Bithoven

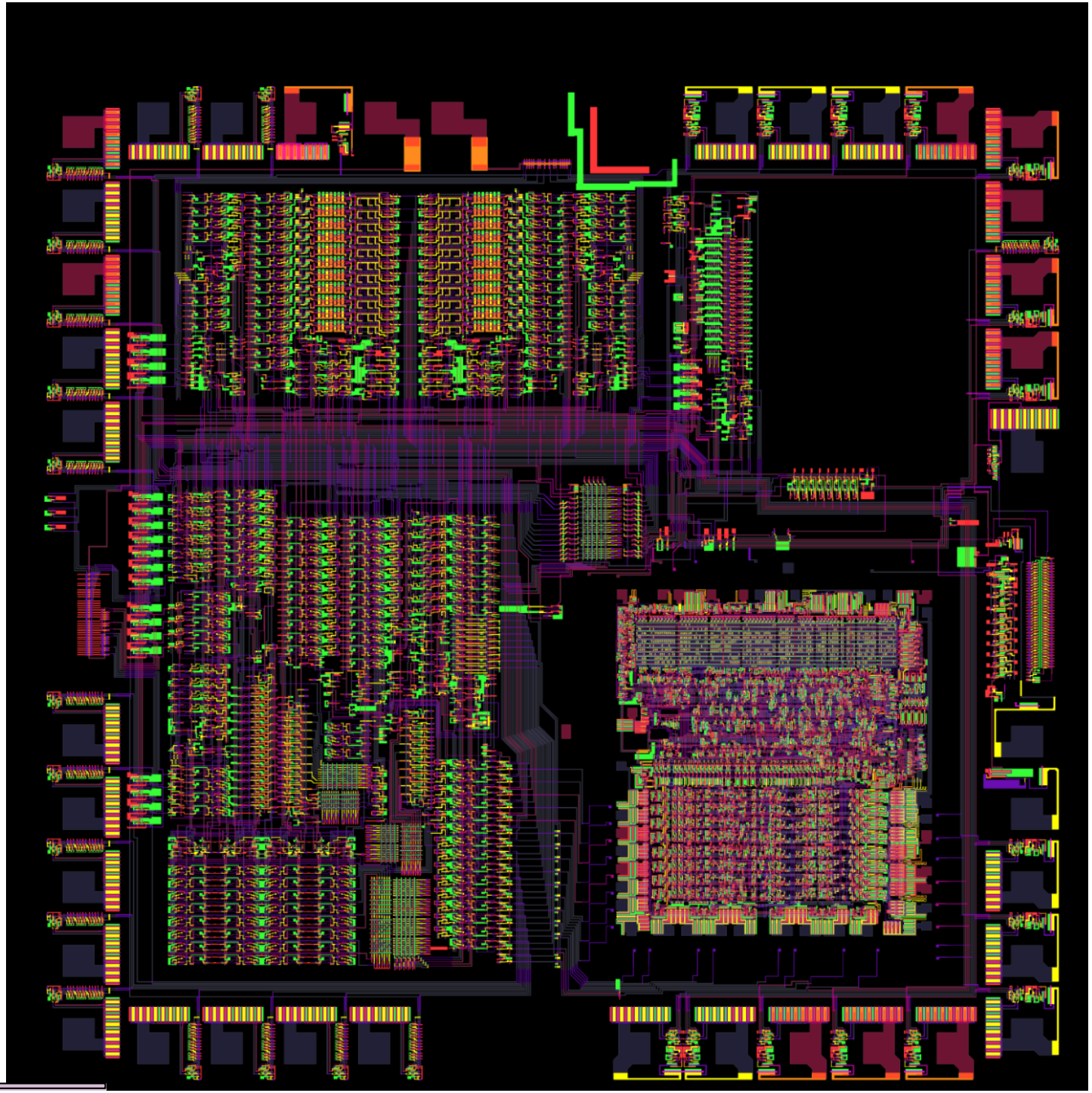
Gödel Encoding of Chamber Music and Functional 8-Bit Audio Synthesis

