Преглед НЦД 21 (2012), 34–40

Georgios Drakakis
13 Thornton Road, Bebington, Wirral,
CH63 5PN, UK
Mícheál Mac an Airchinnigh
School of Computer Science and Statistics
University of Dublin, Trinity College,
Dublin 2, Ireland

COMPUTER GENERATED MUSIC FOR AN INTERACTIVE VIRTUAL MUSICAL INSTRUMENT

Abstract: In this paper we introduce an Interactive Virtual Musical Instrument that plays computer generated music notes. The approach taken for implementing the virtual instrument is based on unique marker tracking in the real world using live video feed. The relationship between music pitches and frequencies is discussed and analysed, as well as the complex behaviour that derives from simple models, such as cellular automata. Music notes are generated using both sine waves and cellular automata, and later used as the sounds played by the Interactive Virtual Musical Instrument.

Keywords: Cellular automata, marker, QR code, virtual musical instrument

1. Introduction

Music is merely one of the areas experiencing daily breakthroughs because of the use of computers. It is evident that computer generated music using mathematical models is taking music to a different level. For this paper, however, we combine such music with an interactive virtual musical instrument. The once unlikely scenario of a computer playing music due to an individual pressing on a piece of paper is implemented with the use of unique marker tracking. The instrument plays two different types of music notes; one generated with the use of cellular automata, the other using sine waves.

2. Background

Even though there are theoretically infinite sounds, the human ear cannot pick up the subtle difference between two similar frequencies. It was therefore necessary to splice sound –a continuous measure- into discrete pitches or frequencies, which are today's music notes. A rather interesting and useful property of sound is that it repeats itself; therefore discrete frequencies sound identical at higher and lower pitches when the frequency is doubled or halved respectively. Ergo, the international notation A, B, C, D, E, F, G, A, B, C...

A Cellular Automaton is a discrete model which consists of a collection of cells on a grid, each of which has a finite number of potential states or values. The state of each individual cell is set by applying a rule set to a number of neighbouring cells. This allows the grid to evolve at discrete time steps [1].

Cellular Automata can have any finite number of dimensions, the simplest being the 1D binary Cellular Automaton, for which each cell can only have two different states. Stephen Wolfram described 1D Cellular Automata that use the “nearest neighbour” rule as “elementary cellular automata” [2]. However, in “A New Kind of Science”, he admitted that “there was no indication that simple programs could ever produce behaviour so diverse and often complex” [3].

The subcategory of Computer Vision which merges the real world with the computer world is Augmented Reality (AR), allowing the user to be part of an application. AR commonly concentrates on graphics by projecting 3D models into live video feed, usually with the use of 2D marker tracking. For this project we focused on marker tracking. The ARTAG SDK (www.artag.net) uses 10x10 grid black and white markers; whereas nowadays more complex markers are widely used for commercial applications, originally known as QR codes (http://www.denso-wave.com/qrcode/). The two types of markers are indicated in Figure 1. ARTAG is originally written in C++. However, we used the Goblin XNA wrapper (http://graphics.cs.columbia.edu/projects/goblin/) for C# in this project.

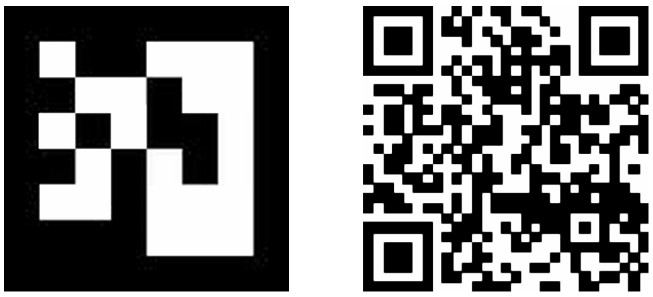


Figure 1. Left: ARTAG Marker; Right: QR Code

3. Methods and Implementation

In order to achieve optimal results with the virtual instrument, we chose to design it as a series of equally-sized, equally-spaced, fully aligned keys, somewhat resembling the white keys of a piano. We also added a button (top right of the virtual instrument) for switching between the two different types of computer generated sound. The prototype virtual musical instrument is shown in Figure 2.

