IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

RE-EXAM CONTROL NO.:	90/009,299
RE-EXAM FILING DATE:	October 8, 2008
PATENT NUMBER:	5,886,274
PATENTEE:	Seer Systems
TITLE:	System and Method for Generating, Distributing, Storing and Performing Musical Work Files
ART UNIT:	3992
EXAMINER:	Albert J. Gagliardi
CONF. NO.:	6427
ATTY. DKT. NO.:	PA612REX

DECLARATION UNDER 37 C.F.R. § 1.131

I, STANLEY JUNGLEIB, AM THE SOLE NAMED INVENTOR FOR U.S. PATENT NUMBER 5,886,274 AND HEREBY DECLARE AS FOLLOWS:

1. I am presently the Chief Executive Officer of Seer Systems, Inc. a California corporation ("Seer"). I am informed and believe and thereupon allege that Seer is the assignee of the entire right, title, and interest in U.S. patent number 5,886,274 (the "'274 Patent"). I make this declaration in the normal course of business for Seer. I am also the sole inventor of the '274 Patent and make this declaration, as appropriate, in the context of the inventor of the '274 Patent.

2. I am informed and believe and thereupon allege that the underlying application for the '274 Patent was filed on July 11, 1997 and that said application was allocated U.S. patent application number 08/891,580 (the "'580 Application").

3. I am informed and believe and thereupon allege that on October 8, 2008, the U.S. Patent and Trademark Office issued a rejection that asserted certain claims of the '274 Patent were anticipated by U.S. patent number 5,734,119 (the "France Patent").

4. I have reviewed the France Patent and understand that the application that matured into the France Patent was filed with the U.S. Patent and Trademark Office on December 9, 1996.

5. I conceived, in the United States of America, the subject matter that is presently claimed in the independent and dependent claims of the '274 Patent prior to December 9, 1996 and, in any instance, no later than October 26, 1995. Attached hereto, as Exhibit A, is a diagram from my personal journal reflecting my conception of the claimed invention(s) and conception date.

6. Relative to my reduction to practice of the subject matter that is presently claimed in the independent and dependent claims of the '274 Patent, reduction to practice occurred, in any event, constructively no later than July 11, 1997 when the '580 Application was filed. I describe my diligent efforts following conception and culminating in constructive reduction to practice as follows:

(a) Obligated to Satisfying Duties to Company and Shareholders: At and before the time of conception, Seer was a corporation with 4 shareholders each of whom was an employee or officer bound by a non-disclosure agreement, myself included as CEO.

(b) Fully Devoted to Satisfying Those Duties: From 1991 and until June 17, 1998 when I suffered what was diagnosed as a heart attack at the age of 45, I worked to try to make Seer a success over 12 hours a day 6 days a week and, other than a one-month doctor ordered medical leave, took no vacations or breaks until 2001.

(c) Obligated to Satisfying Company Contract Duties: Before I conceived my invention(s), Seer relied for its economic survival on a contract with Intel to develop the Satie Music Synthesis project and thereafter an NSP-port project. To and including June 1996, we had no other independent business or immediate business prospects.

(d) Impact of Intel Cancelling Contract 1995: Approximately 7 months before the aforementioned date of conception, on March 17, 1995, Intel decided to buy out the remainder of Seer's Satie contract and only pursue the NSP-port Project. Approximately 3 months before the aforementioned date of conception, on July 12, 1995, Intel canceled the NSP project citing friction with Microsoft. Not only did these events jeopardize the economic future of Seer, these events also locked Seer and all of its personnel (myself included), out of any replacement work related to that Intel technology under a non-compete agreement with Intel.

(e) Response to Intel Cancelling Sustaining Work: From July 1995 until through 1999, I fully devoted myself to developing new business for Seer, starting with a new synthesizer product to replace the Intel work. By September, I had initiated negotiations with a variety of companies and individuals, including S3, ESS, Barry Borden, National Semiconductor, Stanford OTL, Euphonics and Opcode Systems. Unfortunately, while the value of our new synthesizer technology (technology that was entirely unrelated to my later conceived invention(s) as described in the claims of the '274 Patent) was recognized, none of these prospects materialized into a sustainable business proposition. It was this backdrop of disappointment with Intel, the scramble to development a replacement product and work, and lack of new business partners, that ultimately motivated me to think about and conceive what became the '274 Patent.

(f) Result of Efforts, Potential Replacement Partner: About a week after the date of conception, on November 6, 1995, Seer commenced a business relationship with the then most successful hardware sound card manufacturer, Creative Technologies, Ltd. Creative met with Seer to see a demonstration of our new product. The next day, Creative responded by saying they wanted to buy our technology and company. I put my full time efforts into negotiating an acceptable deal for the acquisition of Seer and its technology by Creative.

(g) Result of Creative Interest: Creative negotiated a letter of intent with us on January 11, 1996. That letter of intent included a "no shop" clause precluding us from negotiating with anyone else for a set period of time. While Seer negotiated the specific points of this acquisition with legal counsel, a roadblock developed: Intel exercised a non-compete extension on January 15, 1996. Intel's exercise of this clause caused delays that were not resolved until February 25, 1996. It then took until April 24, 1996 to complete the documentation, which included a technology licensing agreement and various related agreements. I worked full time to try to resolve the Intel roadblock, deal with Creative's disappointment in this regard, and complete the negotiations which remained active notwithstanding Intel's efforts to the contrary.

(h) Creative Contract and Effect on Seer and Me: While we did not sell the company or its technology to Creative, Creative effectively owned Seer full-time and prohibited any other work until their AWE64 product was completed. The AWE64 product was a full-time, "all hands on deck" effort. For the remainder of 1996, we worked on a variety of Creative projects pertaining to its flagship Soundblaster soundcard, the AWE64. Seer also worked on software products for Creative that could have fully replaced Soundblaster with software. Initially, Creative encouraged these products, but by September 1996, it appeared that Creative was no longer encouraging this development. In that timeframe, Creative began a series of what Seer viewed as failures to keep their contractual promises. This culminated in Creative 'walking away' from Seer in January 1998 leaving the company 'high and dry' with large sums due for work performed and under a promised loan Seer needed to survive.

7. The events summarized above in paragraph 6(a)-(h), prevented me from doing anything beyond taking the following actions to reduce to practice what became the '274 Patent. Those actions were:

(a) On September 23, 1996, I was able to arrange with John S. Ferrell of Carr & Ferrell LLP to speak with him about protecting my intellectual property relative to what became the '274 Patent. Mr. Ferrell wanted to meet in this regard. Given my duties to Seer and Creative as describe above, that meeting needed to be scheduled months in advance for December 16, 1996.

(b) A copy of Mr. Ferrell's letter of December 17, 1996 regarding our meeting and the engagement is attached hereto as Exhibit B.

(c) On January 7, 1997, I further discussed protecting my invention with the associate attorney that Mr. Ferrell assigned to the engagement for what became the '274 Patent, Mr. Marc Sockol.

(d) On January 30, 1997, Mr. Sockol returned the original of the document that put my diagram of my invention(s) (Exhibit A) into words. That document is dated December 13, 1996. A copy of his correspondence to me of January 30, 1997, and said document, is attached as Exhibit C. The document is important as it includes excerpts sections of the diagram. See, for example, 12/13/06 document, page 7.

(e) On February 19, 1997, Mr. Sockol sent me a first draft of the application for what became the '580 Application. A copy of his correspondence to me of that date is attached as Exhibit E. I reviewed that draft as quickly as I could given my primary duties to Seer and Creative. I cannot presently locate any records reflecting when I responded to Mr. Sockol, but I believe it was about a month later on March 18, 1997 based upon notes that it appears he left on Exhibit D.

(f) On April 9, 1997, Mr. Sockol send me a revised draft of what became the '580 Application. A copy of his correspondence to me of that date is attached as Exhibit E. Again, I responded as quickly as I could given my other, primary obligations to Seer and Creative. I cannot presently recall when that occurred but believe it was by June 1997.

(g) So far, I have not been able to locate the 1996-1997 billing records from the Carr & Ferrell firm. (Seer closed its offices in 1997 and its records may be lost). But I have been able to find Seer bookkeeping records reflecting Carr & Ferrell's charges for the following work that I understand was for what became the '274 patent as follows: bill paid March 17, 1997 ($8000), bill paid April 1, 1997 ($8000) and bill paid May 14, 1997 ($3130). Thereafter, our books only make generic references to "legal services" for the over $9000 we paid for combined bills in June 1997 followed by no billing references until September 1997 when we show we paid several thousand dollars for bills with a total of $105 relative to what became the '274 patent. The records then appear to be generally for patent work, including $4059.69 for work in July 1997, but there is no clear identification as to whether the work was for what became the '274 patent, although I believe it likely same would be true.

8. As summarized above, from my initial contacts with Carr & Ferrell in September 1996 until the filing of the '580 Application, I worked diligently with them to attend to the constructive reduction to practice of the claimed inventions of the '580 Application.

9. Additional evidence of conception, diligence, and reduction to practice may be available. Notwithstanding, the presently submitted declaration and exhibits are believed to be sufficient for the purposes of evidencing conception and reasonable diligence in constructive reduction to practice of my claimed invention(s). On behalf of the assigned of the entire right, title, and interest, I expressly reserve the right to present such further evidence as may be appropriate in light of ongoing efforts to locate documents, such as the Carr & Ferrell billing files, that may refresh my recollection and otherwise further substantiate their efforts in this regard. The present submission is in no way meant to be interpreted as limiting or indicative of all evidence in this regard that may be available.

10. I declare that all statements made herein of my own knowledge are true and that all statements made herein on information and belief are believed to be true and, further, that these statements were made with the knowledge that willful false statements and the like are punishable by fine or imprisonment, or both, under 18 U.S.C. § 1001 and that such willful false statements may jeopardize the validity of this application or any patent issued thereon.