Jay McCarthy
UMass Lowell & PLT


Nintendo
ENTERTAINMENT SYSTEM
POWER
RESET
Nintendo
SELECT START
B
A


FAMILY COMPUTER™
Nintendo®
I
II
SELECT
START
B
A
VOLUME
MIC


U6
CPU
RP2A07A
20M 3G



Outline

-  Waves

Outline

- Waves
- Notes

Outline

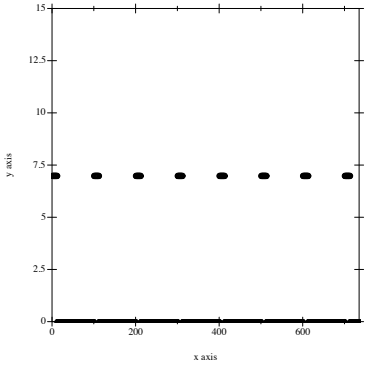
- Waves
- Notes
- Scales and Chords

Outline

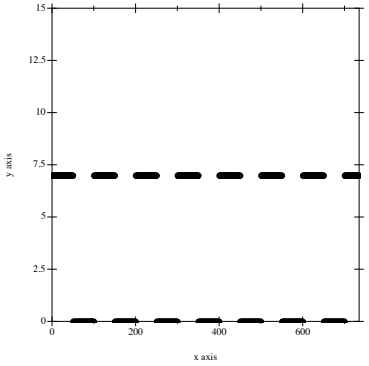
- Waves
- Notes
- Scales and Chords
- Music