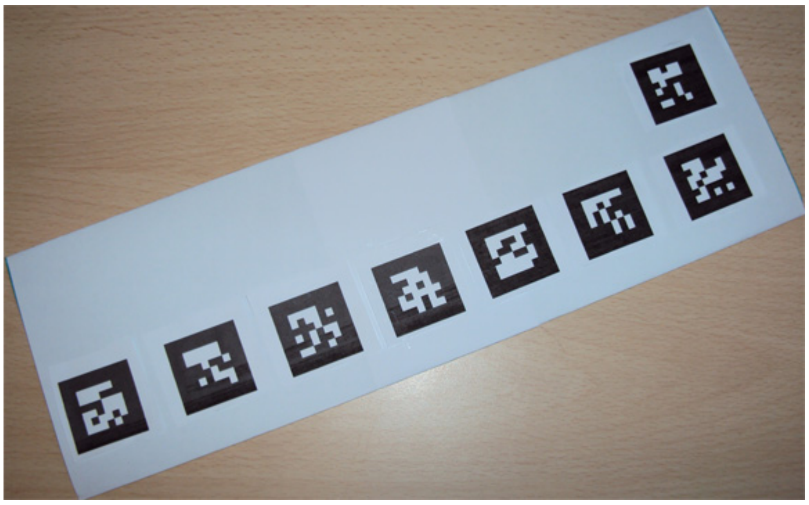


Figure 2. The Prototype Virtual Musical Instrument

3.1 Music notes from sine waves. The sine Waves represent a single frequency and are therefore a simple way to extract music notes. The standard form of the sine function is $y\ (t) = \sin\ (\omega t + \theta)$ where y is a function of time (t), θ is the phase and $\omega = 2\pi f$, f being the frequency. Between two consecutive A notes, there are another eleven discrete music notes, each of which has a corresponding frequency. The frequency of each note is equal to the frequency of the previous note multiplied by the twelfth root of two. Based on the online Wolfram Demonstrations Project "The Sound of Sine Waves" [4], after picking a starting note/frequency, we can calculate as many as we like with the following algorithm:

1. Divide the number of steps chosen to take by twelve.
2. Check if the quotient is larger than zero. This will show whether the note we are looking for is within the next eleven notes or the notes need to be repeated.
3. Take the remainder of the division and divide it by twelve. This will help us determine the number of steps to take, with or without the repetition of notes.
4. Set X as the sum of quotient from Step 2 and the fracture from Step 3.
5. Multiply the original frequency by two to the power of X.

Seven notes starting from C were generated using MATHEMATICA and exported as .wav files in ascending order.

3.2 Music from Cellular Automata. The choice of Cellular Automaton used for this application was somewhat arbitrary, as it was solely based on the fact that Rule 30 is Stephen Wolfram's favourite automaton of all time [5]. As elementary cellular automata rules use two-letter alphabets, we can plot Rule 30 using black and white squares on a grid with the use of MATHEMATICA software to show its behaviour (Figure 3). Each row of black and white squares is one time step. For example, step 9 of Rule 30 (B - Black and W - White) is:
W - B - B - W - B - B - B - B - W - B - B - W - W - B - W - W - W - B - B - B – W

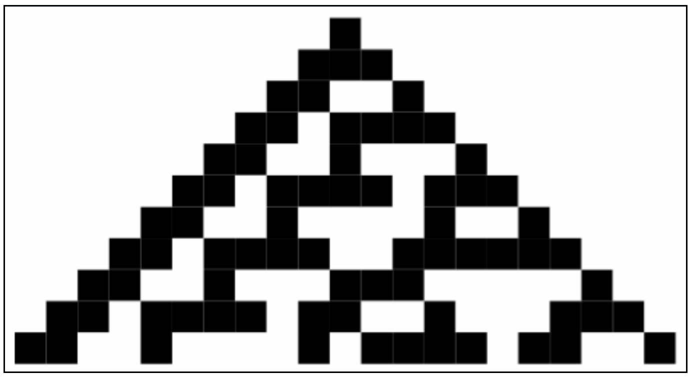
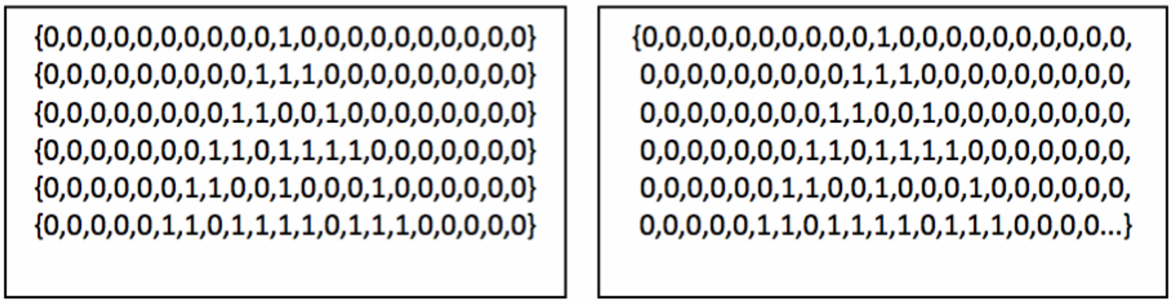