September 21, 2009

Stanley Jungleib

951026

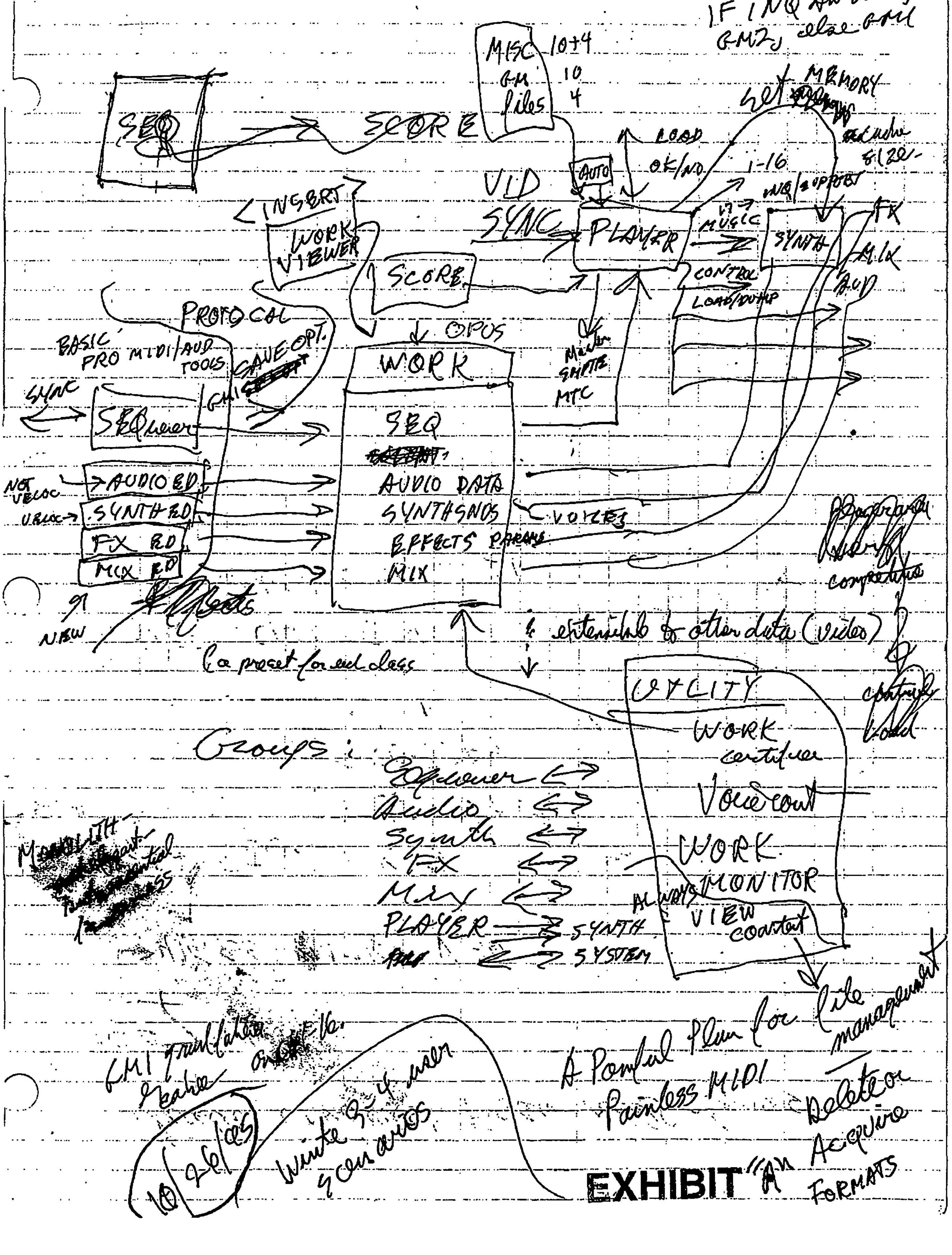


EXHIBIT "A"

CARR, DEFILIPPO & FERRELL LLP
ATTORNEYS AT LAW

WRITER'S DIRECT DIAL NUMBER
(415) 812-3408

December 17, 1996

Stanley Jungleib
Seer Systems
108 Portola Road, Box 137
Portola Valley, CA 94028-7899

Re: Seer Music 2000 Patent Application

Dear Stanley:

Thank you very much for our meeting this past Monday to discuss your Seer Music 2000 invention. I have enlisted the help of Marc Sockol, one of the patent attorneys in our firm, to provide support in preparing and filing this patent application. We have targeted a filing date internally of mid-February for filing your application. Over the next few weeks, Marc will be reviewing the video tape, invention notes, and your book, and will be preparing a set of patent figures related to this material. When completed, Marc will be forwarding you a copy of the figures for your review and comments. Once you have reviewed the figures, we will begin drafting a specification describing your invention.

If you have any questions regarding this process, or any other matter, please don't hesitate to contact me.

Very truly yours,

J L Ferrell

John S. Ferrell

JSF/amz

EXHIBIT "B"

2225 East Bayshore Road, Suite 200, Palo Alto, CA 94303
Telephone (415) 812-3400, Facsimile (415) 812-3444

CARR, DEFILIPPO & FERRELL LLP
ATTORNEYS AT LAW

WRITER'S DIRECT DIAL NUMBER
(415) 812-3407

January 30, 1997

Stanley Jungleib
Seer Systems, Inc.
33-Tintern Lane #5
Portola Valley, CA 94028

Dear Stanley:

Per our phone conversation on January 30, 1997, enclosed please find a copy of your disclosure materials entitled "Distributed Music," draft #1.09.

If you need anything else, please don't hesitate to call.

Sincerely,

Marc A. Sockol

Marc Sockol

MAS:slc
Enclosure

EXHIBIT "C"

2225 EAST BAYSHORE ROAD, SUITE 200, PALO ALTO, CA 94303
TELEPHONE (415) 812-3400, FACSIMILE (415) 812-3444

Distributed Music

DM

A Synthesis and Sound Design System

for Consumer Multi-Media

Draft #1.09

by Stanley Jungleib

TABLE OF CONTENTS

1. INTRODUCTION

This chapter provides some context for the Distributed Music (DM1) system, explains the overall rationale, and surveys its main components.

1.1. The Failure of General MIDI 1

In October of 1991, General MIDI Level 1 (GM1) became a recommended practice of the MMA and JMSC (now AEMI). Basically, GM1 defined certain aspects of multi-timbral synthesizers so that they would play back the same MIDI file similarly.

While an improvement over the previous system of *Sound Blaster* synthesis, the reality of GM1 implementations lead to several problems. GM1 left enough aspects undefined to create ample problems for composers and consumers expecting consistency and ease of use. For example, there are at least five flavors of GM1 data files in circulation; and to manage them you need as much mode and channel expertise as with original MIDI. For a detailed discussion of the strengths and weaknesses of GM1, please see Jungleib, *General MIDI* (1995: A-R Editions).

Since the release of GM1, discussion about a GM2 occurred from time-to-time, typically with calls for downloadable sounds. But the MIDI industry has thus far been unable to evolve a new standard approach. I can say this with certainty because I chair the MMA committee that is supposed to extract the new standard from MIDI manufacturers. Quite simply, few seem willing to come forward and work together on this.

As MI veterans Seer Systems thinks that the current GM industry situation and trend is very bad, and it pains us deeply. For example, while the Web explodes, MIDI should be marching triumphantly through the arches of the multimedia Internet. But because it still remains so esoteric and problematic, GM1 instead is losing ground to the false idol of compressed audio—with all its attendant distortion and degradation.

Therefore it is our conviction that a new GM is necessary, one that makes even more demands upon its practitioners to implement the strict guidelines required to achieve the turn-key playback which GM1 promised but fails to deliver. This document presents that system, top-down.

1.2. The Basic Idea

The design requirements of any new programmable music delivery system purporting to useful encompass a commitment to the total system. Thus, DM1 is a far broader concept than traditional MIDI or GM1 and it offers equally broad musical rewards.

The bird's-eye view of what this is about is that through our unprecedented technology Seer cheaply gives the same instruments to the listener that the professional musician used to make the music in the first place. The instruments are software, which we deliver via the OS or application, CD or network, and which we refresh sporadically or constantly. Some might call this development the mass distribution of virtual studios because we recreate the equivalent of an entire studio in the listener's local system. As a result, playback is predictable because the composer controls the synthesizer subsystem. But ample opportunities remain for the MIDI consumer to participate interactively in the control stream.

With all the sound creation instruments locally installed, the information required to communicate the music itself approaches 1/1000 the bandwidth of the equivalent typical stereo audio program. We take a tremendous load off of the multimedia channel. Yet our fidelity is not constrained by network bandwidth nor by physical cost. Intelligent defaults and scalable customizations serve the content developer. And the control language we use gives us all the interactivity we could desire.

For all these reasons, DM1, other something very much like it, is an excellent candidate for a standard music distribution system of the future.

1.3. Downloadable Audio

Up until now there has been a dichotomy between audio and MIDI. You either used one or the other. The synthesizer always plays from RAM (or ROM), because the engine needs to have fast access to the entire waveform particularly if transposition is going to occur. In contrast, long waves typically play from disk.

In the context of definable instruments, however, the lines start to blur. By easily installing waves into the synthesizer the user or sound designer can play audio without the system interruption that might be caused by having to play waves from disk using standard techniques. In effect MIDI becomes a general control system for audio by offering real-time parametric control over the voice and its manipulation via the sequence. For example what was formerly a mere wave sample can now easily benefit from real-time pitch shifting, looping, enveloping, filtering, and effects processing.

The move towards downloadable instruments offers special advantages to the game and content sound designer as well. Bringing the audio into the synthesizer also happens to eliminate problems associated with attempting to synchronize audio with MIDI outside of the synthesizer.

1.4. Considerations Going Forward

DM1 needs to take such a tremendous burden off of the multimedia consumer that it can't afford to limit itself to GM1's concept of MIDI political correctness. It must be much more aggressive. We're starting over.

Except for backward compatibility, there simply isn't much of GM1's architecture left to salvage going forward. To soothe any concerns and allow you to concentrate on understanding DM1, let me tell you to rest assured that GM1 is taken care of—grandfathered in. Practicality must prevail, so GM1 is not only retained, but automated and improved. The most explicit way to demonstrate this in the system is that Channel numbers 1-16 (0-FH) are reserved for GM1 use. So it is always clear what kind of data rules we are living under: DM1 sound rules only apply to Channels 17 and up.

DM1 does not attempt to retrofit any new requirements on GM1 compliant devices. While strongly recommended, it is up to the composer whether they want their DM1 Work to playback on GM1 devices or not. On all channels, Programs 1-128 are the corresponding Sound Canvas sounds. Drum kit 1 is the Sound Canvas default kit. So, you see, we do respect our elders.

In designing a new system, not to be ignored is the business and economic context. We assume the basic classes of electronic music editing tools with which composers and sound designers are already familiar. DM1 doesn't ask people to switch from their favorite tools; it must be understandable at large. You don't want to force early adopters to abandon a decade of MIDI paradigms or force them to change the way they work. For example, a geodesic dome is elegant but collapses when a single strategic panel fails; whereas a standard wood-frame structure is less elegant, but can withstand lots of stresses, including that of tradesman maintaining it with standard tools. In this sense, DM1 is more like a wood-frame home than a dome.

After sketching out this big picture, the current author will likely lose interest in implementation particulars. I'll try to leave this document in a state ready for a knowledgeable Synthesizer and MIDI programmer to implement the coding required.

Actually, when we start talking about 1000 channels, automatic downloadable sounds, and integrated playback from disk it's not clear that we are talking about MIDI anymore. So the term is used loosely to refer to whatever coding structure we end up with.

1.5. Testing and Certification

I don't have any illusions about testing. It can't really be done on a wide scale until consumer critical mass is attained. The concept of Approved Tools is introduced in a few places below; probably not consistently. What does this mean? Wishful thinking: Self-regulation will likely prevail.

The next best thing to testing is to bring the composer's sequencer into the process so that data is always saved in a determined way. This will eliminate the bulk of playback problems which make people cry for testing.

1.5.1. Variations are Endemic

Another way to reduce the need for testing is to resist the actually baseless idea that all instruments have to sound alike in the first place.

