s.c.r.e.a.m.

System for the Creation of Random Electronic Adaptive Music

The Orange Lunchbox Brigade
Ryan Curtin
Chad D. Kersey

ECE4884-L04: Prof. David V. Anderson

June 1, 2008

Contents

1 Introduction **2**

1.1 Background of Experimental and Random Music 2

1.2 Background of Electronic Music 3

1.3 Motivation 3

1.4 Objectives of s.c.r.e.a.m. 4

2 Project Description and Goals **4**

2.1 List of Design Goals for the s.c.r.e.a.m. Framework 4

2.2 List of Design Goals for the Simple Implementation of the s.c.r.e.a.m. 5

3 Technical Specifications **6**

3.1 Modularity 6

3.2 Simultaneous Playback and Recording 6

3.3 Output Dependence on Environmental Sensor Data 7

4 Design Approach and Details **7**

4.1 Design Approach for the s.c.r.e.a.m. Framework 7

4.2 Design Approach for the Simple s.c.r.e.a.m. Implementation 8

4.3 Interprocess Communication 9

4.4 Architecture and Functionality of the Brain 10

4.5 Coding and Standards 11

4.6 Alternatives and Tradeoffs 12

5 Schedule, Tasks, and Milestones **12**

6 Project Demonstration **14**

7 Marketing and Cost Analysis **15**

7.1 Possible Target Customers 15

7.2 Marketing Ideas 16
7.3 Cost Analysis 16

8 Summary **16**

Executive Summary

The **S**ystem for the **C**reation of **R**andom **E**lectronic **A**daptive **M**usic (s.c.r.e.a.m.) is a framework for the generation and synthesis of random music. Based on the experimental work of John Cage, Karlheinz Stockhausen, and others who questioned and helped reformulate the definition of music, the system allows for the generation of any possible genre of music.

The s.c.r.e.a.m. will interpret predefined probabilistic models and use them to influence the structure of the music it creates. These models can be easily configured, so that the type of music the system creates can be easily modified. The s.c.r.e.a.m. is made up of several smaller components; this modularity allows for further easy modification of the system's functionality. The framework allows for an arbitrary number of instruments to be used; the instruments themselves are open-ended and therefore any collection of sounds can be implemented as an instrument. This allows for almost endless possibilities. The system also allows for an arbitrary number of environmental sensors that can modify the music.

The framework of the s.c.r.e.a.m. will be written in mainly C and C++, conforming to C99 and ISO C++ standards. It will be structured as a set of extensible libraries, furthering modularity and allowing easy design modification.

This project seeks to create the s.c.r.e.a.m. and a simple implementation of it. This simple implementation will use the s.c.r.e.a.m. framework to create simple music of a single genre using a small number of instruments. It will have a few environmental sensors; one of these will be a microphone that can add sounds from the environment into the music. Once completed, the finished project will produce simple musical output that can be easily recognized as music.

1 Introduction

1.1 Background of Experimental and Random Music

Experimental, or avant-garde, music is a very loose term for music that lies at or outside the boundary of traditional music. Artists who have ventured into this category include John Cage, Karlheinz Stockhausen, Philip Glass, Pierre Boulez, and many others. John Cage, probably the most well-known experimental musician, experimented in "chance music" [1], which refers to music where different elements, such as a particular note, are determined purely by chance. Cage was also prolific for his other works of experimental music; the most famous of these was his piece titled 4'33", which was 273 seconds of silence. More important than the music developed by composers like Cage, Stockhausen, and others, though, were the ideas behind their music; Cage, specifically, expressed the opinion that music could be any collection of sounds, and commonly gave performances where random audience noise was meant to be the actual music. These ideas can be summed up simply in a couple quotations by Cage and Stockhausen:

> "The first question I ask myself when something doesn't seem to be beautiful is, 'why do I think it's not beautiful?' And very shortly you discover that there is no reason."
>
> John Cage

> "I became aware that all sounds can make meaningful language."
>
> Karlheinz Stockhausen

