Scanned Synthesis Then & Now – The Design and Enhancement of Csound's Scanned Opcodes

Richard Boulanger[1] and John ffitch[2]

[1] Berklee College of Music
[2] Alta Sounds
rboulanger@berklee.edu jpff@codemist.co.uk

Abstract. Scanned Synthesis was introduced at the 2000 ICMC in Berlin[1,2]. The underlying algorithm was developed by Max Mathews, expanded upon and coded into Csound by Paris Smaragdis, and, over the years, enhanced and expanded further by John ffitch. Recently, when working to clarify the wording and improve the examples in the Csound Manual, a number of questions arose such as: What are the differences between all the scanned opcodes?; How do they interact with each other?; and How do the scanned opcodes actually work under the hood? While answering these questions, and coding new examples, the authors realized that there was room for further improvement and that some powerful optional arguments could be added to support further sonic exploration. In fact, where the scanned opcodes started, and where they are today is quite an inspiring story. This paper will tell the story of the birth of `scanu` and `scans` in the mind of Max Mathews, and their current `scanu2`, `scantable`, and `scanmap` capabilities today.

Keywords: Csound, scanned, *scanu*, *scanu2*, *scans*, *scanmap*, *gen44*

1 In the Beginning and At the End

At Bell Labs, in the mid-50's, Max Mathews developed the first languages for software synthesis and signal processing - MUSIC I through V. The earliest versions of his code featured the foundational *table-lookup oscillator* which allowed for the efficient playback of any stored "contour" at any rate, pitch or duration. Given the specific settings and content of the "lookup-table", it could function as a Low Frequency or Audio-Rate Oscillator, a Sample Player, a Noise Generator, an Envelope Generator, and more! In essence, this one algorithm could emulate a wide assortment of modules on the analog synthesizer that Max was controlling with computers in his lab at Bell. In fact, combinations and interconnections of this same algorithm led to Waveshaping, FM, and Granular synthesis. In his later years, and with a paradigm shift in thinking about the elements in the oscillator "lookup-table" itself, Mathews, in a new variation of his table-lookup oscillator created a pioneering new synthesis technique — `Scanned Synthesis`, a variation of the table-lookup oscillator that, in and of itself, could behave as a spectral morphing oscillator, with filter, envelope, LFO, and more. Once again, Max Mathews had created one algorithm to rule them all!

Rather than simply seeing the samples in the table as a solution to some equation, a plot, a digital representation of an analog pressure wave, a measurement, an envelope, or a contour, Mathews' drew inspiration from Newton's mechanical model of the "string", (as masses connected together with springs), and, as in a Newton's Cradle, he connected the "samples" in his table to their neighbors. Moreover, he assigned each of the "samples" a unique mass, and applied properties of centering and damping to them. This allow each mass (sample) to respond to gravitational forces and thus to oscillate freely. Finally, he implemented ways of "exciting" this dynamic "string", by plucking it, strumming it, pushing it, and hammering it[3]

2 The First Wave of Enhancements

During a summer of study at Interval Research, Paris Smaragdis was asked if he could turn Mathews' C code into Csound opcodes. Given his deep knowledge of Csound, Smaragdis thought of ways to expand the scanned algorithm further. First, he separated the underlying algorithm into two parts, a dynamic table-surface generator that one could excite by plucking, hitting, pushing, strumming, and hammering. This became the `scanu` opcode. Second, he created the *scans* opcode, based on a variation of Mathews' traditional oscillator, to bring the complex and evolving surface waveform into the audio range. Smaragdis then identified each *scanu* surface with a unique ID number that could be read by single (or multiple) *scans* opcodes. Further, knowing Csound as he did, Smaragdis recognized that many of Csound's synthesis opcodes "also" used lookup-tables, and so, he made sure that his *scanu* ID could be passed around to any opcode that read a stored function table. In this way, Smaragdis brought new life to virtually every synthesis technique supported by Csound. With an evolving `scanu` table, one could now explore the new sonic worlds of *scannedWaveshaping*, *scannedFM*, *scannedSubtractive*, and *scannedGranular*. As a last enhancement, Smaragdis made it possible for one to excite the surface with audio input and thus turned Scanned Synthesis into a *ScannedVocoder*!

According to the original code, Smaragdis knew that he could initialize his Csound `scanu` opcode with i-rate settings, as Mathews had done, and thus, start the surface evolving from different states, on a note by note basis; but, in Csound, he also knew that it would be possible for him to add a corresponding set of "krate" inputs to many of the *scanu* parameters, and that this would allow for the initial state to be non-uniform[4] Further, Smaragdis wondered: "why only connect the masses in the evolving *scanu* table, (the samples), to their neighbors?" To overcome this limitation, he included support for a "connection matrix". This addition would allow for the 2-dimensional wave to generate 3-dimensional surfaces.

[3] All from the real-time control of his two "Radio Batons."

[4] For instance, the mass at one end of the table could be the size of a bowling ball and the mass at the other end could be the size of a pea.