To be sure, effective volume is not a simple thing: the overall loudness of an instrument depends substantially on its timbre, or frequency spectrum. GM1 explicitly says you can use any synthesis method you want—which obviously affects the resulting timbres enormously. Thus, right out of the gate manufacturers' powers to cross-match their synthesizers were inherently limited.

When you consider the enormous range of equalization playback conditions and speaker quality in the marketplace, then steadfastly holding to the notion of the perfect sound or mix within the context of GM seems to us a bit extreme.

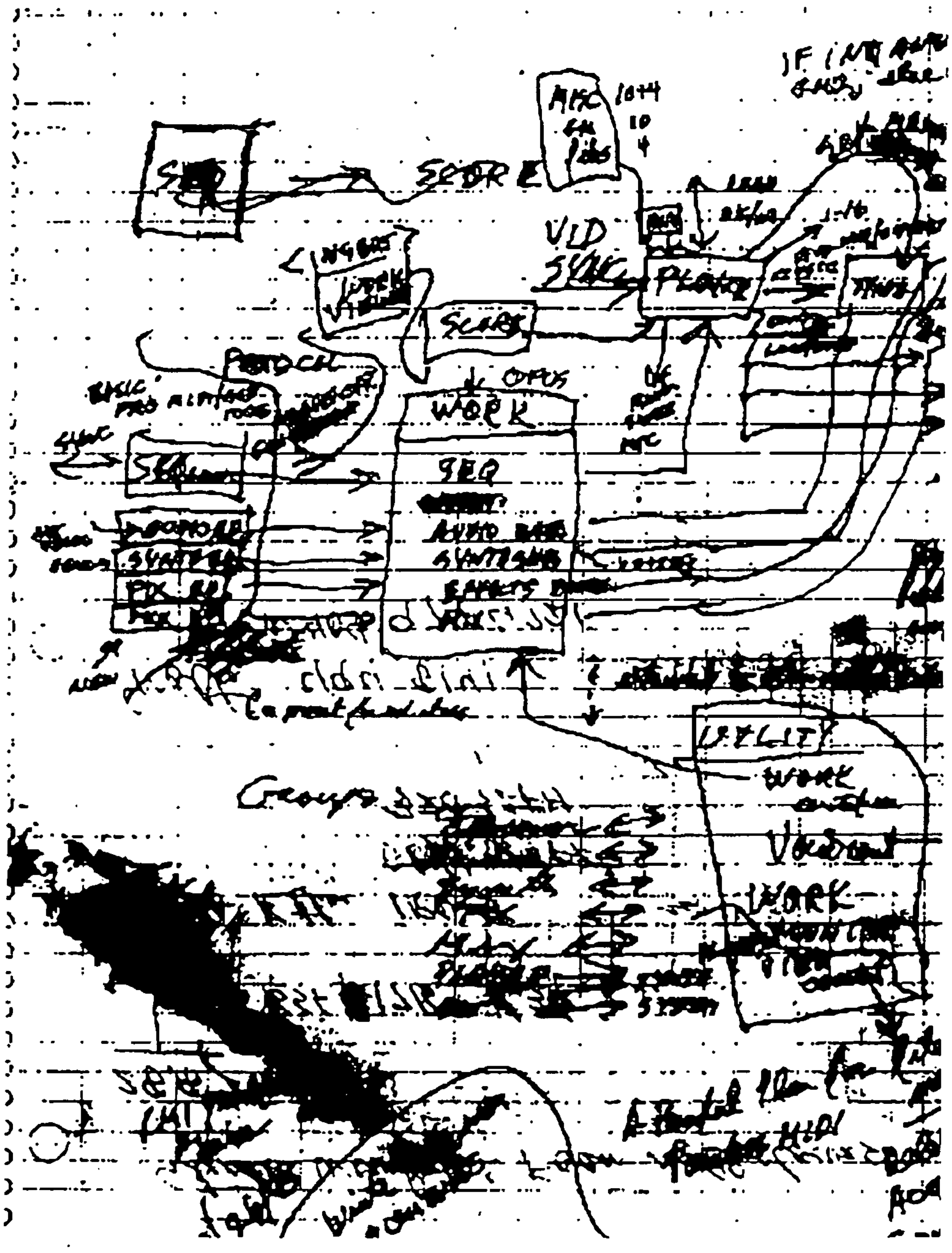
Fact is, professionals deal with these challenges daily: a strong tune withstands mix variations, as does a strong mix withstand wide speaker variations.

So, under DM1 it is OK if like some rare combinations of golden-ear audiophile preamps and amps, some combinations of DM1 Works and Synths sound better together than others. That encourages experimentation and competition.

1.6. System Diagram

Figure 1 diagrams the system in original sketch, if nothing else. I'll redraw it when I can. But for now this will have to do.

Figure 1 A Painful Plan for Painless Playback



There are four main functional areas in the DM1 system:

1.7. Composer's Tools

The composer is the input to the Composer's Tools. The Tools transform audible intentions into reality. Their outputs are file formats or data sets that have a track record. Composers and sound designers are used to an assortment of tools such as sequencers and audio editors. Engineers and producers are used to mixers and effects devices. There are millions of hours behind these modules already. So the model of their use should generally be preserved.

However, we ask of them some new features of the DM1 generation. In general they must be able to write their data to our Work Manager. The Work Manager ensures that the data is registered correctly in the Work file. The Work Manager then echoes editor commands to the synthesizer subsystem so that the edit effect can be heard immediately. Again, the tools don't have to be integrated, because the output itself (the Work) is. This leaves the composer to stick with what they already know (as long as it is slightly updated).

1.8. The Work and Work Manager

The Work is the distributed musical file; the song, the piece, the composer's product. The Work file contains all sequence information but it also contains all synthesizer and audio information required to recreate the song within the DM1 playback system.

The Work Manager lets you view and maintain the Work contents directly. The Work Manager also has a "THROUGH" function for passing editing data to the Synthesizer subsystem.

1.9. The Player

This is the consumer playback tool. As before, the player sends MIDI data to the synth. Unlike before, the Player controls the dynamic loading and unloading of sounds from the Work to the Synthesizer subsystem. The challenge here is to deal with the variable RAM size that will be supported in the playback devices. In some cases this function will be built in to the application or game.

1.10. The Synthesizer Subsystem

This produces the complex sound. It is named a subsystem because it may contain a variety of hard or soft technologies to meet its duties as a synth, processor, mixer, and codec.

The DM1 synthesizer begins with a ROM synth patterned after the GS Sound Canvas. But additionally it emphasizes a RAM-based synthesizer which is capable of rendering fully customizable music, including samples of arbitrary length, custom effects and mixer routings.

Notice that for fastest response, real-time sequence data comes from the Sequencer or Player directly to the Synth subsystem, without passing through the Work. Real-time MIDI processing is implemented in the synth's front end.

At risk of sounding repetitive, in some cases the synthesizer will be built into the application or game.

Each of the four main areas just described and their sub-functions are covered in their own chapters below.

1.11. OS Communication

The OS (Windows, MacOS) can help the system immensely by signalling to DM1 when it is OK to load sounds from disk. Ultimately this message comes from a game or app telling the synth to load either generally or specifically because the app knows the time is right.

Another class of information from the OS relates to real-time playback control. For example, the game or app must sometimes reach outside itself to easily change tempo.

1.11.1.Networking

For simplicity, not covered in this initial version is the placement of the Internet within the system chart. However it is easy to name positive benefits that could result from stretching the distance between any of the four main blocks. For example, Work data can be streamed into the system continuously from the Internet.

Next time.

1.12. Scaling and Boundaries

A major challenge of DM1 is to deal with scalability upwards from minimum standards. Scalability is required to accommodate power increases resulting from competiution and growth. The system wants to make room for upward growth so that it can employ future developments smoothly within the installed base.

Of particular concern is how one deals with boundary conditions. There are two realms of boundaries: processor limits which curtail polyphony, and RAM limits which compromise timbre. The following describes the general approach DM1 takes to each issue.

1.12.1.Polyphony-Processing

The point of the polyphony rules are that when polyphony does exceed the minimum 32, a minimum synthesizer will know what notes to play and not play. Simultaneously, someone can offer a synthesizer with 64 voices that could play the same Work, only with a larger effect.

To allow this, the composer simply breaks up the optional parts into tracks, and arranges the optional tracks in priority.

By allowing up to 272 (256 + 16) channels, this method would let the synthesizer realize realistic orchestral renderings, incorporating up to 280 (256 + 24) voices. This could be done in real-time with sufficient CPU capabilities, or off-line, compiling the MIDI score into a CD-quality audio track.

1.12.1.1.First method

The Sequencer will warn and discourage the composer from saving any DM1 sequence which has tracks enabled that exceed the legal limit for parallel voices at any one time. The legal limit setting will default to 32, but will be user adjustable. (This way, large scale composers don't always have to be annoyed by small-scale defaults.)

1.12.1.2.Second method

If the composer overrides the legal limit they themselves have set, the composer has three choices:

1.12.1.2.1. Accept the default track priority where DM1 tracks are treated in numerical order (17-1024).

1.12.1.2.2.Attach one table of track priorities that serve throughout the Work.

1.12.1.2.3.Attach any number of track priority tables to specific points of the sequence, thus changing the priority dynamically.

1.12.2.Timbre-RAM

In addition, the sequencer will default to insist that for each custom sound a GM1 counterpart selection (or Mute) is assigned. This will ensure that the musical part gets played even on an instrument that has no RAM available to it. Wherever available RAM is less than what is called for by the Work, the Player will follow the effective track priority tables in deciding which sounds to automatically substitute GM1 ROM sounds for.

Typically, the audio channel(s) receive absolute priority. So if memory is too small to allow any custom sounds, the music will still play with GM1 ROM sounds.

2. THE WORK

This chapter actually looks at two of the main components together; the Composer's Tools and the Work. The Tools produce data structures that make the Work. Therefore to discuss the Tool's output is to describe an element of the Work.

The Work has its own Manager maintenance function. The Work Manager essentially allows real-time edit buffering along with file maintenance. Applications or drivers may wish to incorporate some of these functions. The work manager acts as a filter for data sent to the work by the tools

The Manager basically allows the Composer's tools to export their data without having responsibility for the integrity of the whole Work.

2.1. The Need for a New File Format

One doesn't approach this question lightly. Creating a new file format is to be avoided as possible. I first thought that we would have a standard GM Score that pointed to a custom sound bank, and that if the two got separated, it would be fail safe since GM1 fall-back sounds would always be in place

Now I've concluded it's not possible to make that system really work. Although it is tempting to break the Work down into a spectrum of optional file types, this would mean chaos for the user. We only want one file representing the Work.

So, we create a new file, the Work, which holds all the incidental file types. We require communication occur through the Work Manager application. The Manager combines the various files into a monolith, manages memory layout, and maintains the legitimacy of the Work file.

DM1-labeled sequencers will have specific requirements for file checks when saving. These checks will check for excessive NoteOns, illegal data, compliance in general, and linkage to custom banks. The Work may still call for any alternate banks of sounds, drum kits, and effects.