> "I no longer limit myself."
>
> Karlheinz Stockhausen

1.2 Background of Electronic Music

In the past fifty years, electronic music has become a mainstream concept. After the introduction of instruments like the electronic keyboard in the 1960s, musicians like Jean-Michel Jarre and Karlheinz Stockhausen began to experiment with electronically-created music [2, 3]. In the late 1960s and 1970s, synthesizers like the Moog and Minimoog provided a much cheaper way for musicians to incorporate electronic sounds in music, and they were quickly adopted and used by such artists and groups as The Monkees, The Beatles, Miles Davis, and Herbie Hancock, in addition to countless others. Now, in 2008, electronic instruments can be found in nearly every musical setting, from esoteric avant-garde studio productions to commercial jingles and advertisements to video game music.

1.3 Motivation

With the recent advent of powerful computing systems, Cage's ideas of "chance music" can be implemented on a much greater scale than flipping coins to determine melodies. Over the last twenty years, several musical systems implemented on computers incorporating a random element have been designed and put into action. The iMUSE system, created by LucasArts developers Michael Land and Peter McConnell in the early 1990s [4], was a music system for video games that synchronized the music with visual action. Dr. Ulf Berggren, of Uppsala Universitat, developed a system that used a random process to create random sonatas modeled after those of Mozart [5]. A few patents have been filed on music generation systems, but no commercial music generation systems exist [6, 7, 8].

Although exploratory work has been done in the field of random music generation, none of these systems provide a truly abstract system for creating any type of random music. Dr. Berggren's system generates sonatas; this is a small subset of a single genre of music. While this sonata-generation system is useful, a more useful system would be one that was not restricted to generating sonatas, but instead could generate any kind of music.

1.4 Objectives of s.c.r.e.a.m.

The **S**ystem for the **C**reation of **R**andom **E**lectronic **A**daptive **M**usic (s.c.r.e.a.m.) exists to fill this void. The project seeks to create a framework for a completely open-ended random music generation system. This system is meant to be able to generate any genre of music and have as few boundaries as possible limiting what is generated. The system is also meant to be adaptive; it should include the ability of the music to adapt to its environment using external sensors. The system is meant to be as modular as possible, so that each component of the system can be interchanged with another at will. Overall, s.c.r.e.a.m. is an open-ended tool aspiring to the ideals of experimental musicians like John Cage, Karlheinz Stockhausen, and others.

However, it should be noted that the s.c.r.e.a.m. is a framework, and unless it is implemented and configured properly, it will not produce anything that sounds like today's definition of music. Therefore, this project encompasses the construction of the s.c.r.e.a.m. framework, as well as a simple implementation to demonstrate its usefulness and abilities.

2 Project Description and Goals

The s.c.r.e.a.m., as its name implies, is a system for the automatic generation of random music that is affected by its environment. While the s.c.r.e.a.m. itself is merely a framework and not an implementation, this project seeks to design the framework as well as provide an implementation. A list of straightforward design goals for the s.c.r.e.a.m., given that the objectives are the construction of the framework and a simple implementation.

2.1 List of Design Goals for the s.c.r.e.a.m. Framework

The s.c.r.e.a.m. framework should:

1. Accept environmental input from sensors as input

2. Produce fully synthesized music as output
3. Randomly modify music over time; its output should 'evolve'
4. Use sensor input to modify its output
5. Parse configuration files for the probabilistic models that are used to generate music
6. Be able to produce output for any possible 'instrument'
7. Be as modular as possible, so each possible component can easily be rewritten and expanded upon
8. Be as configurable as possible, so as to allow for any possible type of music

2.2 List of Design Goals for the Simple Implementation of the s.c.r.e.a.m.

The simple implementation of the s.c.r.e.a.m. should:

1. Generate simplistic music of a single, well-established genre
2. Produce output recognizable as music
3. Use a small number of 'instruments', or sound collections, to synthesize the music
4. Be clearly affected by its environment and sensor input
5. Include a microphone that uses an adaptive filter (described below)

The aforementioned microphone functions as a microphone that captures samples of the external environment and plays them back in the music. It will need to use an adaptive filter to filter out its own output from the voice sensor's input; otherwise, the system will produce feedback and be unstable.