Figure 3. Rule 30 Cellular Automaton visualized using a black and white grid



```
{0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0}
{0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0}
{0,0,0,0,0,0,0,0,1,1,0,0,1,0,0,0,0,0,0,0,0}
{0,0,0,0,0,0,0,1,1,0,1,1,1,1,0,0,0,0,0,0,0}
{0,0,0,0,0,0,1,1,0,0,1,0,0,0,1,0,0,0,0,0,0}
{0,0,0,0,0,1,1,0,1,1,1,1,0,1,1,1,0,0,0,0,0}
```

```
{0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,1,1,0,0,1,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,1,1,0,1,1,1,1,0,0,0,0,0,0,0,
 0,0,0,0,0,0,1,1,0,0,1,0,0,0,1,0,0,0,0,0,0,
 0,0,0,0,0,1,1,0,1,1,1,1,0,1,1,1,0,0,0,0,0...}
```

Figure 4. Left: Individual Lists; Right: Flattened List for Rule 30 Cellular Automaton

For each step we now derive a list of ones and zeros instead of black and white. Step 9, for instance, becomes: [0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0]. Problems such as limited variety and small list size can be easily solved by using a large number of steps and later merging or flattening all lists into one (Figure 4) according to Wolfram Demonstrations Project "Hearing Cellular Automata" [6].

MATHEMATICA enables us to produce sound by playing a list of integers when provided with a sample rate. The sample rate can be initially chosen arbitrarily in order to produce sound depending on the number of integers in the flattened list. However, there is a relationship between the music note frequency, the number of steps and the sample rate that allows us to generate the specific music note we desire. The sample rate is equal to the product of the number of steps multiplied by the music notes frequency. Therefore, after three simple steps we can generate the sound of a specific music note using the Rule 30 Cellular Automaton.

1. Find frequency of the music note required (Section 3.1).
2. Multiply the number of steps by the frequency from Step 1 and round it up to the nearest integer. The sample rate must be an integer as we are dealing with indices.
3. Play the list of integers using the sample rate from Step 2 as input for MATHEMATICA commands.

The music notes were then exported as .wav files in ascending order.

3.3 The interactive virtual musical instrument. The XNA SOUNDEFFECT class was chosen for the construction of the sound manager of the application. The sound manager creates sixteen new sound effects: seven for the notes generated from sine waves, seven for those generated using cellular automata and another two for pre-recorded messages that inform the user which of the two types of notes is in use. The sounds are then loaded into the application from the Content folder and mapped to the sound effects. The sound effects are played with the use of a sound instance, which allows multiple and simultaneous calls of the functions that play the notes.

The Goblin XNA wrapper uses a scene graph to help with the scene manipulation and rendering [7]. The scene graph consists of various nodes such as the marker node and the camera node. Goblin XNA allows the addition of nodes to the root node. Each marker is defined in the configuration file (unique ID, name, size, minimum number of points), loaded into the program as a marker node and finally added to the root node to be included in the application.

The application begins by capturing live video from a computer's webcam. It waits for ten seconds, allowing the user to place the virtual piano within the camera's range. It then begins to attempt to track all markers that have been and set in the configuration files.

