Music Generation Using Cellular Models

Heather Morris Gabriel A. Wainer

Carleton University Centre for Visualization and Simulation (VSim)
1125 Colonel By. Drive
Ottawa, ON. K1S-5B6 Canada
hmorris@connect.carleton.ca; gwainer@sce.carleton.ca

ABSTRACT

Music is an art form that is created by people who draw inspiration from a variety of sources. However, the basis of music is an organization of sound frequencies, timed in a certain manner. Using the basic principles of music, it is possible to develop a mechanism for the automatic generation of music. The goal of this research project is to explore the mapping of cellular models into musical combinations, using music theory. To accomplish this task we developed a mapping tool for the automated creation of music using cellular models. In particular, the main focus is determining patterns embedded in the Game of Life. A music modulation model is developed in order to model the chord level of the musical composition. A two layered mapping technique shows that the musical composition can become achieved.

1. INTRODUCTION

Music is often thought to be created from a person by external influences. These influences give the composer the power to create beautiful sounding music from deep within their soul. The emotion inherent to music leads to the belief that only humans are capable of creating beautiful compositions. However, the basis of music is an organization of sound frequencies, timed in a certain manner. Also, there is a certain range and discrete values of frequencies used in the creation of music. This implies that there are a discrete number of frequencies that can be used. Since music is also time based, it can be extrapolated that a model can be developed or the musical system, which determines the frequencies to be played. This model can be described as a discrete event system because of the organization of frequencies.

The goal of this research is to explore a different mechanism for the creation of music. The idea is to explore the mapping of cellular models into musical combinations, using music theory. A mapping tool is developed for the automated creation of music using cellular models. Various existing cellular models have shown to form interesting patterns [1], and, in some cases, they have similar patterns to those of musical structures. In this article we focus on determining patterns embedded in the Game of Life, as proposed by Conway [2]. A complex mapping technique is defined and subsequently used to extract the layers of musical composition. This mapping technique uses a top-bottom approach, where the highest musical structure will determine the lower structures.

In the following sections we discuss the background for the project in terms of modeling and simulation, and music theory; then, we present the mapping mechanism, and we discuss the implementation using varied tools. A few examples of music and their cellular models will be posted at http://www.youtube.com/arslab.

2. BACKGROUND

2.1 Introduction to Cellular Models

There are different formalisms to define cellular models. In particular, a Cellular Automaton (CA) is a time based, discrete event model that uses a cell space or a grid. This cell space gives the model spatial sense. The global transition function contains a set of rules which are executed. Each cell on the grid contains a transition function that will determine the next state of the cell, taking into account the current state of the cell and the state of certain cells around it. The group of cells that influence the next state of the cell are said to be part of the cells neighborhood. When evaluating the cell's state change the states of the cells in the neighborhood are also considered.

Neighbor cells can be close or far, and use different shapes. Figure 1 Shows some common neighborhoods used. Moore's neighborhood includes the origin and the eight cells surrounding it; Von Neumann's uses the cells above, below, and to both sides. Hexagonal neighborhoods are very popular because they provide higher isotropy, the capacity to represent equivalent behavior in every possible direction. However, square meshes are popular due to the ease of mapping and visualization.

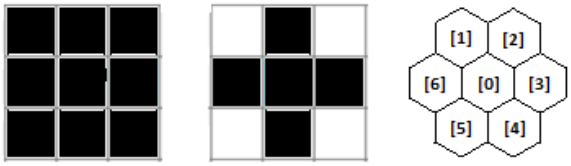


Figure 1: Moore's, Von Neumann, Hexagonal

CA can be formally specified using the following notation

$$CA=<S,n,C,N,T,\tau,q.Z_o^+>$$

where S is the alphabet used to represent the state for each cell; n is the dimension for the cell space; C is the state set for the cell space; N is the neighborhood set; T is the global transition function; τ is the local computing function; $q.Z_o^+$ is the discrete time base for the cellular automata.

CA are n-dimensional cell spaces that progress in time on discrete time steps, where the state of the cell can be taken from the alphabet of states in S. The state of the cell space changes by executing the global transition function. The global transition function is affected by the local computing function changing the states of the neighboring cells. These functions are computed synchronously and in parallel for every cell in the cell space.