3 Technical Specifications

3.1 Modularity

The long-term goals of this project are not to create a particular kind of musical application, but to create a framework in which many different kinds of musical applications can be developed. To make this a reality, the components of the system should be as basic as possible and have well-defined interfaces between them. This is so that in the future it will be possible to combine already-built components in new ways and create new components to interface with these. Then, many different kinds of musical tools can be created with minimum effort.

This will be realized primarily by creating the project as a collection of small programs that talk to each other through various kinds of binary FIFOs, which could be pipes, files, or network sockets. Good software design practices must also be adhered to; documentation of every detail of the framework must be extensive and accurate. This modular approach with extensive documentation will help to ensure and extend the usefulness of the system.

3.2 Simultaneous Playback and Recording

The simple implementation of the s.c.r.e.a.m. proposed for this project includes a microphone. This sensor would record sound from the environment, and play it back through the system. However, problems arise when this input is in the same environment that the system's output is being played into. The system's output will produce an unstable feedback loop, which is certainly undesirable. The solution choosen to combat this problem is an adaptive filter, which will eliminate the system's output from the voice sensor's input and remove the unstable feedback effects.

3.3 Output Dependence on Environmental Sensor Data

To truly tailor the music to its surroundings, a simple microcontroller-based sensor module that communicates with the host computer via a serial port will be developed. The numerical textual data this creates will be processed by the system and influence the music being produced according to preconfigured probability models.

One environmental sensor has already been created and used by one of the project's engineers in the past [9]. This sensor is a humidity sensor, and will be retooled for use as a s.c.r.e.a.m. sensor.

4 Design Approach and Details

4.1 Design Approach for the s.c.r.e.a.m. Framework

The design of the s.c.r.e.a.m. framework, as discussed earlier, is intended to be as modular as possible. This design paradigm lets a developer replace one component of the s.c.r.e.a.m. with another, with minimal effort. Following this design idea, a top-level abstraction of the s.c.r.e.a.m. can be seen in Figure 1.

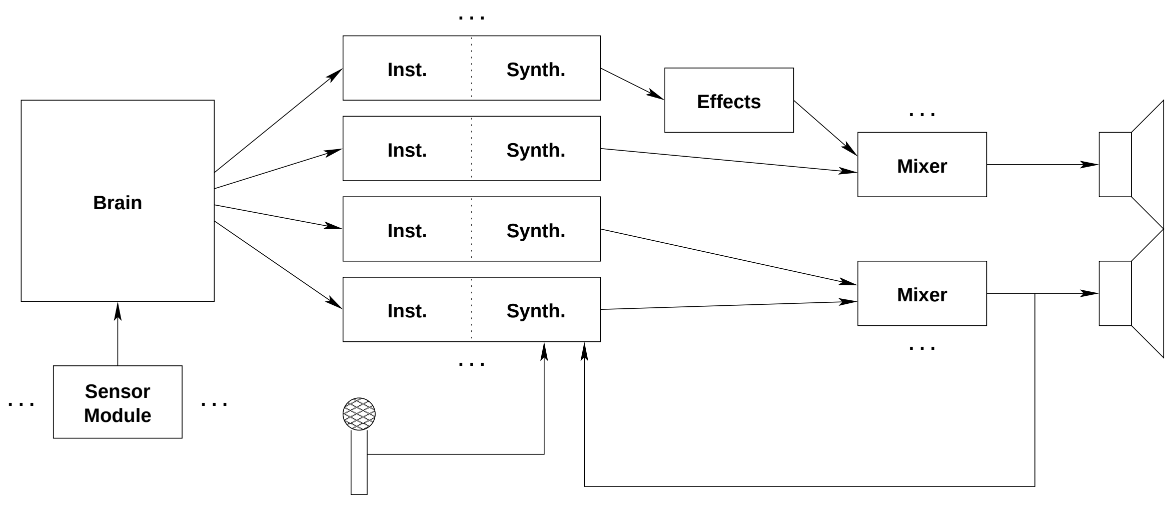

Figure 1: Top-level abstraction of s.c.r.e.a.m.