There are seven equally-sized markers in a row and one more in the top right corner of the virtual piano. If one of the seven aligned markers is pressed, the application can no longer track the specific marker and plays the equivalent music note. The duration of each note is fixed, and the notes pressed will not be played again until the user lifts his or her finger off the marker. The application assumes that a marker is not tracked only when the user is attempting to obstruct it by pressing on it. The marker in the top right corner functions in the same way, but is only used to switch between two different types of music notes, those generated from sine waves and those generated from the Rule 30 automaton.

There is a Boolean variable for each marker that defines whether the task assigned to the marker can be performed or not. The application assumes that the markers will be found when initiated; therefore the initial value for the Boolean variables is 0 or “the task can be performed”.

The function used to check whether a marker is tracked and whether a music note should be played is based on the variation of the Goblin XNA embedded tutorial marker tracking algorithm that follows:

1. Set initial value for the Boolean variable to 0, meaning “the task can be performed”.
2. If the marker is tracked, set the Boolean variable to 0. If the marker is not tracked, check whether the value of the Boolean variable is 0 and proceed to Step 3.
3. If the Boolean variable is set to 0, play the note and set the variable to 1, meaning “the task cannot be performed” as it has already been performed and the marker is obstructed.
4. Go back to Step 2 until the user exits the application.

There is a small difference in the function that checks whether the switch marker has been pressed. Even though it works with the same logic, this function uses one more Boolean variable that stores the type of sound that is getting played, providing the previous algorithm with the information for playing the notes in Step 3.

4. Results and Evaluation

The measure of performance for the computer-generated music notes is based solely on the sound quality. The notes generated using sine waves were clear, as they each represent a single frequency, providing us with a straightforward way to generate pitch-perfect notes.

The music notes generated from Rule 30 were also pitch perfect but did not sound as clear, as we are ultimately playing a discrete list of integers. When the number of steps or the sample rate is small, the only sound produced is a few discrete “taps”. As these numbers increase, the taps also increase and begin to form sounds of a certain pitch. The best word to describe the output sounds is “electronic” music notes.

The virtual musical instrument was the most troublesome part of the application, as it depended on the environment. The ARTAG SDK can lose track of a marker if:

1. There are bad lighting conditions (Shadows, bright lights).
2. The angle between the marker and the camera is smaller than a certain threshold.
3. The marker is too far away.

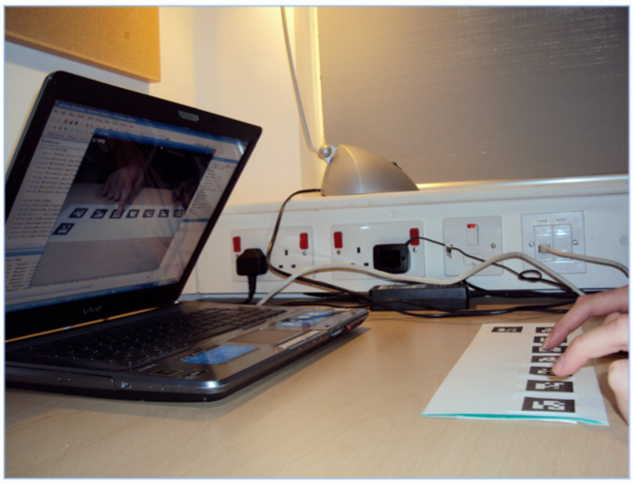


Figure 5. The Interactive Virtual Musical Instrument in action

Finally, ARTAG may give the impression that it runs in real time but there is in fact a small delay [8]. Figure 5 shows the virtual instrument in action and Figure 6 demonstrates three cases for which the application partially lost marker tracking.

