fractaloids

Claude Heiland-Allen

2012

Contents

1 .gitignore

```
-
pdgem/dynamo.pd.*.png
pdgem/dynamo.pd.*.tif
pdgem/dynamo.pd.*.wav
pdgem/dynamo.pd.*.cue
```

2 pdgem/audio.pd

```
#N canvas 326 57 658 592 10;
#X obj 289 28 table \$0-scale 5;
#X obj 87 7 loadbang;
```

```
#N canvas 0 0 450 300 \$0-init 0;
#X obj 119 13 inlet;
#X obj 56 65 t b b;
#N canvas 0 0 457 460 \$0-init-scale 0;
#X obj 19 19 inlet;
#X obj 19 40 t b b b b b;
#X obj 294 118 random 12;
#X obj 294 272 + 40;
#X obj 220 118 random 11;
#X obj 220 139 t f f;
#X obj 220 160 +;
#X obj 247 161 >=;
#X obj 141 116 random 10;
#X obj 141 137 t f f;
#X obj 141 158 +;
#X obj 141 179 t f f;
#X obj 141 200 +;
#X obj 168 158 >=;
#X obj 168 201 >=;
#X obj 71 137 t f f;
#X obj 71 158 +;
#X obj 71 179 t f f;
#X obj 71 200 +;
#X obj 98 158 >=;
#X obj 98 201 >=;
#X obj 71 116 random 9;
#X obj 71 222 t f f;
#X obj 71 243 +;
#X obj 98 244 >=;
#X obj 19 154 f;
#X obj 19 175 + 12;
#X obj 220 272 + 40;
#X obj 141 272 + 40;
#X obj 71 272 + 40;
#X obj 20 272 + 40;
#X obj 20 318 pack f f f f f;
#X obj 20 339 list prepend 0;
#X obj 20 360 s \$0-scale;
#X connect 0 0 1 0;
#X connect 1 0 25 0;
#X connect 1 1 21 0;
#X connect 1 2 8 0;
#X connect 1 3 4 0;
#X connect 1 4 2 0;
#X connect 2 0 3 0;
#X connect 2 0 7 1;
#X connect 2 0 13 1;
#X connect 2 0 19 1;
#X connect 2 0 25 1;
#X connect 3 0 31 4;
#X connect 4 0 5 0;
#X connect 5 0 6 0;
#X connect 5 1 7 0;
#X connect 6 0 14 1;
#X connect 6 0 20 1;
#X connect 6 0 27 0;
#X connect 7 0 6 1;
```

```
#X connect 8 0 9 0;
#X connect 9 0 10 0;
#X connect 9 1 13 0;
#X connect 10 0 11 0;
#X connect 11 0 12 0;
#X connect 11 1 14 0;
#X connect 12 0 24 1;
#X connect 12 0 28 0;
#X connect 13 0 10 1;
#X connect 14 0 12 1;
#X connect 15 0 16 0;
#X connect 15 1 19 0;
#X connect 16 0 17 0;
#X connect 17 0 18 0;
#X connect 17 1 20 0;
#X connect 18 0 22 0;
#X connect 19 0 16 1;
#X connect 20 0 18 1;
#X connect 21 0 15 0;
#X connect 22 0 23 0;
#X connect 22 1 24 0;
#X connect 23 0 29 0;
#X connect 24 0 23 1;
#X connect 25 0 26 0;
#X connect 26 0 30 0;
#X connect 27 0 31 3;
#X connect 28 0 31 2;
#X connect 29 0 31 1;
#X connect 30 0 31 0;
#X connect 31 0 32 0;
#X connect 32 0 33 0;
#X restore 103 66 pd \$0-init-scale;
#X obj 56 92 t b b;
#N canvas 0 0 450 300 \$0-init-bassline 0;
#X obj 16 18 inlet;
#X obj 16 39 t b b;
#X obj 16 60 f 32;
#X obj 16 81 until;
#X obj 16 102 f 0;
#X obj 56 102 + 1;
#X obj 56 123 mod 32;
#X obj 16 144 t b f;
#X obj 16 275 tabwrite \$0-bassline;
#X obj 16 165 random 100;
#X obj 16 186 >;
#X obj 16 228 random 5;
#X obj 35 250 f -1;
#X obj 16 207 sel 0 1;
#X obj 103 102 random 60;
#X obj 103 123 + 5;
#X connect 0 0 1 0;
#X connect 1 0 2 0;
#X connect 1 1 14 0;
#X connect 2 0 3 0;
#X connect 3 0 4 0;
#X connect 4 0 5 0;
#X connect 4 0 7 0;
```

```
#X connect 5 0 6 0;
#X connect 6 0 4 1;
#X connect 7 0 9 0;
#X connect 7 1 8 1;
#X connect 9 0 10 0;
#X connect 10 0 13 0;
#X connect 11 0 8 0;
#X connect 12 0 8 0;
#X connect 13 0 11 0;
#X connect 13 1 12 0;
#X connect 14 0 15 0;
#X connect 15 0 10 1;
#X restore 104 93 pd \$0-init-bassline;
#X obj 56 113 t b b;
#N canvas 0 0 450 300 \$0-init-globals 0;
#X obj 10 11 inlet;