It can be seen in Figure 1 that the "brain" is the controlling unit of the entire system. It accepts input from any number of sensor modules, and the sensor inputs affect the output of the brain. The brain determines the structure of the music (components including current chord, current musical structure, form, tempo, and suggested rhythms) and sends this information to each instrument (denoted "Inst."). The number of instruments is arbitrary. Once each instrument uses the information given by the brain to determine what rhythms and tones it is going to play, it sends this information to its synthesizer (denoted "Synth.") and it is turned into PCM audio. This can then be sent to either an effects processing module (denoted "Effects") or directly to an output mixer. An effects processing module would modify the PCM audio stream in any way, and output the modified PCM audio to either another effects processing module or a mixer. A mixer's function is simple; it takes all the PCM audio inputs it is getting and turns them into one composite PCM output. Then, this output can either be saved to a file, played over speakers, or any number of other possibilities. Any number of mixers in a single system is possible. This has interesting implications; for example, one brain could be simultaneously controlling four isolated music performances.

The modularity of this system allows for nearly endless possibilities for the output of the system. With any number of possible instruments, completely different musical ensembles can be constructed; for instance, the system could be using the instruments of an entire orchestra, or, on the other hand, it could be using only an accordion and a kazoo. The instruments do not even have to be known instruments; they are just a collection of synthesizable sounds, so any conceivable set of sounds can be used as an instrument.

4.2 Design Approach for the Simple s.c.r.e.a.m. Implementation

The proposed simple implementation of the s.c.r.e.a.m. will follow the top-level abstraction found in Figure 1. The system will have only one mixer. It will accept PCM audio from a minimal set of instruments, being controlled by one brain configured to produce simple output that is recognizable as music.

The system will also have a voice sensor with an adaptive filter, used for simultaneous playback and recording, as discussed earlier. A simple design abstraction for the adaptive filter system to be used with the voice sensor can be seen in Figure 2.

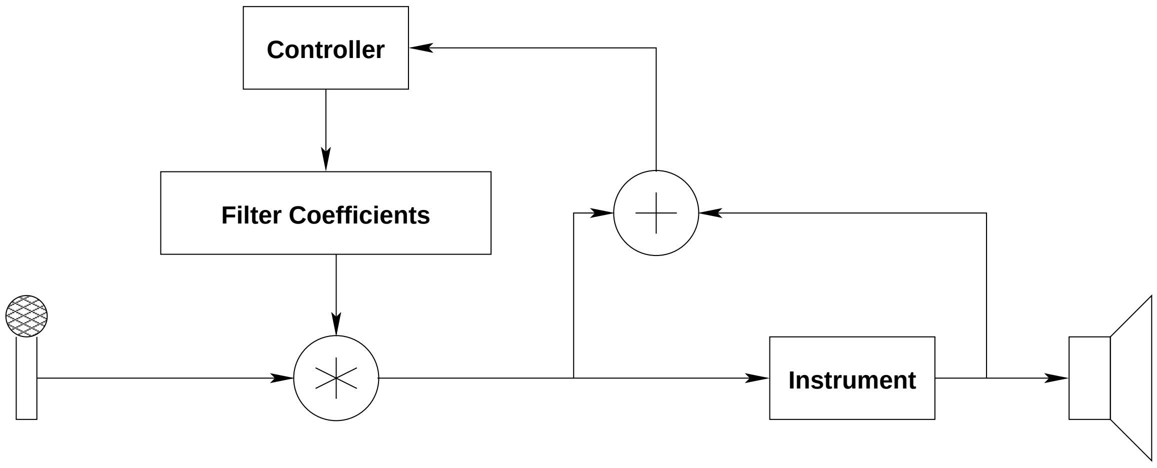

Figure 2: Schematic of adaptive filter system.