2.1.1. SoundFonts

2.1.1.1. Likely to be inserted in each place where applicable below. As far as we know It is a subset of what we are after because it doesn't cache voice types.

2.2. Mixers > Mixes

2.2.1. Global Mix Data

Data	Variables	Default	Current
Topology Type	0 = Preset 1 = Custom with Fallback to Preset 2 = Custom Only	0	0
Preset Topology #	0 = No Connections 1 = RAP-10 2 = QUAD 127 = Last Preset 128 = Custom 1 ... 1023 = X	1	

2.2.2. 0 - Null Connections

2.2.3. 1 Stereo Audio Only

2.2.3.1. Diagram

[RAP-10 block diag]

2.2.3.2. Data

Mute	0 = OFF (Sound On) 1 = ON (Sound Off)	1
Synth Volume	0-255	255
Synth Pan	0-255	127
Audio Volume	0-255	255
Audio Pan	0-255	127
Audio Reverb Send	0-255	127
Audio Chorus Send	0-255	0
Reverb Return Level	0-255	32
Reverb Return Balance	0-255	127
Chorus Return Level	0-255	0
Chorus Return Balance	0-255	127
Load Time		

2.2.4. 2 Sound Canvas (no Audio)

2.2.5. 3 RAP-10 Emulation

2.2.5.1. Diagram

[RAP-10 block diag]

2.2.5.2. Data

Mute	0 = OFF (Sound On) 1 = ON (Sound Off)	1

Synth Volume	0-255	255
Synth Pan	0-255	127
Audio Volume	0-255	255
Audio Pan	0-255	127
Audio Reverb Send	0-255.	127
Audio Chorus Send	0-255	0
Reverb Return Level	0-255	32
Reverb Return Balance	0-255	127
Chorus Return Level	0-255	0
Chorus Return Balance	0-255	127
Load Time		

2.2.6. 4 QUAD

2.2.6.1.Diagram

[Block diag]

2.2.6.2.Data

Mute	OFF
	ON
Synth Volume	0-255
Synth Pan	0-255
Audio 1 Volume	0-255
Audio 1 Pan	0-255
Audio 1 Reverb Send	0-255
Audio 1 Chorus Send	0-255
Audio 2 Volume	0-255
Audio 2 Pan	0-255
Audio 2 Reverb Send	0-255
Audio 2 Chorus Send	0-255
Reverb Return Level	0-255
Reverb Return Balance	0-255
Chorus Return Level	0-255
Chorus Return Balance	0-255
Load Time	

2.2.7. Custom Topology #129

2.2.7.1.Diagram

[Block diag]

2.2.7.2.Data

Mute	OFF
	ON
Synth Volume	0-255
Synth Pan	0-255
Audio 1 Volume	0-255
Audio 1 Pan	0-255
Audio 1 Reverb Send	0-255
Audio 1 Chorus Send	0-255
Audio 2 Volume	0-255
Audio 2 Pan	0-255
Audio 2 Reverb Send	0-255

Audio 2 Chorus Send	0-255
Reverb Return Level	0-255
Reverb Return Balance	0-255
Chorus Return Level	0-255
Chorus Return Balance	0-255
Load Time	

2.2.8. Mix

The song is mixed by adjusting basic volume levels of each synth channel plus up to eight audio channels, equalization, effects sends and returns.

This step contributes a mix topology and parameters to the Work.

Having finally subdued the mixer beast some composers feel that if a playback module adjusts the mix they composed with at all, then a tremendous artistic injustice has occurred. However, it is possible to lighten-up on this issue in cognizance of some basic realities. First and foremost, GM itself does not corroborate your argument. Nowhere is it written that a module has to sound like anything in particular.

2.3. Processors > Effects

This custom thinking also extends to the realm of effects, where most companies are adopting programmable DSPs or general algorithms. In other words after designing the symphony you would also design the room in which you wanted it played.

Reverb and chorus along current GS lines should be declared plus obvious ones like distortion, then as we did with instruments, allow for the free definition of any effect and the assignment of any controller number to control its depth.

2.3.1. Reverb 1 (Sound Canvas)

2.3.2. Chorus 1 (Sound Canvas)

2.3.3. Channel Sends are Here

2.3.4. Create Effects

By default we intend the Sound Canvas Reverb and Chorus settings. A compressor that can be used for auto-ducking is a crucial path. Plus a dozen other standard ones, like Distortion for starters.

This step contributes a set of effects algorithms and parmeters to the Work.

2.4. Voice Editors > Instruments

Another level of information must instruct the synthesizer how to assemble the instruments into a playable ensemble selectable by customary Program Change messages; which channels they respond to; thus, which instruments end up being used together to make one multisampled instrument.

The combination layer would assemble the instruments into a multi-patch or multi-sampled instrument. It is not totally essential to establish this layer, as it is easy to distribute the same parameters to the instruments. However, the layer does allow you to easily conceive of using the same instrument in different places. This is tougher to arrange if instruments declare their own mapping.

The combination needs to have its own selection number above the program changes themselves. (A high Bank number could be used to select banks for Combination settings.)

you can keep the instrument bank smaller by including only ranges that will be played- not necessarily the entire mutli-sampled instrument

Through downloadability, synths gain variety. But they do not gain independence. We still want them to make similar noise from the same patch information. So there is still the matter of converting from MIDI data to audible results. At first look it seems we have made no progress; we have just passed along the definition problem higher up the food chain. But this is not the case. There is a difference: To get the ball rolling the definition we must make is of the specific behavior of a specific superset algorithm, namely a wavetable front-end with subtractive processing attached. (To accommodate any subsets of this voice, like waves alone, any module could be disabled.) The gain is that once we sit down and do this, it will serve all the particular settings (the sounds) themselves generally so that we won't have to do it specifically.

2.4.1. Number

2.4.1.1.1-128 like GM

2.4.1.2.129-10,000 new

2.4.2. Name

2.4.3. Macro

2.4.3.1.Layer

2.4.3.2.Split

2.4.3.3.Drum

Instruments can play ranges as small as one note. Therefore the same basic combination object can be used to create drum kits (range =1 note) or multisampled melodic instruments (range = several notes). The system knows to use drum combinations for drum channels.

2.4.4. Mode

2.4.4.1.Polyphonic

2.4.4.2.Monophonic Retrig

2.4.4.3.Monophonic Legato

2.4.5. Number of Patches

2.4.5.1.1-128

For synths you only need 1 or 2
For drums you need at least 47
128 puts a different sound on each key.

2.4.6. Per-Patch Parameters

2.4.6.1.Patch

2.4.6.2.Patch Name

2.4.6.3.Level.

2.4.6.4.Transpose

2.4.6.5.Detune

2.4.6.6.Key Low

2.4.6.7.Key High

2.4.6.8.Velocity Low

2.4.6.9.Velocity High

2.4.6.10.Pressure Low

2.4.6.11.Pressure High

2.4.6.12.Controller N Low

2.4.6.13.Controller N High

2.4.6.14.Initial Delay

2.4.7. GM1 Fallback Instrument

If the instrument number is 1-128, the fall-back should equal itself.

This field only serves a purpose for instruments 129 and up.

2.4.8. Size

2.4.9. Load Time

This is the advance warning in ms that the Synthesizer needs to be able to discard unused sounds and load this sound from disk. This value can be calculated by the Work Manager for a given typical machine.

2.4.10. Employ Custom Instruments

In this step you make the synthesis more satisfying and convincing by replacing standard GM1 sounds that "aren't quite making it" with custom sounds edited on your synthesizer.

Synthesizer sounds may include percussion sounds that are mapped melodically to the keyboard.

This step contributes custom synthesizer sounds to the Work, These may be various synthesis methods employing greater or less amounts of PCM excitation waves, algorithms, and parameters.

2.5. Voice Editors > Patches

Patches generate sound, but they do not constitute complete instruments.
You do not play patches. But you do build instruments out of them.
Patches are numbered from 1-10,000, have Name and Description fields.
The first field is the Patch type, from which the rest of the patch paramters are decided. The Patch type is chosen by the user for the basic qualities of what is to be played. Each of the voice types have a specific loading value, therefore the user should choose the lowest value voice that gets the job done.

2.5.1. 0 = Mute

2.5.2. 1 = Audio Event

Choose this Sound type if you just want to play a sound file unprocessed. For the convenience of game sound designers, looping is included here. But if you want transposition choose a synth voice instead.

2.5.2.1.Diagram

2.5.2.2.Wave

2.5.2.2.1.1.SourceFileName

2.5.2.2.1.2.FileFormat

2.5.2.2.1.3.FileSize

2.5.2.3.Looping

2.5.2.3.1.Off/On

2.5.2.4.Size

This value is calculated by the system and stored with the patch.
The difference between Drums and Audio is that audio always goes out to the separate LR audio

2.5.3. 2 = "Wavetable" Synth

2.5.3.1.Diagram

2.5.3.2.Oscillator

2.5.3.2.1.wavetable

the requisite and sometimes abundantly-sized waves themselves.

2.5.3.2.2.looping on / off

2.5.3.3.AEG

2.5.3.4.LFO

2.5.3.5.Size

This value is calculated by the system and stored with the patch.

2.5.4. 3 = Single-Oscillator Subtractive Voice

2.5.4.1.Diagram

2.5.4.2.Oscillator

2.5.4.2.1.wavetable

the requisite and sometimes abundantly-sized waves themselves.

2.5.4.2.1.1.SourceFileName

2.5.4.2.1.2.FileFormat

2.5.4.2.1.3.FileSize

2.5.4.2.2.looping on / off

2.5.4.3.AEG

Of greater concern we believe to be the basic expressiveness of the synthesizer voices themselves. The playability and therefore basic character of a synthetic sound is largely determined by its envelope controls, especially including the effects of velocity and its translation into a certain "feel" for the way the instrument wants to be played. Envelope programming is crucial for developing the suitability and expressiveness of a sound particularly in context with other sounds. As instruments diverge in their envelope programming, they relax their ability to render MIDI data to the same musical end. For the natural, imitative instruments especially, consistent envelope response between GM instruments is of course a key requirement for GM's success. Fortunately, time (e.g. milliseconds) is one of the more objective and simpler parameters to agree on. For simplicity, segments should be assumed to be linear, with shape being refined through the addition of intermediate points.

envelope and keyboard depths have to be graphed

number of points

which point for hold

values of each segment

2.5.4.4.Filter

module 2 parameters e.g. filter
filter type, filter q, freq., etc.

filter settings;

2.5.4.5.FEG

2.5.4.6.LFO

2.5.4.7.Size

This value is calculated by the system and stored with the patch.

2.5.5. 4 = Dual-Oscialltor Subtractive Voice

2.5.5.1.Diagram

2.5.5.2.Oscillator 1

2.5.5.3.Oscillator 2

2.5.5.4.Mixer

2.5.5.4.1.Osc 1

2.5.5.4.2.Osc 2

2.5.5.4.3.Noise

2.5.5.5.Filter

2.5.5.6.AEG

2.5.5.7.FEG

2.5.5.8.LFO

2.5.5.9.Size

This value is calculated by the system and stored with the patch.