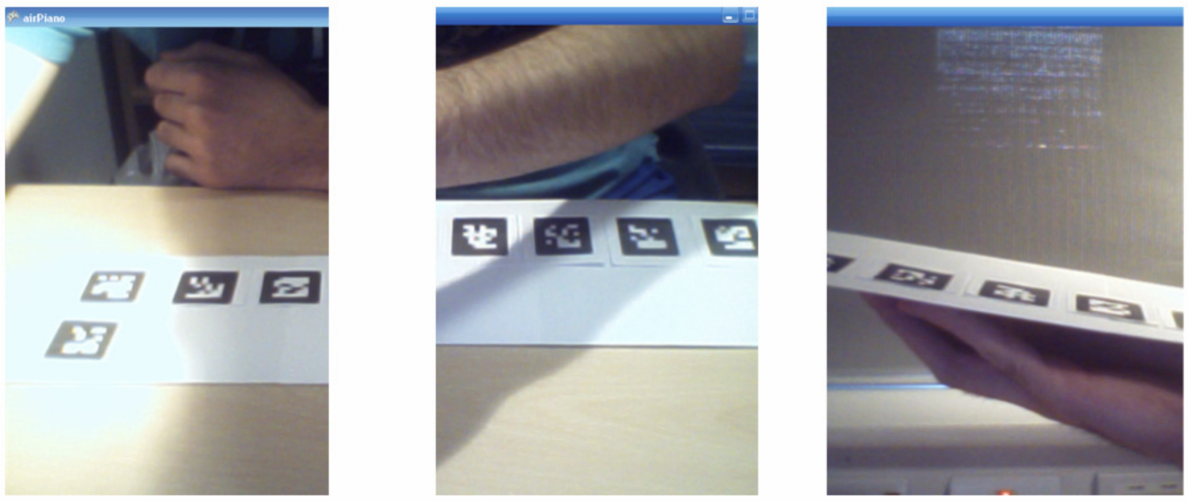


Figure 6. Loss of tracking due to Bright Light (Left), Shadow (Centre) and Large Angle (Right)

5. Conclusion and Future Work

This project combined the areas of music, mathematics and computer vision to successfully create an interactive virtual instrument. The fact that a piece of paper cannot replace a musical instrument is given; however, this project has hopefully opened up a new dimension for researchers, computer scientists and musicians.
Due to the broad gamut of features included in this dissertation, there are countless possibilities to explore in order to improve or evolve this project. Some suggestions are:

1. Visually Relatable Markers: instead of unidentifiable black and white blocks.
2. Extra Controls: such as volume and pitch shifting.
3. Visualization of the Virtual Instrument: for a more realistic look and feel.
4. Multi-Camera Environment: configuration for better marker tracking.
5. Interactive Sound Duration: based on user gesture.
6. More Complex Music Generation: variations and combinations of sine waves, multidimensional Cellular Automata.
7. More Interactive Virtual Musical Instruments.

A final note for the reader: Interactive Musical Instruments implemented with the use of marker tracking could be the start of something very special…

References

[1] Weisstein, E.W. *Cellular Automaton, From MathWorld--A Wolfram Web Resource.* [cited 2011 July 27]; Available from: http://mathworld.wolfram.com/CellularAutomaton.html

[2] Weisstein, E.W. *"Elementary Cellular Automaton" From MathWorld--A Wolfram Web Resource*. [cited 2011 July 27]; Available from: http://mathworld.wolfram.com/ElementaryCellularAutomaton.html

[3] Wolfram, S., *A new kind of science*2002, Champaign, IL: Wolfram Media. xiv, 1197 p.

[4] Nussey, A. *The Sound of Sine Waves, From The Wolfram Demonstrations Project--A Wolfram Web Resource.* [cited 2011 March 1]; Available from: http://demonstrations.wolfram.com/TheSoundOfSineWaves/.
[5] Lockhart, R. *"Hearing Cellular Automata" From The Wolfram Demonstrations Project -- A Wolfram Web Resource.* [cited 2011 March 1]; Available from: http://demonstrations.wolfram.com/TheSoundOfSineWaves/.
[6] Wolfram, S. *" On Starting a Long-Term Company" From Publications by Stephen Wolfram - Some Writings and Speeches, A lecture given at the Y Combinator Startup School held at Harvard University, October 15, 2005.* 2005 [cited 2011 March 1]; Available from: http://www.stephenwolfram.com/publications/recent/ycombinatorschool/
[7] Oda, O. and S.K. Feiner, *Goblin XNA User Manual*, 2009, Columbia University : New York. p. 5-6.
[8] Cawood, S. and M. Fiala, *Augmented Reality: A Practical Guide*2008: The Pragmatic Bookshelf (O'Reilly Media).

drakakig@tcd.ie
mmaa@cs.tcd.ie