#X obj 10 32 t b b;
#X obj 86 126 swap 15000;
#X obj 86 147 /;
#X obj 86 168 v \$0-tempo;
#X obj 13 124 v \$0-swing;
#X obj 13 103 / 400;
#X obj 87 72 / 5;
#X obj 86 105 + 100;
#X obj 86 40 random 300;
#X obj 13 82 random 150;
#X connect 0 0 1 0;
#X connect 1 0 10 0;
#X connect 1 1 9 0;
#X connect 2 0 3 0;
#X connect 2 1 3 1;
#X connect 3 0 4 0;
#X connect 6 0 5 0;
#X connect 7 0 8 0;
#X connect 8 0 2 0;
#X connect 9 0 7 0;
#X connect 10 0 6 0;
#X restore 106 143 pd \$0-init-globals;
#X obj 119 281 outlet;
#X obj 56 144 t b b;
#N canvas 0 0 450 300 \$0-init-accent 0;
#X obj 25 18 inlet;
#X obj 25 39 t b b;
#X obj 25 60 f 32;
#X obj 25 81 until;
#X obj 25 102 f 0;
#X obj 57 101 + 1;
#X obj 57 122 mod 32;
#X obj 24 144 t b f;
#X obj 24 165 random 8;
#X obj 24 186 tabwrite \$0-accent;
#X connect 0 0 1 0;
#X connect 1 0 2 0;
#X connect 2 0 3 0;
#X connect 3 0 4 0;
#X connect 4 0 5 0;
#X connect 4 0 7 0;
```

```
#X connect 5 0 6 0;
#X connect 6 0 4 1;
#X connect 7 0 8 0;
#X connect 7 1 9 1;
#X connect 8 0 9 0;
#X restore 103 115 pd \$0-init-accent;
#X obj 56 165 t b b;
#N canvas 0 0 450 300 \$0-init-dubpad 0;
#X obj 22 14 inlet;
#X obj 22 35 t b b;
#X obj 22 56 random 16;
#X obj 22 77 + 1;
#X obj 22 98 v \$0-dubpad-l-del;
#X obj 139 53 random 16;
#X obj 139 74 + 1;
#X obj 139 95 v \$0-dubpad-r-del;
#X connect 0 0 1 0;
#X connect 1 0 2 0;
#X connect 1 1 5 0;
#X connect 2 0 3 0;
#X connect 3 0 4 0;
#X connect 5 0 6 0;
#X connect 6 0 7 0;
#X restore 106 163 pd \$0-init-dubpad;
#N canvas 0 257 450 300 \$0-init-triline 0;
#X obj 12 16 inlet;
#X obj 12 37 t b b;
#X obj 12 58 f 32;
#X obj 12 79 until;
#X obj 12 100 f 0;
#X obj 52 100 + 1;
#X obj 52 121 mod 32;
#X obj 12 142 t b f;
#X obj 12 163 random 100;
#X obj 12 184 >;
#X obj 12 226 random 5;
#X obj 31 248 f -1;
#X obj 12 205 sel 0 1;
#X obj 12 273 tabwrite \$0-triline;
#X obj 99 121 + 2;
#X obj 100 97 random 15;
#X connect 0 0 1 0;
#X connect 1 0 2 0;
#X connect 1 1 15 0;
#X connect 2 0 3 0;
#X connect 3 0 4 0;
#X connect 4 0 5 0;
#X connect 4 0 7 0;
#X connect 5 0 6 0;
#X connect 6 0 4 1;
#X connect 7 0 8 0;
#X connect 7 1 13 1;
#X connect 8 0 9 0;
#X connect 9 0 12 0;
#X connect 10 0 13 0;
#X connect 11 0 13 0;
#X connect 12 0 10 0;
```

```
#X connect 12 1 11 0;
#X connect 14 0 9 1;
#X connect 15 0 14 0;
#X restore 108 189 pd \$0-init-triline;
#X obj 56 195 t b b;
#X obj 56 216 t b b;
#N canvas 0 0 450 300 \$0-init-syncline 0;
#X obj 18 17 inlet;
#X obj 19 39 t b b;
#X obj 19 60 f 32;
#X obj 19 81 until;
#X obj 19 102 f 0;
#X obj 59 102 + 1;
#X obj 59 123 mod 32;
#X obj 19 144 t b f;
#X obj 19 165 random 100;
#X obj 19 186 >;
#X obj 38 250 f -1;
#X obj 19 207 sel 0 1;
#X obj 19 275 tabwrite \$0-syncline;
#X obj 19 228 random 50;
#X obj 106 123 + 2;
#X obj 106 102 random 20;
#X connect 0 0 1 0;
#X connect 1 0 2 0;
#X connect 1 1 15 0;
#X connect 2 0 3 0;
#X connect 3 0 4 0;
#X connect 4 0 5 0;
#X connect 4 0 7 0;
#X connect 5 0 6 0;
#X connect 6 0 4 1;
#X connect 7 0 8 0;
#X connect 7 1 12 1;
#X connect 8 0 9 0;
#X connect 9 0 11 0;
#X connect 10 0 12 0;
#X connect 11 0 13 0;
#X connect 11 1 10 0;
#X connect 13 0 12 0;
#X connect 14 0 9 1;
#X connect 15 0 14 0;