2.5.6. 5 = Vector Synthesizer Voice

2.5.6.1.Diagram

2.5.6.2....(as above)

envelope looping on/off
number of loops

2.5.6.3.Size

This value is calculated by the system and stored with the patch.

2.5.7. ...more preset voice types

2.5.8. 100 = Network 1 Distortion Guitar

2.5.8.1.Diagram

2.5.8.2.Impulse Wave

2.5.8.3.Modules

2.5.8.3.1.Parameters

2.5.8.4.Size

This value is calculated by the system and stored with the patch.

2.5.9. 101 = Network 2 Acoustic Guitar

2.5.10.102 = Network 3 Flute

2.5.11.103 = Network 4 Clarinet

2.5.12....more preset voice types

2.5.13.400 = DX-7 #1

2.5.14.Patch 500 ... 1000 Custom Voices

2.6. Audio Editors > Audio

The next step is to drop in short audio clips as needed in the sequence. These may be drums, hits, sword clashes, or any samples in general.

The question of length arises: how long are these samples. But we don't want the composer to care. The question of whether we are playing from disk or not should not arise to the content developer.

By default, audio receives priority over synthesis, so what will be heard will depend on how much memory the application (or user) has allocated to the Synthesis Subsystem.

This step contributes audio samples in a variety of formats that are either mapped to a percussion map or played melodically (how every sound gets its own channel).

To key the audio the app must make up Combinations. Drum kit is a Combination.

2.7. Sequencers > Sequences

On the output side the Sequencer plays music and control data to the synth subsystem, for most intents acting just like a Player, which is explained below. And, like a Player, it may serve as a synchronization master or slave.

However, on the input side the difference is that a Sequencer records, displays MIDI data graphically or as Common Music Notation (CMN) for editing. And it enforces some rules on data exported to the Work Manager.

The sequence component of a Work looks very much like a Type 2 Standard MIDIFile.

2.7.1. Music data

Communicated in real-Time. Work Manager does not see real-time stuff.

2.7.1.1.Note Ons, Offs

2.7.1.2.real-time Controllers

2.7.1.2.1.Type 2 SMF

The point is that loops are retained.

2.7.2. Polyphony Enforcement

If the composer calls for so many voices that voices must be stolen to play the passage, then the passage needs to be fixed to use its voices more efficiently. Thus, the sequencer informs the composer when the sequence exceeds a set number of parallel voices, and shows them where the peak is, for example, by way of a histogram.

Unfortunately the actual number of sustaining voices can be greater than the number of simultaneous Note Ons, depending on synthesizer programming such as release times. These are cumbersome if not impossible to analyze dynamically. Therefore the synthesizer will have to occasionally impose its own voice stealing algorithms. But limiting data at the source is certainly the place to start.

If no priority table is defined, cut off voices in the highest numbered channel first, on down.

Alternately. it is easy to conceive of a sequencer feature that would intelligently perform the required voice trimming based on well-known voicing principles. This is a good example of how competition can remain.

The minimum available polyphony is 32 musical voices + 2 (L/R) audio lines.

2.7.3. Data Monitoring

The sequencer can always display the chased state of all Channel messages. This requirement helps satisfy those who complain that MIDI is not a searchable stream.

2.8. Sequencers > Priority Tables

For each bar in which your sequencer tells you that you have exceeded 32 synth + 2 audio voices, assign a priority map which tells the synthesizer which channels to honor before others. This will allow you to create sequences with as many channels as you like, that are scaleable to small or large playback systems.

As mentioned above, for every bar where the polyphony demand exceeds 32 voices according to a simple parallel view, the composer can declare a channel priority table. Reading this table, the Player will decide which of the lowest priority channels to ignore, based on the available polyphony.

For example, your basic string part is in channel 18 with priority 5. Channel 60 has a copied and doubled version of the string part which takes 10 voices. Give the enhancement track a priority high enough to be heard on a 64-voice machine, but not so high as to steal on a 32. Percussion parts can be spread throughout the priority list so that the scaling doesn't just work with melodic parts.

This step contributes optional channel priority tables to the Work. Without them, the player simply defaults to the track order.

2.8.1. Table of Channel Priorities

for examp[e

L + R Waves

Piano

trap set 6

ethnic perc 2

store and name tables

assign to measure numbers

Simple: composer assigns priorities to sounds. any that don't fit into RAM, falls back to the GM1 equivalent.

Voices Exhausted

- In sequnece: cut back or let go with priority table
- in synth, last note priority

Ram exhausted

Boundary cases

2.9. Sequencers > Automation Tables

In this final step any of the synthesizer, audio, effects, or mix parameters can be assigned to sequence channels for time-based dynamic control.

This step contributes a table of parameter channel assignments and associated scalars to the Work.

The automation table assigns any Work parameter to a sequence channel.

2.10. Sequencers > Object Commands

These commands allow the composer to sieze control over the loading process, loading individual sounds or at a higher level, over a specified range of bars.

2.10.1.LOADING

Loading proceeds until memory limit is reached. If memory fills, the load command may not complete.

2.10.1.1.LoadSounds

loads all sounds used in the song

2.10.1.2.LoadSound "S"

loads a specific sound by name or number

2.10.1.3.LoadSound "S1" "S2"

loads a range of sounds by number

2.10.1.4.LoadBar "N"

loads sounds used up to bar N

2.10.1.5.LoadBar "N1" "N2"

loads sounds used between bars N1 and N2

2.10.2.UNLOADING

Reciprocally, unloading allows the composer to forcibly purge.

2.10.2.1.UnLoadSounds

unloads all sounds used in the song

2.10.2.2.UnLoadSound "S"

unloads a specific sound by name or number

2.10.2.3.UnLoadSound "S1" "S2"

unloads a range of sounds by number

2.10.2.4.UnLoadBar "N"

unloads sounds used up to bar N

2.10.2.5.UnLoadBar "N1" "N2"

unloads sounds used between bars N1 and N2

2.11. Plug-Ins > MIDI Processors

Some music can depend significantly on real-time processing of the MIDI data. Examples of this process include mixing by scaling velocity, transposition, and panning. It is easy to envision inputs from a game into a processing module.

2.11.1.MIDI Processing

Certain effects are only attainable through the real-time processing of MIDI data, for example, as afforded by the Max program. Any Max or similar type Patch in place should be exportable to the Work. Other examples of MIDI processors

2.12. Digital Video Editor > MPEG File

2.12.1. Musn't forget the piece of AVI which the DM2 Player must handle in addition to everything in DM1.

2.12.2. WE're just using the MPEG buzzword because it is hot now and this title shows where it goes. Other video formats should be supported of course.

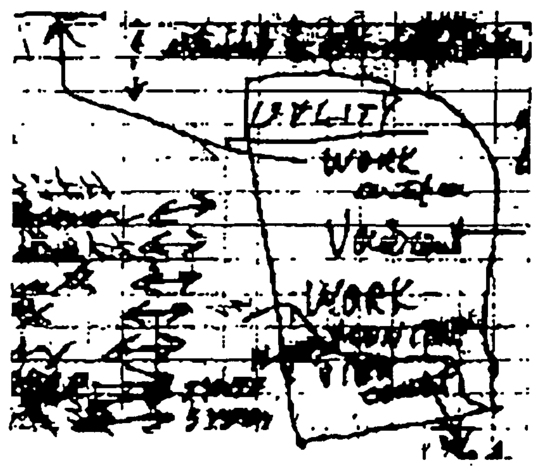
2.13. Work Manager > Global Data

2.13.1.a bunch of overhead fields related to orchestating all this tustff within the system

2.13.2.see next chapter

3. THE WORK MANAGER

Figure4 Work Manager



The work manager acts as a filter for data sent to the work by the tools
For simplicity the diagram shows paths from the tools to the Work but not through the Work to the Synthesizer. Yet it is obvious the Synthesizer subsystem must respond simultaneously in real-time to all changes produced by the editing tools, otherwise you wouldn't hear the results of your edit.

Therefore, the Work Manager re-transmits these commands to the synthesizer subsystem. (To avoid delays, Real-time Note data does not pass through the Work Mgr.)

Provides a global data is edited by the Work Manager itself.

also serves as a progamming level of indirection

3.1. DM Protocol Version Number

3.2. Work Name

3.3. Composer

3.4. Copyright Notice

3.5. Serial Number (optional)

3.6. Protection Enable/Disable

3.7. TotalRAMNeeded

Size of RAM to hold all Instruments, and sufficient pre-roll time for audio if all the sounds in the the Work are loaded at once. This figure allows the Player to quickly determine whether a Load All Instruments can be performed.

The Manager updates this field constantly

3.8. MinAudioBuffer

This is the

3.9. MinSoundBuffer

3.10. GM1 Supported

3.10.1.No/

3.10.1.1.17- go out to

3.10.2.Yes

3.10.2.1.1-16 go to normal GM1

If on, the Sequencer enforces the saving of complete data on tracks 1-16. Otherwise, track 1-16 data is excluded. The Player will post an error message if no suitable synth is downwind -

3.11. GM1 Preset Modes

Channels 1-16 are reserved for GM1 compatibility and cable-based systems. These are only employed if the composer actually desires that the Work support GM1 playback. It doesn't have to.

Assuming it does, the composer can select the type of file they want to save (from the list below), and the Sequencer will initialize as much correct data as it can, as well as enforce data integrity on saving. If on, the Sequencer enforces the saving of complete data on tracks 1-16.

Mode	Chans	Voices
Base	13-16	5 melodic+6 percussion (verify?)
Extended	1-10	10 melodic+12 percussion (verify?)
Base + Extended		
GM	1-16	24
GS	1-16	24 voices + effects
None	1-16 are disabled	

We prefer they didn't use some of the wimpier modes, but it's their decision what type of GM1 file to produce, so the standard tool should support that. It's best to choose GS mode, because only that contains the correct initializations for Sound Canvas.

3.12. GM1 Channel Initializations

Communicated when allowed by the operating system.

3.12.1.Name

3.12.2.Comment

3.12.3.Melodic or Druml

the composer be allowed to declare any number of drum channels

Melodic Middle C = C4

Drum Map = GM1, Korg, Ymam, Custom

Custome drum map table

3.12.4.Sound Select

One of the many features added by this system is the ability to for the synthesizer to do its best with downloadable sounds, possibly including waves files of arbitray length.

If the system is to arrrange for these sounds to be in RAM when called for, then it must be given fair warning. You can't just send a program change at any time then expect any sound to instantly be in place.

Therefore, Program Changes are now taken out of the sequence data stream and put in a separate category of sequencer setup data.

If GM1 support is on, for each Channel 1-16

GM1 sound set preset 1-128

For each Channel 17-1024

More channels are needed because we want one Program Change known in advance for each channel, and you will often want more than 16 sounds in a Work.

Program select 129 - ?