Synthesis

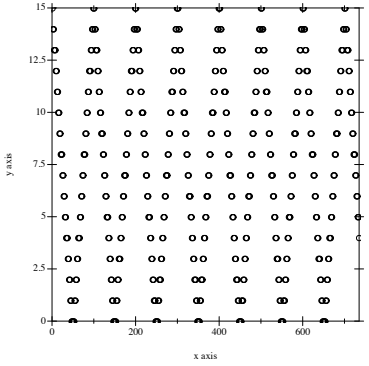
Pulse 1



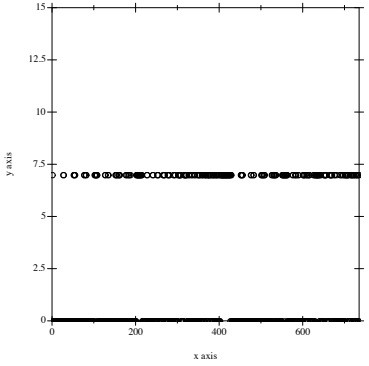
Pulse 2



Triangle



Noise

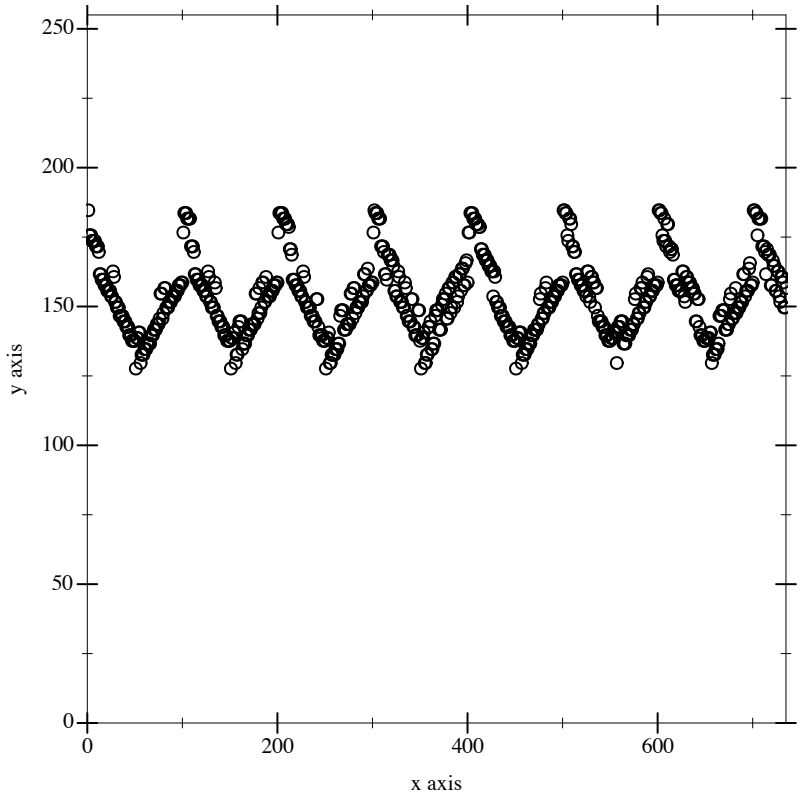


```
(define DUTY (vector 0.125 0.25 0.5 0.75))
(define (pulse-wave n period volume %)
  (define freq
    (pulse-period->freq period))
  (define duty-cycle
    (vector-ref DUTY n))
  (define next-% (cycle%-step % freq))
  (define out
    (if (< next-% duty-cycle)
      volume
      0))
  (values out next-%))
```

```
(define CPU-FREQ-MHz 1.789773)
(define CPU-FREQ-Hz
  (* CPU-FREQ-MHz 1000.0 1000.0))
(define (pulse-period->freq period)
  (/ CPU-FREQ-Hz (* 16.0 (+ 1.0 period))))
(define (cycle%-step % freq)
  (define %step (/ freq 44100.0))
  (define next% (+ % %step))
  (- next% (floor next%)))
```

```
(struct wave:pulse (duty period volume))
(struct wave:triangle (on? period))
(struct wave:noise (short? period volume))
(struct wave:dmc (bs offset))
(struct synth-frame (p1 p2 t n d))
```

```
(struct synth-frame (p1))
(define (synth sf)
  (match-define (synth-frame p1) sf)
  (match-define (wave:pulse d p v) p1)
  (define sample-count 735)
  (define samples (make-bytes sample-count))
  (for/fold ([p1-% 0.0])
            ([i (in-range sample-count)])
    (define-values (p1 new-p1-%)
      (pulse-wave d p v p1-%))
    (bytes-set! samples i p1-d)
    new-p1-%)
  samples)
```


250
200
150
100
50
0
y axis
0
200
400
600
x axis

Instruments

```
(define (pulse-freq->period freq)
  (define pre (/ CPU-FREQ-Hz (* 16.0 freq)))
  (round (- pre 1.0)))
(define pulse-tone->period
  (compose1 pulse-freq->period tone->freq))
```

```
(define (i:pulse #:duty ds
                 #:period ps
                 #:volume vs)
  (λ (frames tone)
    (define d* (stage-spec ds frames))
    (define p* (stage-spec ps frames))
    (define v* (stage-spec vs frames))
    (define base-per
      (pulse-tone->period tone))
    (for/list ([f (in-range frames)])
      (define duty (eval-spec d* f))
      (define per
        (fx+ base-per (eval-spec p* f)))
      (define volume (eval-spec v* f))
      (wave:pulse duty per volume))))
```

```
(define (i:pulse:basic duty)
  (i:pulse
   #:duty (spec:constant duty)
   #:period (spec:constant 0)
   #:volume (spec:constant 7)))
```

```
(define (i:pulse:tremolo freq duty)
  (i:pulse/spec
   #:duty (spec:constant duty)
   #:period (spec:constant 0)
   #:volume (spec:% (spec:modulate freq 7 4))))
```

```
(define (i:pulse:linear duty)
  (i:pulse/spec
   #:duty (spec:constant duty)
   #:period (spec:constant 0)
   #:volume (spec:% (spec:linear 7 0))))
```

```
(define (i:pulse:plucky duty)
  (i:pulse/spec
   #:duty (spec:constant duty)
   #:period (spec:constant 0)
   #:volume
   (spec:adsr 'release
              4 (spec:constant 14)
              4 (spec:linear 14 7)
              4 (spec:constant 7)
              4 (spec:linear 7 0))))
```

```
(define (i:pulse:natural duty)
  (i:pulse/spec
   #:duty (spec:constant duty)
   #:period (spec:constant 0)
   #:volume
   (spec:adsr 'sustain
              4 (spec:constant 14)
              4 (spec:linear 14 7)
              4 (spec:constant 7)
              4 (spec:linear 7 0))))
```

```
(define i:drum:hihat
  (i:noise
   #:mode
   (spec:constant #f)
   #:period
   (spec:constant 12)
   #:volume
   (spec:adsr
    'release
    1 (spec:constant 4)
    2 (spec:constant 3)
    4 (spec:constant 2)
    4 (spec:constant 0))))
```

```
(define i:drum:bass
  (i:noise
   #:mode
   (spec:constant #f)
   #:period
   (spec:constant 9)
   #:volume
   (spec:adsr
    'release
    1 (spec:constant 10)
    2 (spec:constant 7)
    4 (spec:linear 4 2)
    4 (spec:constant 0))))
```

```
(define i:drum:snare
  (i:noise
   #:mode
   (spec:constant #f)
   #:period
   (spec:constant 7)
   #:volume
   (spec:adsr
    'release
    1 (spec:constant 11)
    4 (spec:linear 11 6)
    8 (spec:linear 6 2)
    4 (spec:constant 0))))
```