2.2 The Cell-DEVS Formalism

The use of discrete-time for CA has been discussed thoroughly in literature [3]. The use of a discrete time base also constrains the model precision. Cell-DEVS [4] solves these problems by using DEVS [3] to create a cell space in which each cell is defined as an atomic model. The goal is to build discrete-event cell spaces, improve their definition by making the timing specification more expressive, and allow the combination of the cellular models with discrete-event models in a straightforward fashion. In music, different instruments can be playing at different timescales, which makes the timing management complex for CA. Also, modern musical forms use asynchronous rhythms, which are complex to define using a time-based approach.

In Cell-DEVS, each cell of a cellular model is defined as an atomic DEVS model. Cell-DEVS atomic models use **N** inputs to compute the future state **S** using the function **τ**. The new value of the cell is transmitted to the neighbors after the consumption of the delay function. **Delay** defines the kind of delay, and **d** its duration.

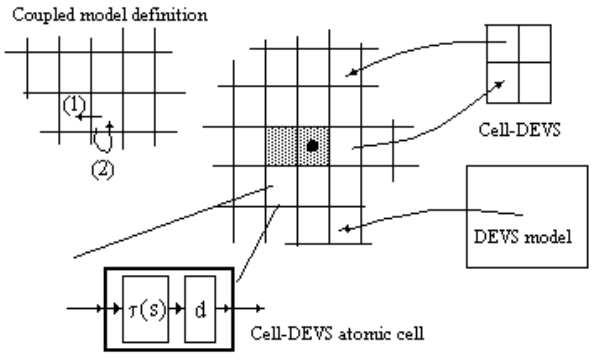


Figure 2: Informal Definition of Cell-DEVS [4].

Once the atomic cell model is defined, a number of cells can be put together to form a coupled model, built as an array of atomic cells. A Cell-DEVS coupled model is defined as an array of atomic cells with size $\{\mathbf{t_1}\ \mathbf{x...x}\ \mathbf{t_n}\}$. Each cell in the space is connected to the cells defined by the neighborhood **N**. The border **B** cells can use a different behavior than the remaining cells. The **Z** function allows one to define the internal and external coupling of cells in the model. $\mathbf{X_{list}}$ and $\mathbf{Y_{list}}$ are input/output coupling lists and are used to interchange data with other models.

The **CD++** tool [4] was developed based on the definitions of the DEVS and Cell-DEVS formalisms. Models can be described using a built-in specification language, which provides a set of primitives to define the size of the cell-space, the type of borders, a cell's interface with other DEVS models and a cell's behavior. The computing function is defined using a set of precondition and post condition rules, offset by a delay. If the precondition is satisfied, the cell takes on the value of the post condition after the given delay. (If the precondition is false, the next rule in the list will be checked until there are no more rules.

2.3 The Game of Life

The game of life is a solitaire-like game that was proposed by the mathematician John Conway. The game was originally intended to simulate the rise and fall of a society of organisms. The game is played on a grid, where a cell on the grid is either alive or dead. A cell stays alive if it has two or three neighbors that are alive. The cell dies from overpopulation if there are four or more neighbors alive, and die from isolation if there are one or less neighbors alive. A new cell is born if there are exactly three neighboring alive cells, using Moore's neighborhood.

There are four possible outcomes to the Game of Life. First, the cells in the space will all die, which means that the population has become extinct. Second, the population will grow indefinitely. Third, the population reaches a steady state, such that there is no change over time. Fourth, there is a repeated pattern in the cells with a known period.

```
[life]
width : 20              height : 20
delay : transport       border : wrapped
neighbors : (-1,-1) (-1,0) (-1,1)
neighbors : (0,-1)  (0,0)  (0,1)
neighbors : (1,-1)  (1,0)  (1,1)
localtransition : new-life-rule

[new-life-rule]
Rule: 1 10 { (0,0) = 1 and ( truecount = 3
                    or truecount = 4 ) }
Rule: 1 10 { (0,0) = 0 and truecount = 3 }
Rule: 0 10 { t }
```

Figure 3: The Game of Life Implemented in CD++

Figure 3 shows the implementation in CD++: we can see from the neighbors that Moore's neighborhood is used, and that the precondition/post condition rules are the ones defined earlier. For instance, the first rule says that if the origin cell *(0,0)* is alive, and there are 3 or 4 living neighbors, determined by the *truecount* function, the cell remains alive. This information is spread to the neighboring

cells after 10 time units. The last rule is executed if no other rule is valid.

