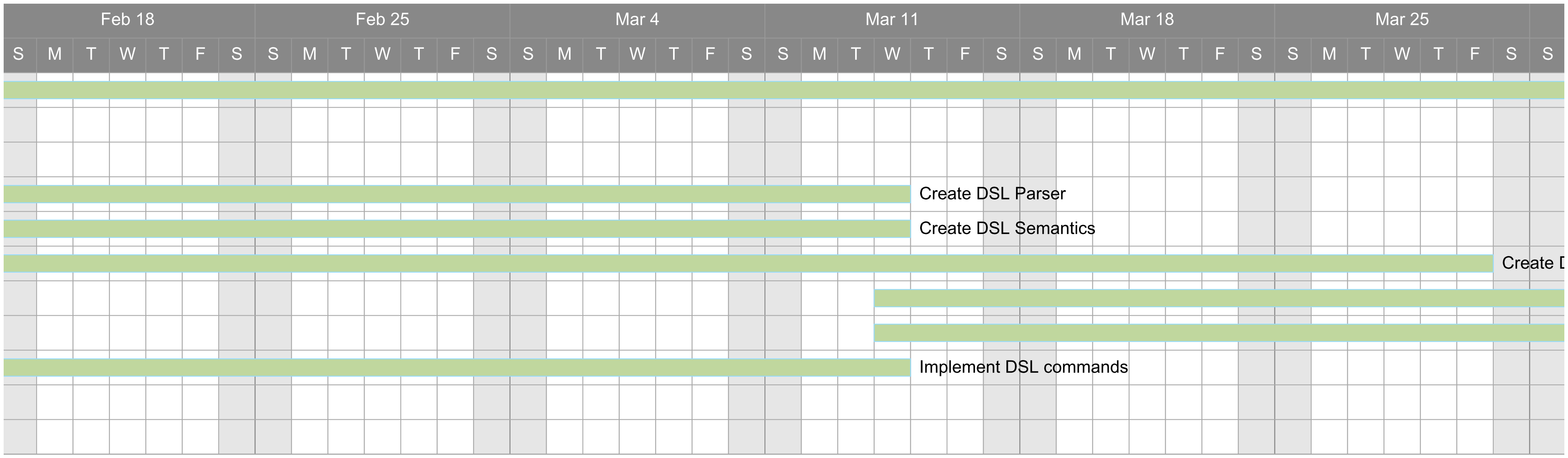
Team 11

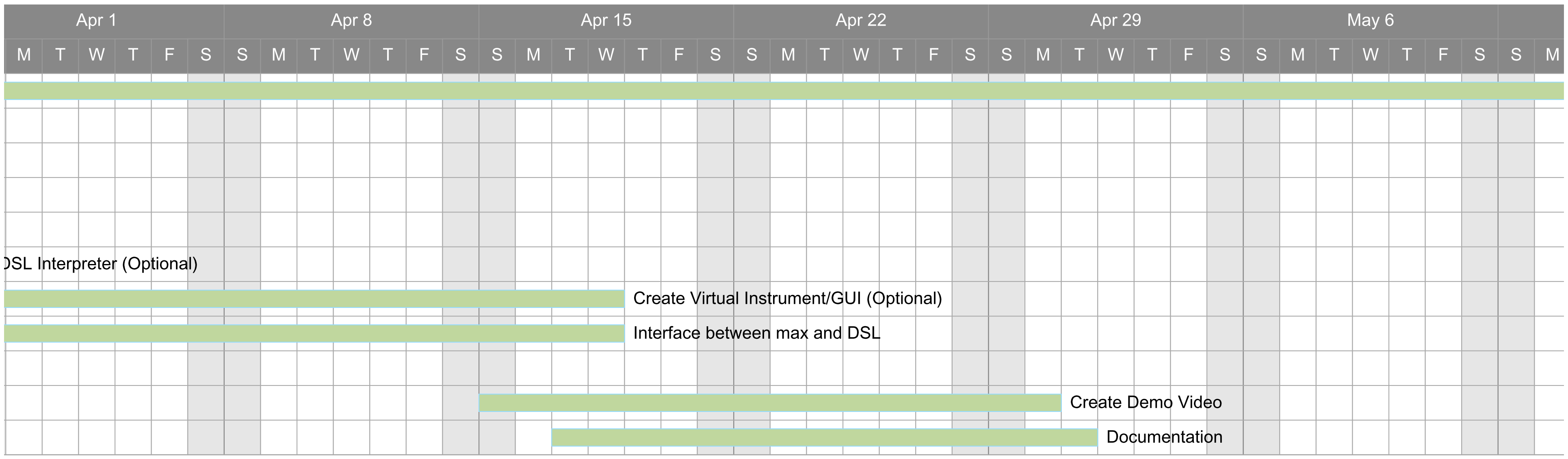
	Task Name
1	WAITING ON IT
2	Achieve Basic Audio Output
3	Install CUDA Dev Kit
4	Create DSL Parser
5	Create DSL Semantics
6	Create DSL Interpreter (Optiona
7	Create Virtual Instrument/GUI (C
8	Interface between max and DSL
9	Implement DSL commands
10	Create Demo Video
11	Documentation