Smaragdis also modified the *scans* oscillator by adding the ability to use a function that would allow one to traverse this evolving and turbulent surface along an arbitrary path, rather than just scanning from one end to another in a straight line[5] In some of his inspiring early examples, Smaragdis showed that one could have a single `scanu` followed by multiple `scans` opcodes, each slightly detuned and each with unique "trajectories"[6].

Max Mathews dream was to use his two radio-transmitting "batons" to deform the surface of a complex evolving sound wave as he moved them over his radio-receiving antenna board. This was one of his main motivations and hopes for scanned synthesis, that with his hands, he could play with, shape, and mold rich, complex and evolving timbres. Smaragdis' implementation of Mathews' scanned algorithm in Csound gave life to that dream and then some.

3 The Second Wave of Enhancements

Once the scanned opcodes began to be explored and understood, Csounders noticed that the sound itself tended to be "overly bright", a bit "raw", and sometimes pretty "nasty". To address these concerns, John ffitch added optional arguments to the `scans` opcode that support alternative interpolation modes. He also added support for larger table sizes in `scanu`. Csounders also wondered, "what programs do I use to draw my own connection matrices?" For that, Steven Yi added a wonderful *Matrix Editor* to his fantastic Csound IDE *BLUE* which allowed one to connect the massed randomly, or by using a mouse to select and draw the connection on a grid. In fact, from the study of Steven Yi's beautiful scanned compositions, one has learned that to "warm things up a bit", users should follow their `scans` opcodes with a lowpass filter, a `dcblock`, and, for ear-safety, a `limit` or `clip` opcode[7] Given the sometimes explosive nature of scanned synthesis, a variation and simplification of the technique was added to Csound in the `scantable` opcode. To "stabilize" the system, things were limited to a "circular-string" matrix and no dynamic and evolving `scanu` surface is read. Still, non-uniform tables can be used for mass, stiffness, centering, etc., and the resulting waveform can be very rich and evolve in unique ways.

4 The Third Wave of Enhancements

It occurred to some Csounders that it might be sonically interesting if one were able to read and map individual samples (nodes) from this dynamic *scanu* surface, and use it as a complex control source for other opcodes. For instance, to use the oscillations and centering of sample 64 to adjust the cutoff of a filter, and sample 21 to adjust the frequency of an oscillator, and sample 111 to control the panning of a stereo instrument. To support this request, John ffitch added the

[5] The authors like to think of this as "surfing the wave".
[6] Imagine multiple cameras filming the surface of the pool during a swimming meet
[7] Scanned surfaces can be explosive given the wide range of interconnected elements.

`scanmap` opcode which brought a whole new set of possibilities to the scanned family.

Upon further study, exploration, composition, and design, it occurred to ffitch that some of the original `scanu` parameters, such as stiffness, left and right pluck positions, and a few others were not exactly working in the way that they were explained in the manual, or in tutorials by Mathews and Boulanger[8] Rather than fix, expand, or update the original `scanu`, and risk breaking a number of interesting instruments and beautiful pieces, John ffitch added the `scanu2` opcode. Further, to simplify the design of matrix files, ffitch added a new Gen routine `GEN44`. And most recently, ffitch expanded the `scanmap` opcode to support the reading of all nodes from *scanu2* into an array, thus providing simultaneous access to multiple control signals from a single `scanu2` using a single `scanmap`!

5 The Enhanced Scanned Family of Opcodes and How they Work

Let's see how the current `scanu2` opcode has been enhanced and works.

```
scanu2 init, irate, ifndisplace, ifnmass, ifnmatrix, ifncentr,
       ifndamp, kmass, kmtrxstiff, kcentr, kdamp, ileft, iright,
       kpos, kdisplace, ain, idisp, id
```

There are really two sets of arguments here, the "igroup", and the "kgroup." The "igroup", which consist of *init, irate, ifndisplace, ifnmass, ifnmatrix, ifncentr, ifndamp, ileft, iright*, are used to set the initial state of the system. The "kgroup", which consists of *kmass, kmtrxstiff, kcentr, kdamp, kpos, kdisplace,* are essentially "push" variables – these allow one to interact with the system as it is evolving. The *ain* parameter also has a "push" role but at audio rate to support real-time audio input into the system. The *idisp* argument turns on a display of the evolving wave, and the *id* argument assigns a number to the surface being generated which other table-based opcodes use to access and read the surface.

Some new features of the *init* argument are now supported. Traditionally, one used this argument to set the initial state of the system `before` excitation[9] It was referred to as the "hammer", and although a hammer implies that you are hitting the masses or setting the system in motion, it is important to note that it `does not` set the system in motion[10].

Internally there are a number of arrays to represent the masses, and velocities, (with their position offset from neutral). On each update, the acceleration

[8] For example, increasing stiffness made the connections less stiff!

[9] This is `not` an exciter or "hammer-blow".