2.4 Musical Notation

Music is the organization of audio frequencies in a time-based system. In western music there is a defined set of frequencies that are used in musical composition. The frequencies are based on a logarithmic scale where the note is said to be an octave higher than the previous note if it has a frequency that is ten times larger. An octave is then parsed into twelve half tones along the logarithmic scale. Each note within the scale has a corresponding octave such that the frequency pattern is repeated. This allows for a repeated notation language, which associates letters to the note. A note an octave higher than another note will also have the same letter. The letters used are the A through G [5].

Music is displayed using a graphical notation. There are five parallel lines used to indicate which notes should be played. The spaces between the lines are also used. The frequencies can be scaled on these five lines by a clef, which is a tool that indicates the mapping of the desired notes to the lines. For example, when a treble clef is indicated at the start, the lines starting from bottom to top indicate the notes E,G,B,D.F. If a bass clef is indicated the notes are G,B,D,F,A. It is also important to know the octave in which these notes are associated. Since the naming convention is repeated these notes can belong to any octave. It should also be noted that the frequency of the note increases with ascending lines, so the line on top has the highest frequency.

2.5 Mapping Cellular Models to Music

The results of an executable cell space can be used to map audio frequencies. That is, the frequencies that are mapped onto the space correspond to the frequencies used in musical compositions. Two mapping techniques are investigated; Camus and a system developed by Paul Reiners.

Camus was developed by Eduardo Miranda as a mapping tool for music composition. It uses a Cartesian space approach in order to represent musical triples [6]. A music triple is a set of three notes which will be called <A,N,D>. The Cartesian space is a two dimensional space in which the horizontal coordinate represents the distance between the notes A and N and the vertical coordinate represents the distance between N and D. The distance between the notes is represented by half tones.

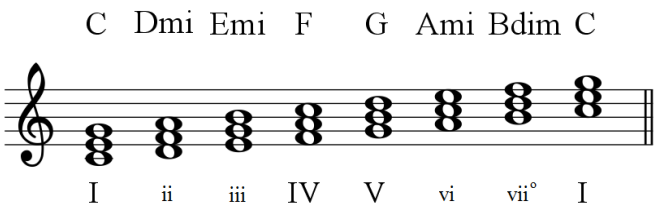
To begin the composition, the cell space is initiated with a starting condition. The cell space is then checked and the live cells are analyzed to create the musical triplets. Each live cell is given a base value on which the other two notes in the triplet can be based upon. The cell space then goes to the next state following some rules from the Game of Life and the space is checked again for live cells to play.

Camus considers the whole cell space when determining which frequencies to be played. Some mapping techniques only use a section of the cell space. For instance, a mapping technique proposed by Paul Reiners is a Row-to-Binary approach [7]. This technique takes a row in the cellular model, and converts the cells true or false sequence into a binary number, where true is a one and false is a zero. This binary number can then be directly converted to a frequency, and then played.

The mapping techniques can be layered such that the current state from one of the model influences the audio mapping on the second. In these mapping techniques there is a model that is used for each layer of the hierarchy of the musical structure.

2.6 Musical Structures and Hierarchy

Music is comprised of several layers that are a part of a hierarchical structure. The notes compose the lowest tier structure. Notes are organized in a scale, which contains seven different notes within an octave. There are two different types of scales, major and minor scales. The type of scale dictates the notes that are played within that scale. From each note on the scale there is an associated three note chord. An example of a scale with the three note



chords on each note can be seen in Figure 4.