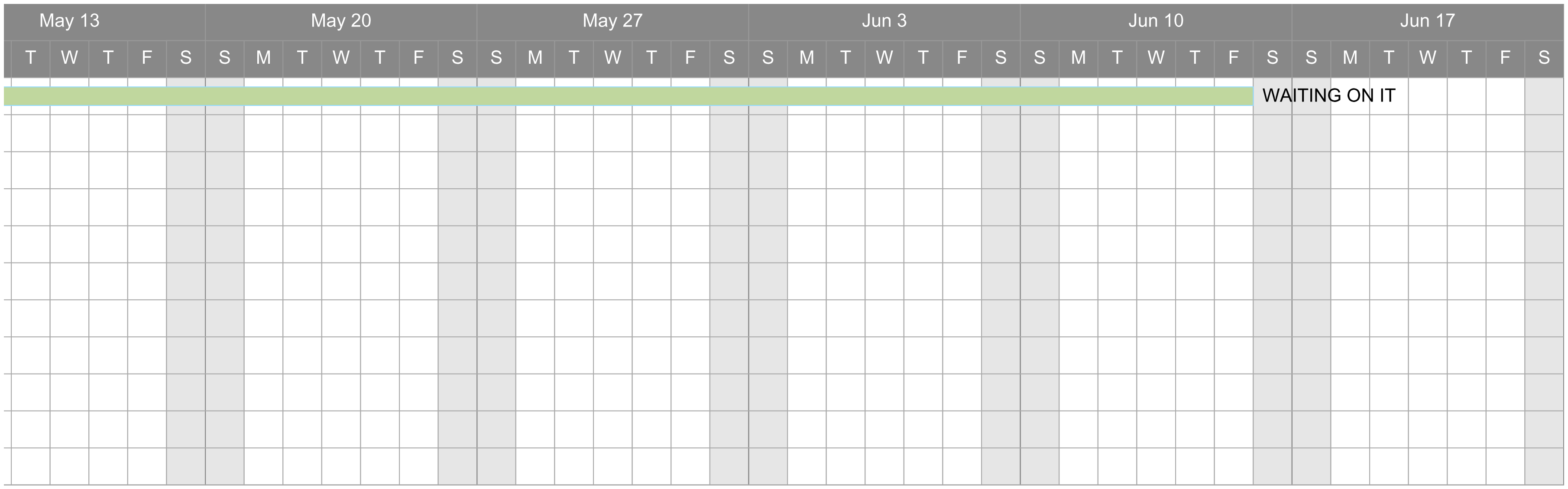



Nov 26
Dec 3
Dec 10
Dec 17
Dec 24
Dec 31
F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F


Jan 7
Jan 14
Jan 21
Jan 28
Feb 4
Feb 11
S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S
Achieve Basic Audio Output
Install CUDA Dev Kit


Feb 18
Feb 25
Mar 4
Mar 11
Mar 18
Mar 25
S M T W T F S
Create DSL Parser
Create DSL Semantics
Create D
Implement DSL commands


Apr 1
Apr 8
Apr 15
Apr 22
Apr 29
May 6
M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M
SL Interpreter (Optional)
Create Virtual Instrument/GUI (Optional)
Interface between max and DSL
Create Demo Video
Documentation


May 13
May 20
May 27
Jun 3
Jun 10
Jun 17
T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S
WAITING ON IT