3.13. Extended Channels

Channel numbers 17 -1024 are reserved for music exploiting the new Work format. Each instrument and sound has its own channel. Channels are also used to control effects and mixers in real-time. (Note that in this system the terms track and channel are essentially equivalent..Track tends to be used in the sequencing context and channel tends to be used in the synthesizer context.)

3.14. Initialization Data

To solve the problem of setup data being on the down beat of the sequence, all data pertaining to the channel setup is removed from the note flow.

As soon as the player opens the Work, it begins to configure the synthesizer according to the default or custom channel priorities, subject to interruption by the operating system. Hopefully the content designer will have loaded the Work in enough time for the synth to configure itself by the time playing has to start. In any case, if started playing proceeds with GM1 default sounds until the real sounds start flying-in. Thus, startup is preserved in all cases.

3.15. Data I/O

3.16. Load Time Calculator

The composer or user can optimize the value for best response.

3.17. Viewer

3.18. Memory Managmenet

3.18.1.Compact Data

3.18.2.ZIP file

3.19. Work Editing

3.19.1.OF course the point is to be able to have several Works open and be able to Directly Cut/Copy/Paste any object bewteen Works

3.20. Work Certifier

3.21. Work Linking

there is a need to allow the composer to create and distribute in two different ways.
For example an Internet composer wants one file that includes all data.
But the game designer wants in addition to whatever custom sounds are required, to be able to refere to common sound banks as well. The use of custom sound banks helps keep Work file size small.
The distribtuion of common sound banks also makes for an interesting upgrade market. Consider mailing a CD each month to your Interner radio subscribers that contains the hottest sounds to be used on next month's songs.
Work Link is simply a list of banks for the Work to refer to for other sounds.
A sound bank is a Work that has no sequence data in it. They'll have to be stored in a predictable place, like System\Seer.

4. THE PLAYER

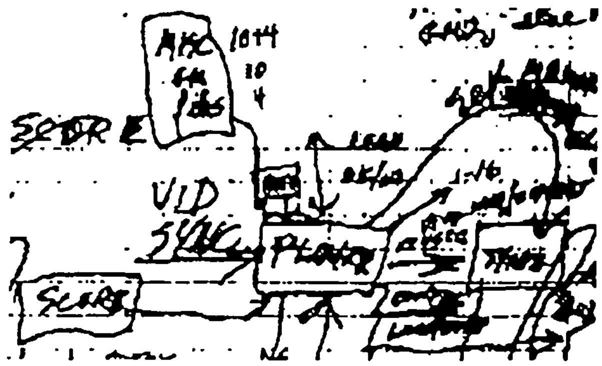
Sousa will allow users to play any of the six conflicting GM instrument-compatible MIDI file types without any other action; that is, without having to understand the MIDI Mapper.

er looks and mini-bank and queries the synth for capabilities

Seer has defined an enhancement application which complements *Satie* and capitalizes on its strategic strengths. Seer will develop and provide a new MIDI player, code-named *Sousa*, which takes advantage of a proprietary interface to the synthesizer, that will not be restricted by MIDI limitations.

This chapter discusseshow the player converts the Work into data whcih the synthesizer announces ...

Figure 4 Player



player opens work

er looks and mini-bank and queries the synth for capabilities

4.1. Basic responsibilities of the Player

When the player loads the sequence it learns what sounds are needed and confirms that the sounds are somewhere in the active banks (or posts a notice if some are missing).

The player examines the SMF to determine what kind of file it is and automatically selects the correct output channels to ensure that redundant (Base + Extended) data is not sent to the synth.

After configuring the channels the player asks the synthesizer about its capabilities, and tells the synthesizer what instruments are needed.

The player must automatically initialize the synth to defaults for any initial data not contained in the sequence, so that there is no undetermined behavior.

4.2. Player to OS

4.2.1. Status

4.2.1.1. WorkInPrep

4.2.1.1.1. Confirms receipt of PlayWorkNow.

4.2.1.1.2. Sent after Player opens Work, to access global data

4.2.1.2. WorkNowPlaying

4.2.1.2.1. reports that playback has indeed started

4.3. Player to Synth

4.3.1. Memory Requests

4.3.1.1.AllocateTotalRAM

Asks synth to set aside this variable amount from the Work Global data.

4.3.1.2.WhatAudioBuffer?

4.3.2. Object Loading

4.3.2.1.WhatObjects?

Returns list of current contents of sound RAM

4.3.2.1.1.Sequence

4.3.2.1.2.Patches

4.3.2.1.3.Instruments

4.3.2.1.4.etc..

4.3.2.2.MarkDiscardable "object"

Used to identify inessential sounds, which will linger until the OS reclaims their memory space.

4.3.2.3.LoadMix

Requests the Work's mix config of the Synth

4.3.2.4.LoadEffects

4.3.2.5.LOADING

Loading proceeds until memory limit is reached. If memory fills, the load command may not complete.

4.3.2.5.1.LoadSounds

loads all sounds used in the song

4.3.2.5.2.LoadSound "S"

loads a specific sound by name or number

4.3.2.5.3.LoadSound "S1" "S2"

loads a range of sounds by number

4.3.2.6.UNLOADING

Reciprocally, unloading allows the composer to forcibly mark discardable. Purging is not necessary, as discardable isntruments will be gobbled up as the OS needs their space. Instead of reloading the sound, the player (synth)might only need to mark it as InUse

4.3.2.6.1. UnLoadSounds

mark discardable all sounds used in the song

4.3.2.6.2. UnLoadSound "S"

mark discardable a specific sound by name or number

4.3.2.6.3. UnLoadSound "S1" "S2"

mark discardable a range of sounds by number

4.3.2.7. ConfirmLoad

if the object is loaded, returns true

4.3.3. misc

4.3.3.1. Communicating Setup Data

no instream program changes
or, instream changes are not gauranteed to have 0 latency
channels so that you don't have to put ins tream
in fact, allchanges sucked out to automatic updating prior to the first beat.
controllers out of main stream (all initializers)
player handle MIDI routing
Device Inquiry
If on, the Sequencer enforces the saving of complete data on tracks 1-16. Otherwise, track 1-16 data is excluded. The Player will post an error message if no suitable synth is downwind
Download Instrument requests instrument
voice type needs to write to a specifc accumulator
The Player looks ahead
When you hit stop the synth should return to defaults e.g. Volume = 127
When you hit pause the synth should stay with chase values e.g. Volume = 87.

4.3.3.1.1. Player Requests up to 1 Gb for Sound RAM

4.3.3.1.1.1.If value is OK, Synth responds ACK

4.3.3.1.1.2.If value is not OK, Synth responds Amount Offered.

4.3.3.1.1.2.1.If Player accepts value, sends Accept Value.

4.3.3.1.1.2.1.1.Synth responds ACK

4.3.3.1.1.2.2.If Player refuses value, sends Refuse.

4.3.3.1.1.2.2.1.Synth responds STOP

4.3.3.1.1.3.Player receives ACK, loading can begin.

4.3.3.2.Patch-Caching

find sequencers that actually do it
Patch caches to the extent components not loaded
ram size sensing
You set a RAM space and a Sound Set.
If the set exceeds the space, you are notified to be in cache mode
always respond to cache messages
cache mode may mean you have to play the song once to ensure right sounds in place
or play a seq which plays only program changes, just prior to the song itself
[or try a different app (provide list of any known supporters of cache msgs) wait for Sousa]
SOUND MEMORY SIZE
Free RAM button
Bank Select button
Load current Bank
Seer will soon outfit *Satie* with patch-caching (known more generically as "downloadable sounds"), which basically allows the synthesizer to use only the memory needed for sounds for the current music, as opposed to having to retain unused sounds.
autosense memory

4.3.3.3.What the Player Knows

By looking at the size of the sounds required, the Player can build a histogram of the RAM required to play audio and custom sounds demanded. To the extent demand exceeds supply, the Player knows to instead use a GM1 ROM sound instead. Starting with the lowest priority channels and working upward, it discards any custom instrument selection until RAM capacity is not exceeded. Similarly
This is an intentional scalability allowing small memory subsystems to play audio with standard GM1 music, while more RAM gets you better synthesis and higher quality music.
The minimum RAM size is that required to preload enough starting snippets to allow real-time response.

4.3.4. Opening other SMFs

4.3.4.1.Base and Extended

As you may know, Microsoft advanced a system of GM in Windows 3.1 which calls for up to three different types of GM (Base, Extended, and GM) and four different file types (Base, Extended, Base+Extended, and GM) to play them. You can get a very clear demonstration of the problem by playing "CANYON.MID" shipped with Windows 3.1 to a GM device. The GM device has no way to know that it is being sent redundant data, as well as a drum part on channel 16.

There is ample concern and disappointment over this situation and the general chaos which exists with regards to channel deployment. There is some interest in preparing a new set of categories—if not logos—to help current users sort out the mess that exists with regards to the variety of different SMFs that are being published and the variety of ways that quasi-GM compliant devices can be mapped particularly within Windows. However, it is equally valid to predict that unleashing four or five similar logos with minor variations are not going to help the general public. Instead, it may be best to wait until we have a fully improved, intelligent system that consumers can really handle, before asking them to identify with it.

One would like to make a clean start with Win 95, which we understand does not support the Base and Extended classes. But it is not so simple. There is a sizable installed base of (sub-GM) Base and Extended files and boards out there. This means that the new system must intelligently examine file content to find out how it is supposed to be mapped, then do the right thing. It is not sufficient to rely on the MS MARKMIDI utility to do this job, since its use is not broadly supported in the installed GM base.

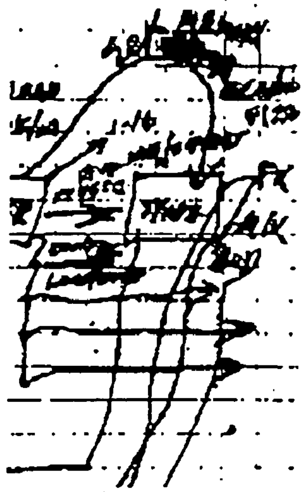
4.3.4.2.Conversion algorithm

player opens SMF drum (ditch MIDI Mapper)
decides type
if 1 - 10, ext
if 13-16, base
if all, if 50% of notes on 10 and 16 are same, then walk through 16 and look for hit on 10.
thon it io dual modc
else, play GM
Player then searches for each initial event except notes, and abstracts them to the setup register
then the player finding an in stream program change moves all following data (until the next change, and puts in on a new channel
all and then abstracts the intializations for the moved parts
abstracting the initializations allow the OS to precommunicate (cache) desired deatures to the synthesizer,

4.3.4.3.1-16 go to normal GM1

5. THE SYNTHESIZER SUBSYSTEM

This chapter describes how the synthesizer realizes a Work; in other words the structure of communication to read the information stored inside a Work file.



With this idea, rather than being unrealistically pressed to sound identical, synthesizers can compete on the size of their RAM above a certain minimum, their ability to support as wide a range of sounds and effects as possible,. fidelity, and the intelligence of their algorithms in dealing with the given cost constraints.