Figure 4: Chords Based of a C Scale

Organizing the chords from the scale into a certain sequence results in a phrase. A phrase is a sequence of notes that follow a chord progression. There are several musical styles that group phrases in certain orders. For example, the rondo forms a phrase order that is ABACA. This means that the A phrase is repeated with different phrases B and C in between. There is also the minuet form which has the phrases ordered as ABACDC, where the first phrase is repeated after the second phrase and so on.

Phrases can also be expanded to contain more than one line of music. An example is when two different instruments are playing. This will comprise the score with the layering of different phrases.

2.7 jMusic Framework

When mapping a cellular model to the musical structure, there is a need for the creation of audio files that could be played in a computer. This is accomplished by using an

open source library created by Andrew Sorensen and Andrew Brown, called the jMusic Framework [9]. The jMusic framework is open source and allows music programs to be written in Java. It uses the Java sound API to create the audio file. The framework is built upon a similar structure as musical composition. There are notes that make up phrases, phrases that make up parts, then parts that make up scores.

There are many functions and classes within jMusic that are particularly useful. The *note* class is an array of values which represent the notes from classical western music. These values in the array can be written as the midi specifications of the notes, using values from zero to one hundred and twenty seven. The notes in the array are organized in a fashion that there is an associated pitch and length of the note. The *phrase* class contains the sequence of notes with their associated duration. The *phrase* class can contain more than one of the note classes. If there is a note class added to the phrase class, the new notes get added to the end of the phrase. Similarly there is a *part* class that contains phrases, and a *score* class that contains parts.

A built-in function is used to create midi audio files from the *score* class. The instrumentation can also be determined from a selection of different instruments such as piano, trumpet, and clarinet.

3. MODEL DEFINITIONS

In this section various methods used to define the models to automate the music creation are discussed. In the first section, the techniques used for mapping layers are defined. In the second section, the model used for chord progression is described. In the third section, the method for mapping the cell space for chord modulation is presented. The implementation of the model is in CD++.

3.1 Layered Mapping Techniques

The technique used in the music composition simulation is a two layered mapping technique. One layer determines the notes that are played, while the other determines the phrase level with the chord progression that is present in the piece. The model that determines the level of the notes is a cellular model of the Game of Life although the cell space chosen and the rules used to define the cell space behavior can be changed by simply choosing a different set of rules. Certain cells within the cell space are assigned different notes, so when the cell is activated, the corresponding note is played.

The second model is a cellular model as well, which has a mapping of the chords within the cell space.

3.2 Chord Progression Model

One layer of the musical composition is the chord progressions. There have been a couple models developed which provide rules to switch from one chord to the next. The model used in this project is that proposed by Frederick Horwood [10].

There are three primary triads for every scale. These triads are the chords made from the I, IV, and V notes of the scale. These chords can be combined in any sequence. From these three chords, there are six possible combinations.

The secondary triads are the chords that are made from the ii, iii, and vi notes on the scale. These chords have special rules when combining them with the primary chords. The chord built on II is a minor triad and it can be used when followed by a V chord. This chord can be approached from the I or VI chords. The chord built on III is also a minor chord and is used less often than the other secondary triads. When used in a chord progression it must be preceded or followed by a I or VI chord. Also, it can be followed by a IV chord. The VI chord may be used on any occasion as a substitute for I. Taking every combination of the chords, this sums up to a total of 18 possible chord progressions as can be seen in Table 1.

Table 1. All Possible Chord Progressions

I-V	I-IV	V-IV	IV-I	IV-V	V-I
ii-V	I-iii	iii-I	iii-VI	VI-iii	iii-IV
vi-V	V-vi	IV-vi	vi-IV	I-ii	VI-ii

3.3 Mapping a Cell Space for Chord Modulation

The chord progression model is defined as a cellular model. The cell space is comprised of hexagonal cells as opposed to the more common square cells. Each of the hexagonal cells is mapped to a corresponding chord. The placement of these chords is such that the chord number in any direction is known. See Figure 5 for the layout mapping of the chord numbers.