In this filter design, the output of the system is fed back into the filter. The filter coefficients are then determined such that when the filter is convoluted with the microphone's input, the music output is filtered out of the microphone's input.

4.3 Interprocess Communication

Because the modular approach chosen for s.c.r.e.a.m., multiple small programs are being created instead of one large multithreaded application. This approach prevents the rapid sharing of data structures in memory and creates the necessity for an interprocess communication system. Many methods have been devised over the years to handle different kinds of communication between processes on the same or different systems, in real time or asynchronously.

Since the system is meant to be as close to real time as permissible while still maintaining its modular and network transparent architecture, any method chosen for interprocess communication should make minimum latency a high priority. While shared memory has the least latency, it also is the least easily extended to network architectures and asynchronous operation. Named pipes allow a convenient method for local interprocess communication, and can be substituted with files when an asynchronous method is desired. Also, they are easily tunneled through network sockets, allowing network transparent operation.

A simple library, `scream_ipc`, has been designed to handle interprocess communication though named pipes, sockets, and files. By creating a data type, `scream_pipe_t`, that represents an individual bidirectional data link, and two functions that operate on this, named `pipe_send` and `pipe_receive`, it becomes possible to write backends that operate using sockets, named pipes, pre-generated files, or any other method that may be implemented in the future. This is in line with the modular approach of the s.c.r.e.a.m. framework; a developer could easily rewrite the code of the `scream_ipc` library to modify its behavior without needing to rewrite any of the other components of the s.c.r.e.a.m. framework.

4.4 Architecture and Functionality of the Brain

Most of the modules in the s.c.r.e.a.m. are fairly straightforward. The mixer mixes music; instruments take musical information and produce music; sensors take environmental data and send this information to the brain. However, the brain itself is a complicated structure.

On the most basic level, using the specifications put forth in the top-level abstraction, the brain takes environmental data and past output and uses those to generate musical information, which it sends to each individual instrument.

However, this concept must be elaborated on for this project's implementation of the brain. The planned implementation will use detailed probabilistic models to compose and modify the music. These models will be contained in simple text files, and will contain probabilities of the form

$$P(X_n|C_1C_2...C_n) = x$$

This gives the probability of the event X_n given any number of conditions C_1 through C_n. However, the list of probabilities is of an arbitrary length, so a full probability tree for each event X_n and condition C_n cannot be created. Therefore, the brain will need to use statistical inference to estimate the probabilities it has not been given.

This method of statistical inference allows for incredibly simple as well as intensely complicated probabilistic models defined in configuration files.

4.5 Coding and Standards

C and C++ have been chosen as the primary programming languages for the project, with Bash shell scripts being used to assemble the components into a working system at runtime and tear down the components at exit. The target compilers for the C and C++ code will be the GNU Compiler Collection (gcc) version 4, though conformity to C99 and ISO C++ standards for the code, to maximize portability, is highly desirable. Building will be entirely automated with GNU Make, and revision control will be handled by the Concurrent Versioning System (CVS).

It is intended that the s.c.r.e.a.m. framework will use as few third-party or external libraries as possible. Therefore, the only third-party libraries to be linked against by the vast majority of the s.c.r.e.a.m. code should be the C and C++ standard libraries and POSIX system libraries. However, it would still be useful to use a third-party library for final-stage audio output and mixing. Candidates for a library to be used for this are the JACK Audio Connection Kit (JACK) and the Simple Directmedia Layer (SDL). Through abstractions in the s.c.r.e.a.m. code, changing the audio output library being used should be a trivial task.

The coding standards being followed will have the same strict commitment to modularity as the system as a whole. Any calls to third-party library functions will be wrapped with functions or classes that hide any pecularities of the third-party system behind a common abstraction that can be maintained even if the back-end library is changed.