5.1. Synth to OS

5.1.1. Memory Requests

5.1.1.1.NeedTotalRAM

Requests complete RAM space for the Work from the OS, referring to the TotalRAMNeeded variable.

5.2. Synth to Player

5.2.1. Memory Commands

5.2.1.1.TotalRAMIsAllocated

sent after the OS allocates the RAM

5.2.1.2.SoundRAMIs

5.2.1.2.1.Reports current allocation as limited by the OS

5.2.1.3.AudioRAMIs

5.2.1.3.1.Reports current allocation as limited by the OS

5.2.1.4.TotalUse

Lists all objects currently in memory

5.2.2. Object Loading

5.2.2.1.ObjectsAre

Answers WhatObjects? from the Player
The list of all objects in memory.

5.2.2.2.MixLoaded

the Work's mix type is supported

5.2.2.3.MixError

the Work's mix type is not supported

5.2.2.4.EffectsLoaded

5.2.2.5.EffectsError

5.2.2.6.InstrumentLoaded

reports that a requested instrument is in place

5.2.2.7.InstrumentError

5.3. GM1 ROM Set

All instruments support the GM Sound Set with GS extensions (processors). This allows the user to make smaller Works by using the default instruments which are known to reside locally.
No surprise: the standard synth definition is the SC-55.

Note: In a system such as Seer's the distinction between ROM and RAM and spurious. There is no idfference; it is all RAM. And all sounds 1-are therefore treated uniformly.

5.4. RAM

Ultimately, the biggest problems we have with GM1 going forward is the lack of sound variety. The ROM approach of selecting fixed sounds is a dead end. ROM gives us stiff, inflexible sound that is boring to listen to on an extended basis. It matters little if you have a 128 or 1024 sounds; synthesis dies thereby.

On one hand you can listen to some of the impressive, professionally-done sequences play back on a high-fidelity sampler and easily believe that you are hearing music right off of a CD. In the hands of some pros, the GM1 sounds and drums are certainly enough to produce compelling and convincing music. But if this is true then why isn't everyone listening to GM1 instead of the radio? Yo! Check it out: what you hear on all non-nostalgic radio is a constant carnival of new sounds and effects competing for your attention. Even if you listen to a classical piano broadcast you likely hear a new piano in a new room with each song. But with GM it's the same unrelenting Piano 1 day in and out. Bo-ring. Then, as a composer if one of GM1's few acoustic pianos does not speak to you, you have no recourse.

Summarizing some of the original backlash against GM1 from the pro synth community, you could say that many composers do not want to force their ideas into one of 128 synthesizer slots. Much of our technologically-based music begins in the sheer delight of sound itself. Today's professional synthesizers offer far more sound selection power than most musicians can assimilate. Thousands of interesting and articulate instruments are available. Sure, relative to the *Sound Blaster* the GM1 system is heaven-sent. But, relative to the world of professional synthesis, composing for GM1 is to many knowledgeable synthesists somewhat like being forced to eat Velveeta in Paris.

Aside from the practical impossibility of getting everyone to agree on the parameters of hundreds of synthetic instruments, there is artistic danger in attempting to define acceptable synthesis to death. That way of thinking ignores the constant need for freshness and artistic freedom which is the hallmark of real, enjoyable music and which is therefore required of any music delivery system hoping to be more than a blip on the long-term radar. Although it does contain some provision for real-time parametric control, GS doesn't solve this problem. It just gives you more mannequins to put in the window. Same for Yamaha's new XG, (which is why we can't get too upset about the supposed format split it may cause; because it is all retrograde thinking).

GM2 must not abdicate its responsibilities to convey custom synthesizer power to the people. But how can it do this? The good news is that when you think about what is really needed and what can be done easily even now with multimedia personal computers, you see it is not necessary to undertake huge bureaucratic ordeals on behalf of dated hardware concepts. In fact a far more elegant and permanent solution is close at hand.

To summarize the argument against ROM: without a spec, the composers still aren't getting predictable results; and without variety, consumers do not yet seem to find very much satisfaction in GM music.

The urge to provide variety must be valid, because alternate sound bank layouts are the chief differentiating features of the GM "enhancements" GS and XG. But in contrast to what seems to be the Japanese approach,. among many in the U.S. synthesizer business there is remarkable agreement that a better answer lies down the completely different path of expecting that there will be custom sounds to be downloaded into the synthesizer's RAM. The problem with GM1 is not that it is too variable. The problem is that it is not variable enough. Instead of fearing embarrassment, GM2 needs to embrace and make a home for variation. This move towards downloadable custom instruments will provide the desired capacity for both variety and predictability to serve composers as well as game and content sound designers, incidentally giving GM2 the longevity to deserve being renamed GM2000.

Now we are inside the computer. The parallel PCI bus to which most computer companies are moving operates in the range of 190 MHz. To take advantage of this astounding power we must throw off our habitual limitations of thinking about what MIDI can do, starting with the abandonment of features limited by a 31.25 kHz cable. In general, to get the integrated and painless solution we all say we want, then we must demand far more of sequence players and synthesizers, as well as of sound editors and sequencers themselves. We require much more of the interface in terms of intelligent communication among devices as to their needs and capabilities.

Several companies have entered the area of dynamic allocation of hard or soft synth processing resources and already offer downloading or customization in the PC game or Windows market. Intel/Seer Systems is one. Creative/Emu has SoundFonts supported by its Vienna Editor for the AWE32. Turtle Beach (ICS) has .wav-based SampleStore. Ensoniq is also active in this area, including with the idea of downloadable effects. We seem to still be at the stage of competing proprietary systems. No one yet seems interested in really sitting down and standardizing the approaches. Nevertheless a superset of features must emerge which completes the thought about a complete MIDI music delivery system for the consumer.

In truth, when we start talking about 256 voices, 256 channels, unlimited sounds, bus speeds of 190 Mhz, virtual wave instruments, it is not clear that we are talking about MIDI any more. So let's just say that some universal MIDI-like system along the lines described below is expected to emerge. Who does it and what it is called matters less, but we are betting it will look functionally similar to the following scenario.

5.5. MIDI Processors

[Ooops, what if we want to have MIDI data processors—seems they have to implemented in the synth, so that's OK.

5.6. Voice Stealing

only steals when forcedto by long envelopes
last note priority FOR CONSISTENCY

Differences in dynamic allocation strategies cannot be ignored nor tolerated. Of all the systems around: 24 free; 16 instrument / 8 drum, last-note priority, drum channel priority (as in the Canvases), only one can prevail. As many voices as can be agreed upon (be it 24, 32, or 256) combined with last note priority offers the most consistent response and lets the sequence composer decide where the resources go. If there are abundant voices available, allocation nuances matter less, and last note priority will always serve fine.

caches voice to the the extent the voice is not loaded

mode 4 - retriggering goes into the composers per channel

the synth is not the right place for musical decisions to be made.

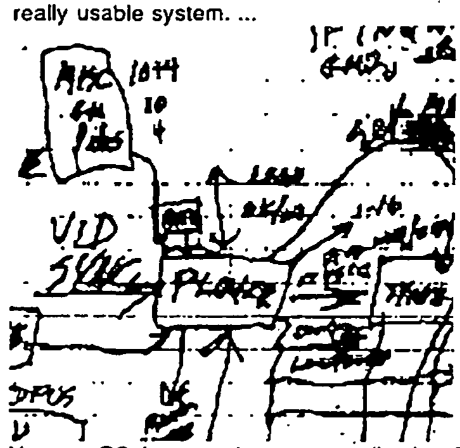
5.6.1. GM1 Synthezier

6. OS INTERFACE

The OS can help us immensely by telling us when it is cool to load sounds from disk. In this context cool means the MCI supports a game or app telling the synth to load because it knows the time is right.

What we have described so far is the system free running. In reality we want the OS, or really some apps under the OS, to have control over the Player's and Synth's activities. Either to allow the refreshing of RAM

This chapter talks about the influence of the OS on the player to complete the picture and give us a really usable system. ...



You say OS, but you also mean application. The OS is the intermediary. Obviously, it won't work for the player to simply send the instruments on the downbeat of the sequence. Instead the system must communicate the instrument definitions and data gracefully within the real-world multimedia context. For example, we want to say that the synthesizer must be able to play and download sounds simultaneously. But when talking about system services (such as the hard drive in this case) some things (like the effect of disk access on pending processes) leave our direct control. Basically, you wnat the app to be able to control sound loading from disk, because only the app knows when the best time to do that is.
the Work's mix type is supported

6.1. OS to Player

6.1.1. Playback Control

6.1.1.1. PlayWorkNow "Path:\WorkName"

Demands immediate playback of the Work from the Player.

6.1.1.2. RenderWorkNow "Path:\WorkName"

Writes to a wave file.
Choosen by an application whenever the Work is too large to realize in real-time

6.1.1.3. ForceStartWork

This command tells the Player to not wait any longer.

6.1.1.4. PauseWork

6.1.1.5. StopWork

6.1.1.6. LastBar?

Asks for the last bar that has been successfully loaded.

6.1.2. Synchronization

6.1.2.1. UseSMPTEOn

6.1.2.2. UseSMPTEOff

Slaves the Player to system-provided time code.

6.1.2.3. Cpu usage

6.1.2.4. reverb switching

6.1.2.5. Speed Detected

6.1.2.6. Type Detected

6.1.2.7. Standalone Control Panel

6.2. OS to Synth

All of htese functions deal with memory managmeent.

6.2.1. Memory Commands

6.2.1.1.TotalRAMAllocated

Request for TotalRAM approved and done.

6.2.1.2.AudioBufferSet

Reports current setting.

6.2.1.3.WaitLoading

Delays loading the next instrument in priority.
The content developer can always issue these from the title while something more critical is going on.

6.2.1.4.ContinueLoading

Resumes loading the next instrument in priority.

6.2.1.5.Deallocate

This commands causes the synthesizer to discard the next higher priority.

6.2.1.6.Purge

Frees all sound memory. Leaves drivers intact

7. USE SCENARIOS

A popular song or game accompaniment is not usually made by a strict procedure. The steps rarely occur in any order—there is considerable back-and-forth. For example, although the final mix is left until near-last, an initial mix must be made if you are to hear anything at all. And the whole recording process often involves muting and unmuting instruments (gross mixing) so that parts can be isolated, identified, and corrected as needed. Thus several scenarios should be explored which exercise the system in a variety of ways.

7.1. Start-Up Sound

This is the story of how you work with the new system inside your computer. Starting with power-off.

This is written from the perspective of Seer's software synth. The procedure for others is a little different.

A startup Work is the minimum case for the synthesizer, so let's examine what is needed to make it happen.

7.1.1. Power-On

7.1.1.1.The synthesizer drivers install in the OS.

7.1.1.2.No default sounds.

7.1.2. Boot-Up

The user has used the Sound Control Panel to assign the Work Startup.wrk to the Startup event. The OS wants to play the Work as soon as it has arrived. Since this is the first application to load, we there are no current RAM limitations (i.e. this is a 16M machine).