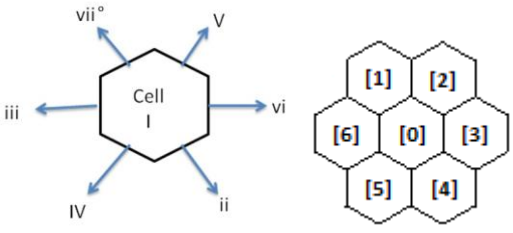


Figure 5: Cell Mapping for Chord Progression Model

In Figure 5, it can be seen that in the direct neighborhood, the cell in the [0] position is the I chord. The cell in the [1] position is the VII chord and the cell in the [2] position of the V chord. The cell in the [3] position is the VI chord and the cell in the [4] position is the II chord. The cell in the [5] position is the IV chord and the cell in the [6] position is

the III chord. This can be extended for every known chord and it will have a mapping grid that looks like the one shown in Figure 6. A representative sample of chords is used in Figure 6 to present the concept behind the mapping of chords to a cell space. In Figure 6 the chords are written in terms of the note letter that the chord is built on. A plus (+) represents a major chord and a minus (-) represents a minor chord.

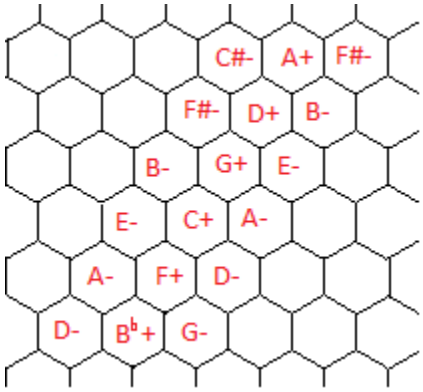


Figure 6: Mapping of the Chords to the Cell Space

3.4 Defining Rules and Cell States

A chord change in the model is defined as a transformation from one chord to another. Chord changes are referenced from the I chord. The states on the cells must contain information on the chord number the cell is representing and which chord number is to be played next. The state of the cell is thus defined as a two digit number where the first digit represents the chord number for that cell, and the second digit represents the chord that is coming next. For example, if the state on the cell is 25, then that cell is on the II chord and the next cell to be activated is the cell that is in the V direction.

The rules are defined in such a way that there is only one active cell in the cell space. This cell corresponds to the current key of the composition. An initial state of the system is defined by initializing a cell with a chord number and specifying the next desired chord number. Since this is a unique cell state, the next cell that has to be activated is known. For a cell with the state 62, then examining the mapping of the chords on the hexagonal cell space reveals that the cell representing the II chord is in the [5] position from the current cell. In this case the rule states that if the cell in the [2] position of the cell has the state 62 then the cell activates with a state of 25. Another rule is that a cell has a state of zero if there is any value in the cell, including zero.

3.5 Defining the Model in CD++

The model is implemented using CD++. Figure 7 shows the initialization of the model. There is a cell space that is 9 by 9 cells with 6 output ports. These output ports correspond to the 6 cells that are activated during the simulation. These output ports provide messages for when each of the cells is activated. Therefore it is possible to know what chord is to be played at each time step.

```
components : MusicModulation
out : output
link : out1@MusicModulation output
link : out2@MusicModulation output
link : out3@MusicModulation output
link : out4@MusicModulation output
link : out5@MusicModulation output
link : out6@MusicModulation output

[MusicModulation]
out : out1 out2 out3 out4 out5 out6
link : out@MusicModulation(6,5) out1
link : out@MusicModulation(6,6) out6
link : out@MusicModulation(6,4) out3
link : out@MusicModulation(7,5) out4
link : out@MusicModulation(7,6) out2
link : out@MusicModulation(5,6) out5
type : cell
dim : (9,9)
delay : transport
defaultDelayTime  : 100
border : wrapped
neighbors : (-1,-1) (-1,0) (-1,1)
neighbors : (0,-1)  (0,0)  (0,1)
neighbors : (1,-1)  (1,0)  (1,1)
```

Figure 7: Coupled Modulation Model

The rules are defined using the hexagonal neighborhood; the *ltrans* (lattice translator) function converts rules written with the direct hexagonal neighbors into square celled rules. It can be seen that the neighborhood defined using the CD++ specification is a Moore's neighborhood. This transformation is carried out analyzing the parity of the row where a cell is located. Considering a cell that is in the position (x,y), where x is the row and y the column, the position is translated as shown in Figure 8.