Portability is also a concern; as a result, the data types of all interfaces will be expressed in terms of types defined in `stdint.h`, which defines types in terms of signedness and bit width (for example, `uint8_t` is defined as an 8-bit unsigned integer). `stdint.h` was chosen because it is part of the standard C and C++ libraries. However, `stdint.h` makes no guarantee of byte order. This could lead to complications with network interaction between machines with different endiannesses. This incompatibility is noted, but ignored due to the overwhelmingly Intel-centric development environment. A solution to this problem would be to convert all integer data in all data structures transmitted and received into network order; however, this process adds considerable overhead and complexity to an otherwise seamless task, so it will not be done.

4.6 Alternatives and Tradeoffs

There is a significant number of tradeoffs involved in creating a system the size of the s.c.r.e.a.m., many of which affect the design of the project deeply. The fundamental specifications of the design were an effective guide in determining which path to take at each step of the design process. From the inception of the ideas behind the s.c.r.e.a.m., the goal has been a modular and loosely cooperating collection of independent processes and not a single monolithic process. The ability to operate transparently over network connections further fed into the decision to create s.c.r.e.a.m. as a constellation of programs instead of a single program, and was a primary factor in the decision to use first-in-first-out (FIFO) data structures for interprocess communication.

5 Schedule, Tasks, and Milestones

The work flow described in Figure 3 breaks down the development of the s.c.r.e.a.m. into weeks and tasks. Weeks are assigned capital letters beginning with the week of January 20 and skipping the week of March 16, which is assigned an asterisk instead. The final two weeks are referred to by symbols and will not be used for active project development. Lowercase letters are assigned to tasks, which are either independent or lumped into one of

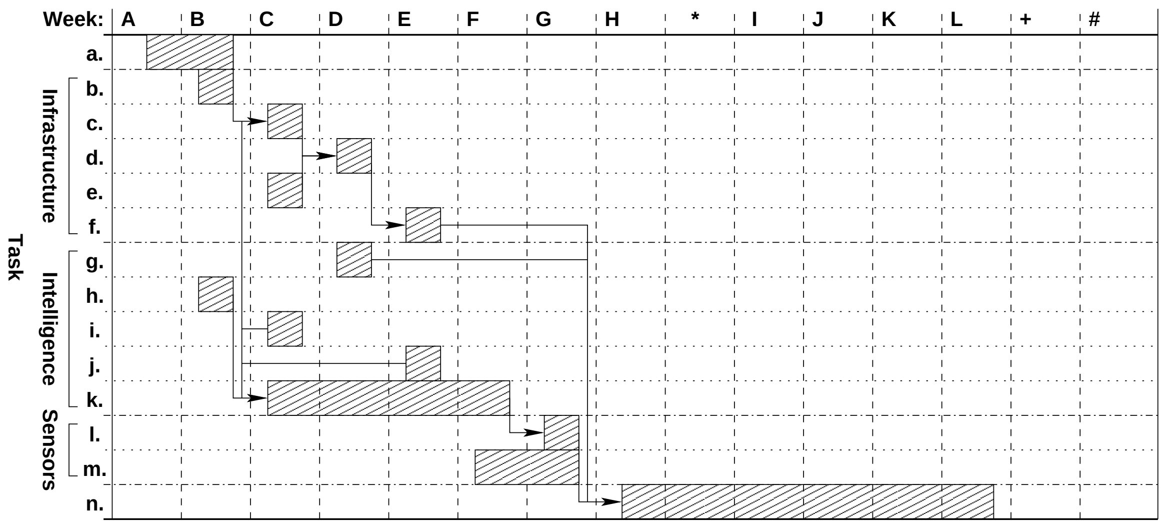

Figure 3: Workflow for s.c.r.e.a.m. development.

three broad categories. These tasks are:

a. Define and document all interfaces.

b. Develop basically functional interprocess communication library.

c. Develop an audio output library.

d. Develop a simple sample-based synthesizer.

e. Devise a format for sample sets for the sample-based synthesizer.

f. Develop a synthesizer that plays sound from the microphone without feedback.

g. Develop a minimal instrumentalist.

h. Develop a theoretical basis for the brain.

i. Define a file format for the brain's probability tables.

j. Write a minimal probability table for the brain.

k. Develop and code the brain.

l. Add hooks for sensor input to the brain.

m. Build a simple sensor with a serial interface on a protoboard.

n. Test and adjust the prototypical system.