7.1.2.1.Sound applet tells OS to tell Player to Play the Work

7.1.2.1.1.Work Description

7.1.2.1.1.1.This Work consists of a two-bar jazz quartet where

7.1.2.1.1.1.1.the sax joins on bar 2 beat 1 and

7.1.2.1.1.1.2.with your own voice singing "Yeah!" on bar 2 beat 4.

7.1.2.1.1.2.The sequence has five tracks:

7.1.2.1.1.2.1.1 bass

7.1.2.1.1.2.2.2 kbd

7.1.2.1.1.2.3.3 sax

7.1.2.1.1.2.4.10 drums

7.1.2.1.1.2.5.17 audio

7.1.2.1.1.3.And uses basic GS effects.

7.1.2.1.1.4.The first four tracks are conventional GM1 instrumentation.

7.1.2.1.1.5.Number 17 is the first that makes use of DisMus custom features.

7.1.2.2.Player opens the Work.

7.1.2.2.1.Opening reads the basic Work requirements

7.1.2.2.1.1.TotalRAMNeeded

7.1.2.2.1.1.1.It finds a value of 200K.

7.1.2.2.2.Asks synth how much RAM using.

7.1.2.3. Player asks the Synthesizer for 200K

7.1.2.3.1. and receives an ACK back

7.1.2.4. It loads the mix first

7.1.2.4.1. Select topology 1

7.1.2.4.2. params

7.1.2.5. Load the effects next.

7.1.2.6. The player sees that bass, kbd, and drums, all play in bar 1

7.1.2.7. Player tells synth to load two melodic instruments

7.1.2.7.1. instrument 33 bass

7.1.2.7.1.1. patch 1 low bass

7.1.2.7.1.2. patch 2 high bass

7.1.2.7.2. instrument 5 kbd

7.1.2.7.2.1. patch 3 kbd octave 2

7.1.2.7.2.2. patch 3 kbd octave 3

7.1.2.7.2.3. patch 3 kbd octave 4

7.1.2.7.2.4. patch 3 kbd octave 5

7.1.2.8. The Player then sees that only four drum sounds are needed in the first bar:

7.1.2.8.1. instrument 300 tiny drum kit

7.1.2.8.1.1.patch 12 kick

7.1.2.8.1.2.patch 15 snare

7.1.2.8.1.3.patch 38 hi hat closed

7.1.2.8.1.4.patch 44 ride cymbal

7.1.2.9.With all sounds for bar 1 in place, the player now must decide whether to start or not based on how much time it will take to load the saxophone and the audio clip.

7.1.2.9.1.Request load time for instrument 62 saxophone

7.1.2.9.2.result is 250 Ms

7.1.2.9.3.Request load time for instrument 156 audio clip

7.1.2.9.4.result is 1.25 seconds (for example)

7.1.2.10.Since the Player knows (at Tempo = 120) that it has about 2 seconds until bar 2 beat 1, it starts playing.

7.1.2.11.The Player immediately starts loading the saxophone. Worst case, sax loading is complete by bar 1 beat 2.

7.1.2.12.No later than bar 1 beat 3, loading of the audio clip starts.

7.1.2.12.1.If audio can't load in time, what happens?

7.1.2.12.2.Well, it will miss it's Note On unless that is somehow saved to key it albeit late.

7.1.2.13.Everything Plays.

7.1.2.14.After all envelopes clear, synth volumes automatically set to default 255.

7.1.2.15.Conventional GM1 instruments are retained by default

7.1.2.15.1.Instruments 1-128 are retained

7.1.2.15.2.Drums 1-47 are retained

7.1.2.16.Other stuff is marked purgable since it likely won't be used next sequence.

7.1.3. Power-Off

7.1.3.1.save RAM preferences

7.2. Big MIDI in Small RAM

7.2.1. A few GM1 sounds are already installed

7.2.2. Several other applications are open, like Director and a Sequencer

7.2.3. User opens large Work into Player

7.2.3.1.Work Description

7.2.3.1.1.This Work consists of 30-track techno-rap tune

7.2.3.1.2.Lots of instrumentation

7.2.3.1.3.Lots of samples

7.2.3.1.4.Polyphony ranges to 50 voices at times

7.2.3.1.5.Uses GS Hall + Chorus + Delay + Distortion.

7.2.3.1.6.Real-time control of some effects depths.

(The following chain of events might be interrupted by the user clickingon PLAY at any moment. We may have to set a minimum point at which user triggering is considered.)

7.2.4. Learn Needed Objects and Load Them

7.2.4.1.Player requests from synth the list of its current objects

7.2.4.2.Player compares to list of objects for current Work.

7.2.4.3.Player tells synth to mark all non-current objects discardable.

7.2.4.4.Player tells synth to load mix and effects

7.2.4.5.Then load non-duplicated patches of instruments in priority.

7.2.5. Boot-Up <<<start here >>>

7.2.5.1.Opening reads the basic Work requirements

7.2.5.2.Player asks the Synthesizer for 200K

7.2.5.2.1.and receives an ACK back

7.2.5.3. It loads the mix first

7.2.5.3.1. Select topology 1

7.2.5.3.2. params

7.2.5.4. Load the effects next

7.2.5.5. The player sees that bass, kbd, and drums, all play in bar 1

7.2.5.6. Player tells synth to load two melodic instruments

7.2.5.6.1. instrument 33 bass

7.2.5.6.1.1. patch 1 low bass

7.2.5.6.1.2. patch 2 high bass

7.2.5.6.2. instrument 5 kbd

7.2.5.6.2.1. patch 3 kbd octave 2

7.2.5.6.2.2. patch 3 kbd octave 3

7.2.5.6.2.3. patch 3 kbd octave 4

7.2.5.6.2.4. patch 3 kbd octave 5

7.2.5.7. The Player then sees that only four drum sounds are needed in the first bar:

7.2.5.7.1. instrument 300 tiny drum kit

7.2.5.7.1.1.patch 12 kick

7.2.5.7.1.2.patch 15 snare

7.2.5.7.1.3.patch 38 hi hat closed

7.2.5.7.1.4.patch 44 ride cymbal

7.2.5.8.With all sounds for bar 1 in place, the player now must decide whether to start or not based on how much time it will take to load the saxophone and the audio clip.

7.2.5.8.1.Request load time for instrument 62 saxophone

7.2.5.8.2.result is 250 Ms

7.2.5.8.3.Request load time for instrument 156 audio clip

7.2.5.8.4.result is 1.25 seconds (for example)

7.2.5.9.Since the Player knows (at Tempo = 120) that it has about 2 seconds until bar 2 beat 1, it starts playing.

7.2.5.10.The Player immediately starts loading the saxophone. Worst case, sax loading is complete by bar 1 beat 2.

7.2.5.11.No later than bar 1 beat 3, loading of the audio clip starts.

7.2.5.11.1.If audio can't load in time, what happens?

7.2.5.11.2.Well, it will miss it's Note On unless that is somehow saved to key it albeit late.

7.2.5.12.Everything Plays.

7.2.5.13.After all envelopes clear, synth volumes automatically set to default 255.

7.2.5.14.Conventional GM1 instruments are retained by default

7.2.5.14.1.Instruments 1-128 are retained

7.2.5.14.2.Drums 1-47 are retained

7.2.5.15.Other stuff is marked purgable since it likely won't be used next sequence.

7.2.5.16.Power-Off

7.2.5.16.1.save RAM preferences

7.3. Composing

These are professional and semi-pro sequencing and sound products used to create the Work. The composer exports desired data structures into the Work. And reciprocally, the Synthesizer subsystem extracts these data structures to recreate the Work.

In many cases, existing successful file types such as .mid and .wav are appropriated so that existing tools don't have to modify themselves more than necessary to support DisMus.

7.4. Hack the Tune

Start off with a basic synthesizer, likely GM1 instruments in a standard effects mix such as on a Sound Canvas. We could call this default mode. Use standard sequencer features to basically outline the sections of a tune and arrange them in song format.

This step contributes the Type 2 MIDI File and GM1 setup defaults to the Work.

7.5. Custom Synth Sound Data

The composer or sound designer will use wave and instrument editors which produce banks of custom sounds and instruments needed for the sequence. These sounds can range from raw waves in any format, through wavetable synthesis with looping oscillators (as is now customary), to subtractive synthesis, FM, ultimately using any synthesis technique, including physical modeling.

The basic question which governs this approach is how finely to divide up a given module. For example, are modulation controls best grouped with their sources, destinations, or left as self-contained attenuators? Similarly, should this protocol offer a variety of algorithm choices, or allow you to freely define a voice as an arbitrary collection of modules?

The answer is that it should do both. We need to balance short term solutions with open-ended growth. Everything should have an operating fallback and everything should be fully scaleable outward to the future. We can easily start off with known and accepted algorithms (such as the basic ones in the K2000, just for example), and provide escape codes to areas of broad custom definitions.

The GM1 Sound set (and drum map) are about as much as we will get in terms of a ROM map. MMA not able to agree on using GS. And XG is incompatible.

8. APPENDIX

8.1. Overview of Appendix

8.2. Credits

Stanley Jungleib is Chairman of the MMA's General MIDI Level 2 Working Group. He is President / CEO of Seer Systems, Inc. inventors of the first commercial real-time GM software synthesizer. Portions of the above are excerpted from his book *General MIDI,* recently published by A-R Editions.

8.3. References

, DEFILIPPO & FERRELL LLP
ATTORNEYS AT LAW

3/18/97 — work links many reference samples, sounds... located anywhere & retrieved for performance by the client.

February 19, 1997

94028

U.S. Patent Application
System and Method for Generating, Distributing, Storing and Performing Musical Work Files
Ref.: 612

ɔlease find a first draft of the patent application for your invention . and Method for Generating, Distributing, Storing and Performing ʻiles."

ɜview the application for accuracy and completeness, and inform me ꞩ you might have. Feel free to make any necessary changes directly

ɔmpletion of your review, I will prepare the formal papers for your forward the application to the patent office for filing.

have any comments or questions, please do not hesitate to contact me.

Sincerely,

Marc Sockol

Marc Sockol

Enclosure

EXHIBIT "D"

CARR, DEFILIPPO & FERRELL LLP
ATTORNEYS AT LAW

WRITER'S DIRECT DIAL NUMBER
(415) 812-3407

April 9, 1997

VIA FEDERAL EXPRESS
415-851-7995

Stanley Jungleib
Seer Systems, Inc.
33-Tintern Lane #5
Portola Valley, CA 94028

Re: New U.S. Patent Application
Title: System and Method for Generating, Distributing, Storing and Performing Musical Work Files
Our Ref.: 612

Dear Stanley:

Enclosed please find the revised draft of the patent application for your invention entitled "System and Method for Generating, Distributing, Storing and Performing Musical Work Files."

Kindly review the revised draft for accuracy and completeness, and inform me of any revisions you might have. If you have no more changes, I will prepare the formal papers for your signature and forward the application to the patent office for filing.

If you have any comments or questions, please do not hesitate to contact me.

Sincerely,

Marc A. Sockol

Marc Sockol

Enclosure

EXHIBIT "E"