#X restore 106 215 pd \$0-init-syncline;
#X obj 58 13 inlet;
#X obj 52 280 outlet;
#X obj 119 40 t b b b;
#N canvas 0 0 450 300 \$0-init-samples 0;
#X obj 14 20 inlet;
#X obj 14 41 t b b;
#X obj 14 72 symbol \$0-breakbeat-l;
#X obj 34 92 symbol \$0-breakbeat-r;
#X obj 13 118 pack s s;
#X msg 13 139 read -resize breakbeat.wav \$1 \$2;
#X obj 13 160 soundfiler;
#X obj 13 181 v \$0-breakbeat-len;
#X connect 0 0 1 0;
#X connect 1 0 2 0;
```

```
#X connect 1 1 3 0;
#X connect 2 0 4 0;
#X connect 3 0 4 1;
#X connect 4 0 5 0;
#X connect 5 0 6 0;
#X connect 6 0 7 0;
#X restore 247 64 pd \$0-init-samples;
#X obj 56 245 t b b;
#N canvas 18 274 450 300 \$0-init-flangeline 0;
#X obj 16 18 inlet;
#X obj 16 39 t b b;
#X obj 16 60 f 32;
#X obj 16 81 until;
#X obj 16 102 f 0;
#X obj 56 102 + 1;
#X obj 56 123 mod 32;
#X obj 16 144 t b f;
#X obj 16 165 random 100;
#X obj 16 186 >;
#X obj 16 228 random 5;
#X obj 35 250 f -1;
#X obj 16 207 sel 0 1;
#X obj 16 275 tabwrite \$0-flangeline;
#X obj 103 123 + 50;
#X obj 103 102 random 50;
#X connect 0 0 1 0;
#X connect 1 0 2 0;
#X connect 1 1 15 0;
#X connect 2 0 3 0;
#X connect 3 0 4 0;
#X connect 4 0 5 0;
#X connect 4 0 7 0;
#X connect 5 0 6 0;
#X connect 6 0 4 1;
#X connect 7 0 8 0;
#X connect 7 1 13 1;
#X connect 8 0 9 0;
#X connect 9 0 12 0;
#X connect 10 0 13 0;
#X connect 11 0 13 0;
#X connect 12 0 10 0;
#X connect 12 1 11 0;
#X connect 14 0 9 1;
#X connect 15 0 14 0;
#X restore 106 241 pd \$0-init-flangeline;
#X obj 360 284 outlet;
#X connect 0 0 18 0;
#X connect 1 0 3 0;
#X connect 1 1 2 0;
#X connect 3 0 5 0;
#X connect 3 1 4 0;
#X connect 5 0 8 0;
#X connect 5 1 9 0;
#X connect 8 0 10 0;
#X connect 8 1 6 0;
#X connect 10 0 13 0;
#X connect 10 1 11 0;
```

```
#X connect 13 0 14 0;
#X connect 13 1 12 0;
#X connect 14 0 20 0;
#X connect 14 1 15 0;
#X connect 16 0 1 0;
#X connect 18 0 7 0;
#X connect 18 1 1 0;
#X connect 18 2 19 0;
#X connect 20 0 17 0;
#X connect 20 1 21 0;
#X connect 20 1 22 0;
#X restore 12 45 pd \$0-init;
#X obj 51 22 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
-1;
#X obj 288 50 table \$0-bassline 32;
#X obj 289 98 value \$0-tempo;
#N canvas 0 0 450 300 \$0-clock 0;
#X obj 162 8 inlet;
#X obj 22 92 metro;
#X obj 21 37 t b b b;
#X obj 50 63 v \$0-tempo;
#X obj 22 119 f 0;
#X obj 66 92 f 0;
#X obj 21 266 outlet;
#X obj 21 244 f 0;
#X obj 53 244 + 1;
#X obj 54 119 ==;
#X obj 22 141 sel 0 1;
#X obj 31 221 delay;
#X obj 41 160 t b b;
#X obj 40 181 v \$0-swing;
#X obj 40 202 *;
#X obj 67 201 v \$0-tempo;
#X obj 34 6 inlet;
#X connect 0 0 2 0;
#X connect 1 0 4 0;
#X connect 2 0 1 0;
#X connect 2 1 3 0;
#X connect 2 2 5 0;
#X connect 3 0 1 1;
#X connect 4 0 9 0;
#X connect 4 0 10 0;
#X connect 5 0 4 1;
#X connect 5 0 7 1;
#X connect 7 0 8 0;
#X connect 7 0 6 0;
#X connect 8 0 7 1;
#X connect 9 0 4 1;
#X connect 10 0 7 0;
#X connect 10 1 12 0;
#X connect 11 0 7 0;
#X connect 12 0 13 0;
#X connect 12 1 15 0;
#X connect 13 0 14 0;
#X connect 14 0 11 0;
#X connect 15 0 14 1;
#X connect 16 0 3 0;
```

```
#X restore 54 113 pd \$0-clock;
#N canvas 0 0 450 681 \$0-bassist 0;
#X obj 59 2 inlet;
#X obj 59 101 tabread \$0-bassline;
#X obj 59 122 sel -1;
#X obj 56 150 tabread \$0-scale;