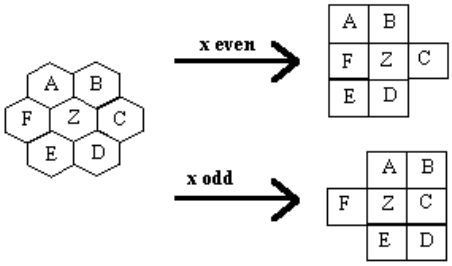


Figure 8: Distribution of close neighbors in the mapping process.

The mapping process results in a correspondence that differs according to the original row position. For instance, from the example described in section 3.4

```
rule: 125 50 { [0] = 0 and [4] = 25 }
```

We obtain the following translated version

```
rule: 125 50 {(0,0)=0 and (1,0)=25 and even(cellpos(1) }
rule: 125 50 {(0,0)=0 and (1,1)=25 and odd(cellpos(1) }
```

After the mapping process, the model is combined with a 16x16 Game of Life model. Output ports are added to provide the messages about which cells are activated. The cells that are of interest are the cells at the bottom row of

the cell space. These cells are of interest because these are the cells that provide initiation of the music notes for the model.

4. SIMULATION RESULTS

4.1 Music Modulation Model Verification

First, the music modulation model will be verified. The model is initiated with an initial cell value of 54 in the cell (5,6). From the model definition, the expected result is that the cell representing the I chord, or in this case the cell in the [5] position from the cell (5,6) should have the state 154. This test confirms that the intermediate states between the V and IV chords are working properly. The result is obtained using the CD++ modeling specification and the results are shown graphically in Figure 9. Also, it can be seen that there is only one cell active at any given time. The cell representing the IV chord has a proper state where the first digit in the state value is the chord number. Additionally, the next desired chord is one using a proper chord progression. This test indicates that the model defined using the CD++ modeler is capable of correctly defining the model.

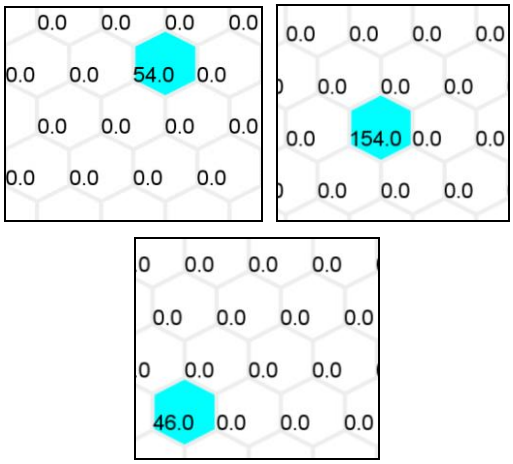


Figure 9: Verifying the Modulation Model

Currently, the model does not allow for the random selection of the next desired chord. The random selection of subsequent chords is not implemented in the definition of the model in CD++. As a result, a repetitious pattern in the chord progression eventually develops. This pattern has a period of about 8 chords. Many compositions use a repetitive chord progression, for example Pachelbel's Canon in D Major. The entire song is based on a chord progression of a period of 8 chords and is a very popular piece of music. Also, the music form known as the Theme and Variation uses the same chord progressions from the introductory theme and modifies the melodies on top of that chord progression in subsequent variations.

4.2 Converting the Simulation Results to Music

Music is created from the cellular model using the mapping techniques. The two layered technique for mapping. The first layer of the music composition is defined by the music modulation model which has been developed and verified. The second layer is from the results from the Game of Life.

There are sixteen cells that are of interest along the bottom row of the cell space. These cells are mapped such that the notes from an ascending music scale are placed in the cells. This mapping can be seen in Figure 10 for seven of the cells. This is expanded for the sixteen cells.

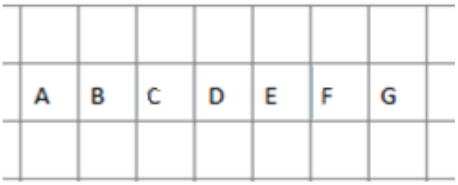


Figure 10: Mapping the Notes to the Game of Life

By knowing which cell is activated at a certain time and what the corresponding note for that cell is, music can be created by playing back the frequencies. The music frequencies are played back by using the jMusic framework.