[10] It is better to think of this as a template to which the masses conform prior to excitation - an initial displacement shape which can be specified by an f-table whose size is equal to the number of masses in the system - typically 128

of each mass is calculated from the current velocity and the other masses connected to the mass, as given in the matrix describing the topology of the configuration[11] In the code below, we see how acceleration is computed from weight, c(centering) d(damping) kf and kd, m(mass), km, v(velocity), x1[i](last position, and x0[i](next position).

```
MYFLT kf = *p->k_f;
     for (j = 0 ; j != len ; j++) {
        MYFLT weight = p->f[i*len+j];
        if (weight!=FL(0.0))
        a += (x1[j] - x1[i]) /(weight*kf);
        }
a += -x1[i] * p->c[i] * *p->k_c - FABS(x2[i] - x1[i]) * p->d[i] * *p->k_d;
a /= p->m[i] * *p->k_m;
     /* From which we get velocity */
v[i] += dt * a;
     /* ... and future position */
x0[i] += v[i] * dt;
```

The original implementation supplied a set of these connection matrices for a string and a circular string. To allow further experimentation, we now offer an ftable-generator that reads a simple script and creates a connection matrix. This ftable-generator is described in section 5.1. For each mass the acceleration is used to change the velocity from the stiffness of the connection in a simplification of Newtonian mechanics similar to the finite difference calculation of physical modeling. The velocity then is used to update the position.

5.1 Designing the Connection Matrix

The newly added `GEN44` routine is given a file that specifies the connection matrix and translates it to the internal structure. The first line of the file should be `<MATRIX size=integer>` which is used to create a square matrix of the indicated size. This is followed by lines of two or three numbers, the first two denoting a connection from the first to the second. The third number is a weight. If the third number is omitted, it is taken as having a value of 1. The list is terminated by a line with the word `</MATRIX>` or the end of the file. To avoid confusion with other matrix formats, it is recommended to save this matrix format file using the extension ".matrxT"

5.2 An Example

In this example, the initial displacement condition (a sine shape) is used because *init* is negative. Still, one needs a velocity to set the system in motion[12]

[11] This connection matrix can be used to treat the collection of masses as if they were many shapes - one, two or three dimensional.

[12] The parameters *ifndisplace* and *kdisplace* are velocity and acceleration (the first and second derivative of displacement). The initial velocity is this displacement, as a

```
  instr scanmap_Additive
     kp[] init 128
     kv[] init 128
      giTableKP ftgen 100, 0, -128, 2, 0
      giTableKV ftgen 101, 0, -128, 2, 0
     scanu2 -1, .1, 6, 2, 3, 4, 5, 2, 9, .01, .01, .1,  \
               .9, 0, 0, 0, 1, 2
     kp,kv scanmap 2, 1000, 2
      copya2ftab kp, giTableKP
      copya2ftab kv, giTableKV
     a1 oscil ampdbfs(p4)*kv[8], cpsmidinn(p5)+kp[8]
     a2 oscil ampdbfs(p4)*kv[13], cpsmidinn(p5)+kp[13]
     a3 oscil ampdbfs(p4)*kv[55], cpsmidinn(p5)+kp[55]
     a4 oscil ampdbfs(p4)*kv[89], cpsmidinn(p5)+kp[89]
      outs a1+a4, a2+a3
  endin
    f1 0 128 10 1 ; initial displacement condition (sine shape)
    f2 0 128 -7 1 128 1 ; uniform masses
    f3 0 0 -44 "string_with_extras-16.matrxT"
    f4 0 128 -7 .001 128 1 ; ramped centering
    f5 0 128 -7 .1 128 1 ; ramped damping
    f6 0 128 -7 .01 128 .01 ; uniform initial velocity
    f7 0 128 -5 .001 128 128  ; scans trajectory
i "scanmap_Additive" 0  8  -1  72}
```

6 Conclusion

Scanned Synthesis was the final gift to Csounders from "the father of computer music", Max Mathews. It offers sound designers a wide range of expressive and dynamic timbral possibilities. In Csound, developers like Paris Smaragdis and John ffitch expanded the capabilities in ways that Max Mathews had never imagined, but would have made him very happy. They opened up new possibilities in which this dynamically evolving surface can take many forms, be traversed along many paths, be excited by any function, and re-energized by audio input. Moreover, now, it's many nodes can be mapped to any k-rate parameter, and integrated into many of Csound's table-driven and table-based synthesis algorithms - thereby bringing new "life" to classic synthesis techniques. Using scanned synthesis, a number of truly beautiful instruments have already been designed, and some great pieces have been written. More are sure to follow, especially as understanding and appreciation continue to grow, and Csounders begin to explore and discover the new ways to work with these enhanced opcodes.

---

measure of velocity and acceleration, and is calculated from the stiffness matrix - a connection matrix, both assigning and weighting the connections. Note that positive entries in the stiffness matrix increase acceleration.

References

1. Bill Verplank, Max V. Mathews, and Robert Shaw. Scanned Synthesis. In Ioannis Zannos, editor, *ICMC2000*, pages 368–371. ICMA, August 2000.
2. Richard Boulanger, Paris Smaragdis, and John ffitch. Scanned Synthesis: An Introduction and Demonstration of a New Synthesis and Signal Processing Technique. In Ioannis Zannos, editor, *ICMC2000*, pages 372–375. ICMA, August 2000.