#X obj 56 171 - 12;
#X obj 56 268 pack f f;
#X obj 101 208 v \$0-tempo;
#X obj 56 189 t f b b;
#X obj 69 226 random;
#X obj 69 247 / 4;
#X obj 56 289 vline~;
#X obj 56 310 mtof~;
#X obj 56 331 phasor~;
#X obj 51 473 -~ 0.25;
#X obj 51 494 cos~;
#X obj 3 642 outlet~;
#X obj 78 381 cos~;
#X obj 78 360 *~ 2;
#X obj 182 20 inlet;
#X obj 262 19 inlet;
#X obj 364 17 inlet;
#X obj 182 41 t b b;
#X obj 209 66 v \$0-tempo;
#X obj 183 90 random 16;
#X obj 183 111 + 1;
#X obj 183 132 *;
#X obj 183 195 vline~;
#X obj 183 216 *~;
#X obj 183 237 lop~ 25;
#X obj 128 381 cos~;
#X obj 262 41 t b b;
#X obj 289 66 v \$0-tempo;
#X obj 263 90 random 16;
#X obj 263 111 + 1;
#X obj 263 132 *;
#X obj 263 195 vline~;
#X obj 263 216 *~;
#X obj 263 237 lop~ 25;
#X obj 208 381 cos~;
#X obj 208 360 *~ 5;
#X obj 362 41 t b b;
#X obj 389 66 v \$0-tempo;
#X obj 363 90 random 16;
#X obj 363 111 + 1;
#X obj 363 132 *;
#X obj 363 195 vline~;
#X obj 363 216 *~;
#X obj 363 237 lop~ 25;
#X obj 308 381 cos~;
#X obj 308 402 *~;
#X obj 308 360 *~ 6;
#X msg 363 174 2 \, 0 \$1;
#X obj 308 423 expr~ 0.2 * tanh($v1);
#X obj 208 422 *~;
#X obj 138 442 *~;
```

```
#X msg 263 174 2 \, 0 \$1;
#X msg 183 174 2 \, 0 \$1;
#X obj 78 402 *~ 0.3;
#X obj 363 153 / 4;
#X obj 263 153 / 4;
#X obj 183 153 / 4;
#X obj 208 443 expr~ 0.2 * tanh($v1);
#X obj 138 463 expr~ 0.2 * tanh($v1);
#X obj 128 360 *~ 7;
#X obj 3 360 -~ 0.25;
#X obj 3 381 cos~;
#X obj 2 581 expr~ tanh($v1);
#X obj 107 612 expr~ tanh($v1);
#X obj 106 543 *~;
#X obj 59 23 t f b;
#X obj 284 494 v \$0-tempo;
#X msg 284 553 2 \, 0 \$1;
#X obj 284 577 vline~;
#X obj 284 598 *~;
#X obj 284 619 lop~ 25;
#X obj 106 568 vcf~ 50;
#X obj 3 402 *~ 2;
#X obj 146 517 *~ 8;
#X obj 284 472 t b b;
#X obj 284 524 *;
#X obj 314 513 random 4;
#X obj 314 534 + 1;
#X obj 59 45 mod 64;
#X obj 101 82 - 32;
#X obj 58 64 moses 47.5;
#X obj 106 589 *~ 15;
#X obj 58 82 mod 16;
#X obj 4 7 inlet;
#X obj 199 495 route bassfx;
#X obj 199 516 vline~;
#X obj 109 639 *~;
#X obj 109 660 outlet~;
#X connect 0 0 69 0;
#X connect 1 0 2 0;
#X connect 2 1 3 0;
#X connect 2 1 78 0;
#X connect 3 0 4 0;
#X connect 4 0 7 0;
#X connect 5 0 10 0;
#X connect 6 0 8 1;
#X connect 7 0 5 0;
#X connect 7 1 8 0;
#X connect 7 2 6 0;
#X connect 8 0 9 0;
#X connect 9 0 5 1;
#X connect 10 0 11 0;
#X connect 11 0 12 0;
#X connect 11 0 77 0;
#X connect 12 0 13 0;
#X connect 12 0 17 0;
#X connect 12 0 39 0;
#X connect 12 0 50 0;
```

```
#X connect 12 0 63 0;
#X connect 12 0 64 0;
#X connect 13 0 14 0;
#X connect 14 0 68 0;
#X connect 14 0 66 0;
#X connect 16 0 57 0;
#X connect 17 0 16 0;
#X connect 18 0 21 0;
#X connect 19 0 30 0;
#X connect 20 0 40 0;
#X connect 21 0 23 0;
#X connect 21 1 22 0;
#X connect 22 0 25 1;
#X connect 23 0 24 0;
#X connect 24 0 25 0;
#X connect 25 0 60 0;
#X connect 26 0 27 0;
#X connect 26 0 27 1;
#X connect 27 0 28 0;
#X connect 28 0 54 1;
#X connect 29 0 54 0;
#X connect 30 0 32 0;
#X connect 30 1 31 0;
#X connect 31 0 34 1;
#X connect 32 0 33 0;
#X connect 33 0 34 0;
#X connect 34 0 59 0;
#X connect 35 0 36 0;
#X connect 35 0 36 1;
#X connect 36 0 37 0;
#X connect 37 0 53 1;
#X connect 38 0 53 0;
#X connect 39 0 38 0;
#X connect 40 0 42 0;