The major milestones in the development of the s.c.r.e.a.m. are the moments when new combinations of components can be tested and heard. The first such milestone will be in the week of February 17 when the sample-based synthesizer is ready for testing with a simple instrumentalist to drive it. After that, in the week of February 24, the playback of sound processed from a microphone input without audible feedback will provide another checkpoint. By the weekend of March 2, the s.c.r.e.a.m. should make its first completely autonomous sounds. From then on, the development team's only goals are to improve the quality of the music produced and provide more ways for it to be controlled.

6 Project Demonstration

Demonstration and evaluation of the s.c.r.e.a.m. is simple and complicated at the same time. To prove that the system works and is functional, one merely needs to run it and listen to the output, and from there, determine if it is music. However, in-depth evaluation of the s.c.r.e.a.m. is not so easy. Since the system depends heavily on random values, testing particular pieces of the system is unreliable. If, for example, the effect of the sensors on the music wanted to be shown, one might consider changing the sensor's input drastically. Unfortunately, though, this might not always change the music, since the system could randomly decide to ignore the sensor's input at that point in time.

Instead, quantitative evaluation of the s.c.r.e.a.m. as a whole requires probability analysis over time. Metrics would need to be designed to measure the music, and would need to be recorded over time. Sensor stimuli would also need to be recorded and measured over time. This process is tedious and difficult, and therefore will not be attempted in this project to evaluate the system.

A better approach to evaluating the system is instead to examine each individual component. For example, to evaluate the brain, the musical data being sent to each instrument should be observed. Again, this will be hard to make conclusions about because it depends on a random element. However, components like the synthesizers and the mixer should be much easier to evaluate, since there is no random element in those particular components. Therefore, this method of examining each module in the s.c.r.e.a.m. will be used to evaluate its performance.

Qualitative evaluation of the system, though, will be performed by the simple act of listening to the output. If this output satisfies the design goals established earlier for this project's particular implementation of the s.c.r.e.a.m., then the project can be classified as successful.

7 Marketing and Cost Analysis

As a project with primarily aesthetic goals, the s.c.r.e.a.m. does not strive to be marketable in the same sense that an ingot of copper or a loaf of bread are marketable, or even in the same sense that a violin is marketable. What the s.c.r.e.a.m. provides is unique form of expression and a set of tools to accomplish that expression, most of which exist only as software, and are thus freed from any of the physical restrictions associated with tangible items.

7.1 Possible Target Customers

The opportunity to experience the s.c.r.e.a.m. on some level is an inalienable right granted to anyone with a functioning set of ears. Anyone who listens to the sound generated by the s.c.r.e.a.m. is devoting a portion of their resources to processing this data, and is therefore the only kind of customer the s.c.r.e.a.m. has a desire to create. Others may take it upon themselves to obtain the source code and create their own derivative works based on the s.c.r.e.a.m., thus devoting their resources to enhancing the project. These customers are on a different level, and obtaining them requires a method of spreading knowledge of the project.

7.2 Marketing Ideas

The primary method of marketing the s.c.r.e.a.m. is simply allowing people to hear the music it produces. The easiest way to do that is to perform it; devote computing resources to it and allow it to play on speakers that are within earshot of large numbers of people. If people who are capable hear it and understand its source, they will want to obtain the s.c.r.e.a.m. for themselves and perhaps even augment it. Therefore, if it is successful, the most powerful marketing tool for the s.c.r.e.a.m. is the s.c.r.e.a.m. itself.

7.3 Cost Analysis

Free sample parts that were already on hand will be used to create a sensor system on a breadboard that had already been obtained for other purposes. Computers that would have already been running anyway will be used for project development and testing. Therefore, the cost of the physical components used to make up the s.c.r.e.a.m. rests at an easy zero dollars.