Music Theory

Music Theory

- Notes

Music Theory

- Notes
- Tempo

Music Theory

- Notes
- Tempo
- Time Signature

Music Theory

- Notes
- Tempo
- Time Signature
- Scale

Music Theory

- Notes
- Tempo
- Time Signature
- Scale
  - Detached Scale

Music Theory

- Notes
- Tempo
- Time Signature
- Scale
  - Detached Scale
  - Fixed Scale

Music Theory

- Notes
- Tempo
- Time Signature
- Scale
  - Detached Scale
  - Fixed Scale
  - Abstract Scale

Music Theory

- Notes
- Tempo
- Time Signature
- Scale
  - Detached Scale
  - Fixed Scale
  - Abstract Scale
  - Abstract Tone

Music Theory

- Notes
- Tempo
- Time Signature
- Scale
  - Detached Scale
  - Fixed Scale
  - Abstract Scale
  - Abstract Tone
- Tracker

Music Theory

- Notes
- Tempo
- Time Signature
- Scale
  - Detached Scale
  - Fixed Scale
  - Abstract Scale
  - Abstract Tone
- Tracker
- Chord 

```
(require data/enumerate)
```

```
(require data/enumerate)
```

```
(to-nat string/e "Bithoven")
```

```
=> 29446701124374494676409535373198388243249180
```

```
(require data/enumerate)
```

```
(to-nat string/e "Bithoven")
```

```
=> 29446701124374494676409535373198388243249180
```

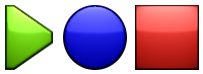
```
(from-nat string/e 42)
```

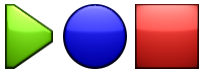
```
=> "Q"
```

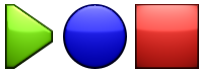
```
(define (part/e ts ap cp measures len)
  (define cp-s (progression-seq cp))
  (define pulses
    (* len measures (accent-pattern-pulses-per-measure ap)))
  (define cp/e
    (chord-pulses/e pulses (length cp-s)))
  (do/e cps <- cp/e
        cp  <~ cps
        (rhythm/e ts (* cp (accent-pattern-notes-per-pulse ap)))))
(define bithoven/e
  (vector/e
   (do/e           ts <- time-sig/e
                   ap <- (accent-pattern/e ts)
         (cons f cp) <- (cons/e form/e chord-progression/e)
                    p <~ (form-part-lens f)
         (part/e ts ap cp (length (progression-seq cp)) (cdr p)))
   bass-notes/e))
```

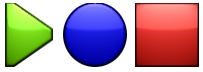
```
(define (make-nestration/e c)
  (match-define (vector ts ap pattern parts) c)
  (vector/e
   tone-names/e scales/e tempo/e
   pulse1/e pulse2/e triangle/e drums/e
   mhtb/e
   (fin/e 2 3) (fin/e 1 2) (fin/e 0 2)
   (do/e _ <~ parts
         (drum-measure/e ts ap))
   (do/e ms <~ parts
         rest-n <- rest-n/e
         (if rest-n
           (listof-n/e
            (below/e rest-n)
            (add1 (ceiling (/ (length (append* ms)) rest-n))))
           (single/e '())))))
```

Synthesizer	415
Instrument DSL	254
Music Theory	301
Tracker	141
Bithoven	442
NEStration	528
TOTAL	2226

Classic 

No Drums 

Happy 

Sad 

All