#X connect 40 1 41 0;
#X connect 41 0 44 1;
#X connect 42 0 43 0;
#X connect 43 0 44 0;
#X connect 44 0 58 0;
#X connect 45 0 46 0;
#X connect 45 0 46 1;
#X connect 46 0 47 0;
#X connect 47 0 49 1;
#X connect 48 0 49 0;
#X connect 49 0 52 0;
#X connect 50 0 48 0;
#X connect 51 0 45 0;
#X connect 52 0 13 0;
#X connect 53 0 61 0;
#X connect 54 0 62 0;
#X connect 55 0 35 0;
#X connect 56 0 26 0;
#X connect 57 0 13 0;
#X connect 58 0 51 0;
#X connect 59 0 55 0;
#X connect 60 0 56 0;
#X connect 61 0 13 0;
```

```
#X connect 62 0 13 0;
#X connect 63 0 29 0;
#X connect 64 0 65 0;
#X connect 65 0 76 0;
#X connect 66 0 15 0;
#X connect 66 0 90 0;
#X connect 67 0 66 0;
#X connect 68 0 75 0;
#X connect 69 0 82 0;
#X connect 70 0 79 0;
#X connect 71 0 72 0;
#X connect 72 0 73 0;
#X connect 72 0 73 1;
#X connect 73 0 74 0;
#X connect 74 0 68 1;
#X connect 75 0 85 0;
#X connect 76 0 66 0;
#X connect 77 0 75 1;
#X connect 78 0 70 0;
#X connect 78 1 80 0;
#X connect 79 0 71 0;
#X connect 80 0 81 0;
#X connect 81 0 79 1;
#X connect 82 0 84 0;
#X connect 83 0 1 0;
#X connect 84 0 86 0;
#X connect 84 1 83 0;
#X connect 85 0 67 0;
#X connect 86 0 1 0;
#X connect 87 0 88 0;
#X connect 88 0 89 0;
#X connect 89 0 90 1;
#X connect 90 0 91 0;
#X restore 158 212 pd \$0-bassist;
#X obj 110 369 dac~;
#N canvas 0 0 450 300 \$0-drummer 0;
#X obj 22 19 inlet;
#X obj 22 40 t f f f f;
#N canvas 0 0 432 362 \$0-kick 0;
#X obj 16 18 inlet;
#X obj 16 39 mod 4;
#X obj 16 60 sel 0;
#X obj 16 102 vline~;
#X obj 16 123 *~;
#X obj 16 144 *~;
#X obj 16 186 -~ 0.25;
#X obj 16 207 cos~;
#X obj 16 296 outlet~;
#X msg 16 81 1 \, 0 150;
#X obj 16 165 *~ 10;
#X connect 0 0 1 0;
#X connect 1 0 2 0;
#X connect 2 0 9 0;
#X connect 3 0 4 0;
#X connect 3 0 4 1;
#X connect 3 0 5 1;
#X connect 4 0 5 0;
```

```
#X connect 5 0 10 0;
#X connect 6 0 7 0;
#X connect 7 0 8 0;
#X connect 9 0 3 0;
#X connect 10 0 6 0;
#X restore 22 113 pd \$0-kick;
#X obj 22 164 outlet~;
#N canvas 4 0 446 470 \$0-hihat 0;
#X obj 15 15 inlet;
#X obj 15 36 mod 8;
#X obj 15 57 t f f;
#X obj 15 78 mod 4;
#X obj 15 99 sel 2;
#X obj 15 141 > 0;
#X obj 86 98 b;
#X obj 86 140 == 0;
#X obj 31 168 sel 0 1;
#X obj 32 231 v \$0-tempo;
#X obj 32 252 / 4;
#X msg 32 273 1 \, 0 \$1;
#X obj 32 294 vline~;
#X obj 32 315 *~;
#X obj 32 336 *~;
#X obj 31 379 *~;
#X obj 255 248 noise~;
#X obj 31 400 outlet~;
#X msg 102 276 1 \, 0 \$1;
#X obj 102 297 vline~;
#X obj 102 318 *~;
#X obj 102 231 v \$0-tempo;
#X obj 32 355 lop~ 50;
#X obj 180 342 hip~ 4000;
#X obj 91 379 *~;
#X obj 91 400 outlet~;
#X obj 252 345 hip~ 4000;
#X obj 265 268 noise~;
#X obj 275 288 noise~;
#X obj 285 308 noise~;
#X obj 86 119 random 32;
#X obj 15 120 random 32;
#X obj 31 202 t b b;
#X obj 69 206 random 16;
#X obj 60 251 + 1;
#X obj 134 206 t b b;
#X obj 102 255 *;
#X obj 174 232 random 16;
#X obj 174 253 / 8;
#X obj 180 362 hip~ 4000;
#X obj 252 365 hip~ 4000;
#X connect 0 0 1 0;
#X connect 1 0 2 0;
#X connect 2 0 3 0;
#X connect 3 0 4 0;
#X connect 4 0 31 0;
#X connect 4 1 6 0;
#X connect 5 0 8 0;
#X connect 6 0 30 0;
```

```
#X connect 7 0 8 0;
#X connect 8 0 32 0;
#X connect 8 1 35 0;
#X connect 9 0 10 0;
#X connect 10 0 11 0;