Based on this, a Java application is built to convert the layers into midi files. The input file from the model must contain one column of midi pitch values, where each row represents the current pitch to be played. The midi pitch scale is a scale with integer values between 0 and 128. The note called middle C is given the value of 60 on the midi scale. Each increase of one on the midi scale is an increase of half a tone of a note. Using the midi scale, the notes in the cells in Figure 10 could also be represented by the numbers as seen in Figure 11.

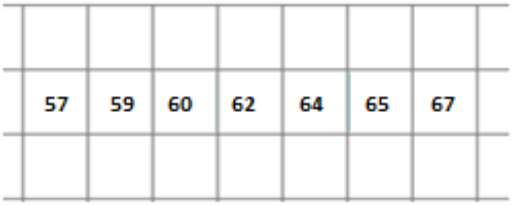


Figure 11: Mapping the Midi Scale to Life Game

The values from the input file are then stored into a note array. This note array is a special class from the jMusic framework. The note array contains a sequence of midi scale values which can be played back. These notes have to be placed within the hierarchy of the framework which mimics that of music. Once the score has been created a midi file with the pitches from the input file is created. Also, the framework allows for the printing music notation.

In order to create the input file, the output data from the Game of Life must be organized. The data is organized in

an excel spreadsheet. As defined in the mapping process, the cells on the bottom of the cell space are of interest. As a result of the mapping technique, the cell sending the messages to output port 1 is corresponding to the midi value 48, and the cell sending messages to output port 2 has the midi value of 50, and so on. When taking the results of the first test of the Game of Life, the piece of music seen in Figure 12 is created.

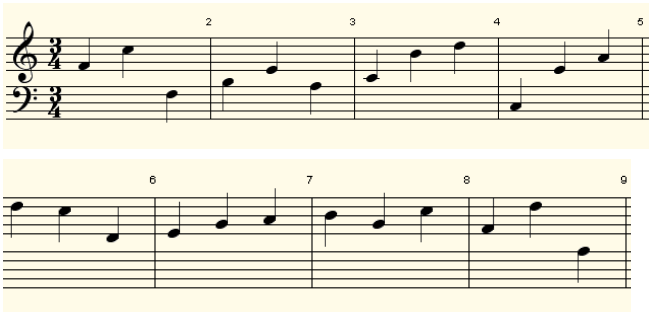


Figure 12: Musical Composition from One Layered Mapping of the Game of Life

The one layered mapping technique for the Game of Life works using the jMusic framework. Now the second layer of the mapping is applied. The modulation model is used as a scale from the original key. This means that the original values are scaled by a constant value depending on which cell is active in the modulation model. The outputs of the model are organized such that the output number corresponds to the mapped chord that the cell would represent. Also, passing the states through the I chord cell, the states like 154, need to be removed for the mapping process to function properly. After applying the two layered mapping technique to the Game of Life cell space, the following piece of music is created, a portion of which is seen in Figure 13 in music notation.

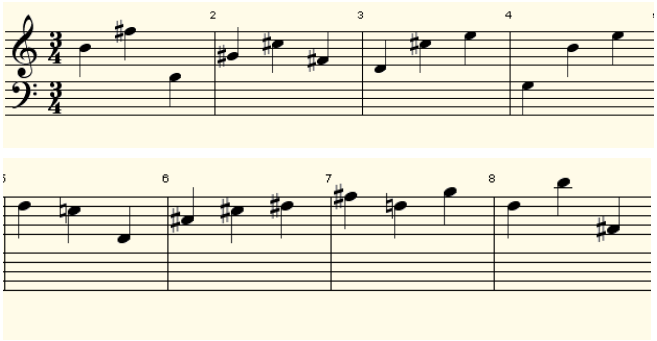


Figure 13: Musical Composition from Two Layered Mapping of the Game of Life

4.3 Alternative Mapping Techniques

The mapping technique presented thus far only uses a one layer approach to musical composition. Most current Western classical music includes harmonies. A modification of the mapping technique can create harmonies using the same results from the Game of Life. Listening to the two compositions developed using the developed model, it can be heard that there is a lot of large variation in the pitches of the piece. This alternative mapping technique will eliminate the jumping in the melodies by creating the second layer of music from the lower notes. It eliminates the jumping because it uses all of the 16 cells in the composition with the bottom 8 cells as one layer of the music and the top 8 cells as the melody.