The Orange Lunchbox Brigade, a team of two engineers, will work tirelessly to complete the s.c.r.e.a.m. by the set deadlines. As believers in their own product, they hedge their ability to profit on its success. Because of this, they believe it is only fair for all profit produced by s.c.r.e.a.m. to be split evenly between them.

8 Summary

The **S**ystem for the **C**reation of **R**andom **E**lectronic **A**daptive **M**usic (s.c.r.e.a.m.) is an open-ended system for the generation of random, environmentally-modified music. Inspired partially by the ideas of John Cage, Karlheinz Stockhausen, and other leading experimental musicians, the system is meant to be able to create any sort of music, even that which is currently considered outside the realm of traditional music.

By using a simple top-level abstraction, the s.c.r.e.a.m. framework can be split into four basic parts: the brain, the instruments, the environmental sensors, and the mixer. The brain uses the input sent by the environmental sensors as well as the previous output of the entire system to decide on the structure, form, tempo, rhythm, and chordal structure of the music. This information is given to each of the instruments, who in turn decide the exact notes and rhythms that they will play, based on the sent information. These notes and rhythms are then synthesized and sent to the mixer, which mixes all of the sounds to produce one sound output.

It should be noted that any number of mixers, instruments, environmental sensors, and even brains can be used. This allows for endless possibilities in instrumentation, performance, and composition of the music.

The s.c.r.e.a.m. framework will be a collection of libraries written mainly in C and C++. The code will conform as closely as possible to C99 and ISO C++ standards, to make the system portable to almost any computing environment. Each module of the system (brain, instruments, sensors, and mixers) will be a standalone process which communicates with all the other processes.

However, this project also encompasses the simple implementation of the s.c.r.e.a.m. framework. This implementation will create simple music of a specified genre, using a few simple instruments, one mixer, one brain, and a couple environmental sensors. A humidity sensor has already been designed and could be used for this purpose. Another planned sensor is a microphone; however, since the microphone will be in the same environment as the music is being played in, an adaptive filter must be used to cancel out any possible feedback effects so that the system will not become unstable.

Overall, the s.c.r.e.a.m. framework will provide a limitless platform for the generation of random music. Depending on the implementation of the system, it could be used to create engulfing orchestral works like Vivaldi's 'The Four Seasons', or revolutionary artistic works like Miles Davis' 'Bitches Brew', or even bizarre experimental music like Karlheinz Stockhausen's 'Helikopter-Streichquartett'. However, the real beauty of the system lies in its open-endedness; it is capable, in theory, of designing music far outside the bounds of what human ears comprehend as music. The possibilities are endless.

References

[1] S. C. Funk, "John Cage: Avant-Garde Composer." http://classicalmusic.suite101.com/article.cfm/aleatoricperformance, Apr 2006.

[2] S. Surovec, "History of electronic music: - the early years." http://nmnm.essortment.com/musicelectronic_rccz.htm, 2002.

[3] K. Stockhausen, "Official Short Biography of Karlheinz Stockhausen." http://www.stockhausen.org/biography.html, 2004.

[4] "iMuse Island: What's iMuse?." http://imuse.mixnmojo.com/what.shtml, 2004.

[5] U. Berggren, *Ars combinatoria: Algorithmic construction of sonata movements by means of building blocks derived from W. A. Mozart's piano sonatas.* PhD thesis, Uppsala Universitat, 1995.

[6] R. Curtin, "Automated music generation." ECE4884-L04: Senior Design Project, Jan 2008.

[7] J. W. Wieder, "Generating music and sound that varies from playback to playback." U.S. Patent 7,319,185, Jan 2008.

[8] C. Browne, "System and method for automatic music generation using a neural network architecture." U.S. Patent 6,297,439, Aug 1999.

[9] C. D. Kersey, "Humidity sensor with threshold indicator (a last-minute redo)." ECE4175: Embedded Microcontroller Design, Dec 2007.