#X connect 11 0 12 0;
#X connect 12 0 13 0;
#X connect 12 0 13 1;
#X connect 13 0 14 0;
#X connect 13 0 14 1;
#X connect 14 0 22 0;
#X connect 15 0 17 0;
#X connect 16 0 23 0;
#X connect 16 0 26 0;
#X connect 18 0 19 0;
#X connect 19 0 20 0;
#X connect 19 0 20 1;
#X connect 20 0 22 0;
#X connect 21 0 36 0;
#X connect 22 0 15 0;
#X connect 22 0 24 0;
#X connect 23 0 39 0;
#X connect 24 0 25 0;
#X connect 26 0 40 0;
#X connect 27 0 23 0;
#X connect 27 0 26 0;
#X connect 28 0 23 0;
#X connect 29 0 26 0;
#X connect 30 0 7 0;
#X connect 31 0 5 0;
#X connect 32 0 9 0;
#X connect 32 1 33 0;
#X connect 33 0 34 0;
#X connect 34 0 10 1;
#X connect 35 0 21 0;
#X connect 35 1 37 0;
#X connect 36 0 18 0;
#X connect 37 0 38 0;
#X connect 38 0 36 1;
#X connect 39 0 15 1;
#X connect 40 0 24 1;
#X restore 385 111 pd \$0-hihat;
#X obj 385 162 outlet~;
#X obj 445 162 outlet~;
#N canvas 0 0 1066 589 \$0-snare 0;
#X obj 9 4 inlet;
#X obj 44 171 noise~;
#X obj 179 218 vline~;
#X obj 178 242 *~;
#X obj 178 262 *~;
#X obj 48 310 *~;
#X obj 97 212 vline~;
#X obj 47 258 vcf~ 2.5;
#X obj 103 231 vline~;
#X obj 53 277 vcf~ 2.5;
#X obj 54 191 noise~;
#X obj 79 311 *~;
```

```
#X obj 209 238 vline~;
#X obj 208 262 *~;
#X obj 208 282 *~;
#X obj 104 120 t f f f f;
#X obj 78 145 * 1000;
#X msg 212 169 1 1 \, 0 \$1 1;
#X msg 233 192 1 1 \, 0 \$1 1;
#X obj 219 140 * 500;
#X obj 179 140 * 600;
#X obj 128 145 * 800;
#X msg 97 171 1500 1 \, 300 \$1 1;
#X msg 103 190 2500 1 \, 100 \$1 1;
#X obj 57 332 +~;
#X obj 472 172 hilbert~;
#X obj 466 210 complex-mod~;
#X obj 544 188 sig~ 175;
#X obj 467 93 phasor~ 111;
#X obj 557 209 complex-mod~;
#X obj 635 186 sig~ 224;
#X obj 313 211 osc~ 330;
#X obj 404 211 osc~ 180;
#X obj 588 323 *~;
#X obj 473 322 *~;
#X obj 408 321 *~;
#X obj 332 257 vline~;
#X obj 316 319 *~;
#X obj 331 279 *~;
#X obj 332 299 *~;
#X obj 425 257 vline~;
#X obj 424 279 *~;
#X obj 425 299 *~;
#X obj 512 257 vline~;
#X obj 511 279 *~;
#X obj 512 299 *~;
#X obj 595 257 vline~;
#X obj 594 279 *~;
#X obj 595 299 *~;
#X msg 332 236 1 1 \, 0 \$1 1;
#X msg 425 236 1 1 \, 0 \$1 1;
#X msg 512 236 1 1 \, 0 \$1 1;
#X msg 595 236 1 1 \, 0 \$1 1;
#X msg 521 144 bang;
#X obj 304 139 * 500;
#X obj 344 141 * 800;
#X obj 384 139 * 1000;
#X obj 401 117 * 1600;
#X obj 246 550 outlet~;
#X obj 410 549 outlet~;
#X obj 467 119 expr~ if($v1<0.5 \, $v1-0.5 \, 0.5-$v1);
#X obj 714 181 noise~;
#X obj 849 228 vline~;
#X obj 848 252 *~;
#X obj 848 272 *~;
#X obj 718 320 *~;
#X obj 767 222 vline~;
#X obj 717 268 vcf~ 2.5;
#X obj 773 241 vline~;
```

```
#X obj 723 287 vcf~ 2.5;
#X obj 724 201 noise~;
#X obj 749 321 *~;
#X obj 879 248 vline~;
#X obj 878 272 *~;
#X obj 878 292 *~;
#X obj 787 111 t f f f f;
#X obj 758 155 * 1000;
#X msg 882 179 1 1 \, 0 \$1 1;
#X msg 903 202 1 1 \, 0 \$1 1;
#X obj 899 150 * 500;
#X obj 859 150 * 600;
#X obj 808 155 * 800;
#X msg 767 181 1500 1 \, 300 \$1 1;
#X msg 773 200 2500 1 \, 100 \$1 1;
#X obj 727 342 +~;
#X obj 306 94 t f f f f b f;
#X obj 384 192 + 180;
#X obj 325 191 + 330;
#X obj 362 162 * 100;
#X obj 246 476 expr~ tanh($v2*5+4*$v1) \; tanh($v2*5+4*$v3);
#X obj 102 103 / 200;
#X obj 8 30 mod 8;
#X obj 8 50 sel 4;
#X obj 8 72 random 50;