4.4 Listening to Cellular Models

The implementation of the chord modulation model was hypothesized to have a higher level of organization for the selection of the pitches for the cellular model music. This model was thought to be able to create groups of sounds, as opposed to the random scatter that is typically generated in the final audio result. This mapping technique does show promise. From listening to the pieces without and with the music modulation model, it can be heard that there are specific groups of chords. However, the apparent randomness of the Game of Life causes the compositions to lack a fundamental structure in the note level. This randomness makes the resulting audio sound random.

There are many other possible ways to map cellular model to musical structures. The mapping techniques presented in this paper are more of a sequential method where the notes are played one after the other in order. There could also be mapping techniques where the length of the note can change as well. Also, there are many different layers to musical composition. These layers can be modeled and implemented in the mapping technique. An example is the overlaying form of the music which includes the Rondo or Sonata forms. A model could be developed that would allow the mapping to conform to the compositional structure of music.

5. CONCLUSIONS AND FUTURE WORK

We have showed how to use cellular models in musical composition. Cellular models have been shown to have similar patterns and forms to that of musical structures. It is for this reason that we believed they could be used to create varied musical compositions. The method for creating the musical composition was to map the cell space of the cell space to musical pitches.

A music modulation model was developed using the CD++ tool, in order to model the chord level of the musical composition. The second layer of the composition was mapped from Conway's game of life. From this layer the mapping was taken for the pitches of each of the notes. The two layered mapping technique showed that the musical composition can become more organized with the two layered mapping technique, but there is still refinement needed in the mapping.

There are still many layers of the musical composition that can be modeled to add to the complexity of the mapping scheme. Using different time functions in the cell-DEVS models would allow for creating different rhythms to the music, such as Latin beats. Also, there can be improvements to automate the mapping sequence. Currently the mapping is performed mostly in Microsoft Excel and organized by the user. A simple program could be created in order to automate the process. In conclusion, the two layer mapping technique for automatic music composition has been introduced and shown to create musical compositions following the rules defined in the model.

6. REFERENCES

[1] Wolfram, Stephen. 2002. A New Kind of Science. Wolfram Media, Inc.

[2] Gardner, Martin. 1970. Mathematical Games: The fantastic combinations of John Conway's new solitaire game "life". *Scientific America* 223 (Oct. 1970), 120-123. DOI= http://www.ibiblio.org/lifepatterns/october1970.html.

[3] Zeigler, B.; Praehofer, H.; Kim, T., 2000. *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. Academic Press.

[4] Wainer, Gabriel A. 2009. *Descrete-Event Modeling and Simulation: A Practitioner's Approach*. CRC Press, Taylor and Francis Group, NW.

[5] Horwood, Frederick J. 1948. *The Basis of Harmony*. Gordon V. Thompson Music, c/o Warner Bros. Publications, Miami, Florida.

[6] Miranda, Eduardo Reck. Evolving Cellular Automata Music: From Sound Synthesis to Composition. Technical Report. Sony Computer Science Laboratory Paris.

[7] Reiners, Paul D. 2004. Cellular Automata and Music: Using the Java language for algorithmic music composition. Technical Article. DOI= http://www.ibm.com/developerworks/java/library/j-camusic/

[8] Cellular Automata Music: An Interdisciplinary Project. Eduardo Reck Miranda. Interface. Vol. 22, Iss. 1, 1993.

[9] Brown, Andrew; Sorensen, Andrew C. 2000. Introducing jMusic. Australasian Computer Music Conference. Brisbane, Australia.

[10] Horwood, Frederick J. 1989. The basis of harmony. G.V. Thompson Music.