#X obj 8 93 + 50;
#X obj 87 7 b;
#X obj 87 28 random 4;
#X obj 87 49 sel 0;
#X obj 87 70 random 30;
#X obj 87 88 + 10;
#X obj 520 404 *~ 3;
#X obj 117 389 *~ 3;
#X obj 431 399 *~ 1;
#X connect 0 0 91 0;
#X connect 1 0 7 0;
#X connect 2 0 3 0;
#X connect 2 0 3 1;
#X connect 3 0 4 0;
#X connect 3 0 4 1;
#X connect 4 0 5 1;
#X connect 5 0 24 0;
#X connect 6 0 7 1;
#X connect 7 0 5 0;
#X connect 8 0 9 1;
#X connect 9 0 11 0;
#X connect 10 0 9 0;
#X connect 11 0 24 1;
#X connect 12 0 13 0;
#X connect 12 0 13 1;
#X connect 13 0 14 0;
#X connect 13 0 14 1;
#X connect 14 0 11 1;
#X connect 15 0 16 0;
#X connect 15 1 21 0;
#X connect 15 2 20 0;
#X connect 15 3 19 0;
```

```
#X connect 16 0 22 0;
#X connect 17 0 2 0;
#X connect 18 0 12 0;
#X connect 19 0 18 0;
#X connect 20 0 17 0;
#X connect 21 0 23 0;
#X connect 22 0 6 0;
#X connect 23 0 8 0;
#X connect 24 0 101 0;
#X connect 25 0 26 0;
#X connect 25 0 29 0;
#X connect 25 1 26 1;
#X connect 25 1 29 1;
#X connect 26 0 34 0;
#X connect 27 0 26 2;
#X connect 28 0 60 0;
#X connect 29 0 33 0;
#X connect 30 0 29 2;
#X connect 31 0 37 0;
#X connect 32 0 35 0;
#X connect 33 0 102 0;
#X connect 34 0 102 0;
#X connect 35 0 102 0;
#X connect 36 0 38 0;
#X connect 36 0 38 1;
#X connect 37 0 102 0;
#X connect 38 0 39 0;
#X connect 38 0 39 1;
#X connect 39 0 37 1;
#X connect 40 0 41 0;
#X connect 40 0 41 1;
#X connect 41 0 42 0;
#X connect 41 0 42 1;
#X connect 42 0 35 1;
#X connect 43 0 44 0;
#X connect 43 0 44 1;
#X connect 44 0 45 0;
#X connect 44 0 45 1;
#X connect 45 0 34 1;
#X connect 46 0 47 0;
#X connect 46 0 47 1;
#X connect 47 0 48 0;
#X connect 47 0 48 1;
#X connect 48 0 33 1;
#X connect 49 0 36 0;
#X connect 50 0 40 0;
#X connect 51 0 43 0;
#X connect 52 0 46 0;
#X connect 53 0 25 1;
#X connect 54 0 49 0;
#X connect 55 0 50 0;
#X connect 56 0 51 0;
#X connect 57 0 52 0;
#X connect 60 0 25 0;
#X connect 61 0 67 0;
#X connect 62 0 63 0;
#X connect 62 0 63 1;
```

```
#X connect 63 0 64 0;
#X connect 63 0 64 1;
#X connect 64 0 65 1;
#X connect 65 0 84 0;
#X connect 66 0 67 1;
#X connect 67 0 65 0;
#X connect 68 0 69 1;
#X connect 69 0 71 0;
#X connect 70 0 69 0;
#X connect 71 0 84 1;
#X connect 72 0 73 0;
#X connect 72 0 73 1;
#X connect 73 0 74 0;
#X connect 73 0 74 1;
#X connect 74 0 71 1;
#X connect 75 0 76 0;
#X connect 75 1 81 0;
#X connect 75 2 80 0;
#X connect 75 3 79 0;
#X connect 76 0 82 0;
#X connect 77 0 62 0;
#X connect 78 0 72 0;
#X connect 79 0 78 0;
#X connect 80 0 77 0;
#X connect 81 0 83 0;
#X connect 82 0 66 0;
#X connect 83 0 68 0;
#X connect 84 0 100 0;
#X connect 85 0 54 0;
#X connect 85 1 55 0;
#X connect 85 2 56 0;
#X connect 85 3 57 0;
#X connect 85 4 53 0;
#X connect 85 5 88 0;
#X connect 86 0 32 0;
#X connect 87 0 31 0;
#X connect 88 0 87 0;
#X connect 88 0 86 0;
#X connect 89 0 58 0;
#X connect 89 1 59 0;
#X connect 90 0 15 0;
#X connect 90 0 75 0;
#X connect 90 0 85 0;
#X connect 91 0 92 0;
#X connect 92 0 93 0;
#X connect 92 1 95 0;
#X connect 93 0 94 0;
#X connect 94 0 90 0;
#X connect 95 0 96 0;
#X connect 96 0 97 0;
#X connect 97 0 98 0;
#X connect 98 0 99 0;
#X connect 99 0 90 0;
#X connect 100 0 89 2;
#X connect 101 0 89 0;
#X connect 102 0 89 1;
#X restore 135 112 pd \$0-snare;
```

```
#X obj 135 165 outlet~;
#X obj 205 165 outlet~;
#N canvas 0 0 975 620 \$0-clap 0;
#X obj 244 70 noise~;
#X obj -50 501 *~;
#X obj -20 466 *~;
#X obj 184 317 *~ 0.8;
#X obj 29 497 *~ 0.2;
#X obj -36 215 del 30;
#X obj -11 247 del 20;
#X obj 6 282 del 15;
#X obj 25 320 del 10;
#X obj 80 284 *~ 0.1;
#X obj 134 254 bp~ 1300 10;
#X obj 302 254 bp~ 700 10;
#X obj 224 254 bp~ 2500 4;
#X obj -42 531 bp~ 2700 2;
#X obj 195 115 hip~ 300;
#X obj -191 384 *~;
#X obj -190 335 vline~;
#X obj -190 359 *~;
#X obj -100 381 *~;
#X obj -99 332 vline~;
#X obj -99 356 *~;
#X obj -96 286 bang;
#X obj 52 254 lop~ 10000;
#X obj -90 132 t b f;
#X obj -174 158 * 100;
#X obj -122 157 * 13;
#X obj -175 181 + 1000;
#X obj -122 181 + 1000;
#X obj -36 159 * 25;
#X obj 45 156 * 7;
#X obj 45 180 + 100;
#X obj -37 182 + 800;
#X obj -89 110 + 100;
#X obj -389 50 noise~;
#X obj -252 312 *~ 0.8;
#X obj -356 279 *~ 0.1;
#X obj -338 250 bp~ 1300 10;
#X obj -170 250 bp~ 700 10;
#X obj -248 250 bp~ 2500 4;
#X obj -241 110 hip~ 300;
#X obj -420 250 lop~ 10000;
#X obj -254 503 *~;
#X obj -229 454 *~;
#X obj -175 499 *~ 0.2;
#X obj -246 533 bp~ 2700 2;
#X obj -33 55 noise~;
#X obj -144 10 inlet;
#X obj -246 581 outlet~;
#X obj -246 554 expr~ tanh($v1);
#X obj -42 582 outlet~;
#X obj -42 555 expr~ tanh($v1);
#X msg -97 309 1.3 3 \, 0 100 3;
#X msg -198 308 1.3 1 \, 0 800 1;
#X obj -90 89 * 2;
```

```
#X obj -144 31 mod 8;
#X obj -144 52 sel 4;
#X obj -144 73 f 50;
#X connect 0 0 14 0;
#X connect 0 0 22 0;
#X connect 1 0 13 0;
#X connect 2 0 1 1;
#X connect 2 0 4 0;
#X connect 3 0 2 1;
#X connect 4 0 13 0;
#X connect 5 0 6 0;
#X connect 5 0 21 0;
#X connect 6 0 7 0;
#X connect 6 0 21 0;
#X connect 7 0 8 0;
#X connect 7 0 21 0;
#X connect 8 0 21 0;
#X connect 9 0 3 0;
#X connect 10 0 3 0;
#X connect 11 0 3 0;
#X connect 12 0 3 0;
#X connect 13 0 50 0;
#X connect 14 0 10 0;
#X connect 14 0 11 0;
#X connect 14 0 12 0;
#X connect 15 0 2 0;
#X connect 15 0 42 1;
#X connect 16 0 17 0;
#X connect 16 0 17 1;
#X connect 17 0 15 0;
#X connect 17 0 15 1;
#X connect 18 0 1 0;
#X connect 18 0 41 1;
#X connect 19 0 20 0;
#X connect 19 0 20 1;
#X connect 20 0 18 0;
#X connect 20 0 18 1;
#X connect 21 0 51 0;
#X connect 22 0 9 0;
#X connect 23 0 5 0;
#X connect 23 0 52 0;
#X connect 23 0 21 0;
#X connect 23 1 29 0;
#X connect 23 1 28 0;
#X connect 23 1 25 0;
#X connect 23 1 24 0;
#X connect 24 0 26 0;
#X connect 25 0 27 0;
#X connect 26 0 22 1;
#X connect 26 0 40 1;
#X connect 27 0 10 1;
#X connect 27 0 36 1;
#X connect 28 0 31 0;
#X connect 29 0 30 0;
#X connect 30 0 11 1;
#X connect 30 0 37 1;
#X connect 31 0 12 1;
```

```
#X connect 31 0 38 1;
#X connect 32 0 23 0;
#X connect 33 0 39 0;
#X connect 33 0 40 0;
#X connect 34 0 42 0;
#X connect 35 0 34 0;
#X connect 36 0 34 0;
#X connect 37 0 34 0;
#X connect 38 0 34 0;
#X connect 39 0 36 0;
#X connect 39 0 37 0;
#X connect 39 0 38 0;
#X connect 40 0 35 0;
#X connect 41 0 44 0;
#X connect 42 0 43 0;
#X connect 42 0 41 0;
#X connect 43 0 44 0;
#X connect 44 0 48 0;
#X connect 45 0 14 0;
#X connect 45 0 39 0;
#X connect 45 0 40 0;
#X connect 45 0 22 0;
#X connect 46 0 54 0;
#X connect 48 0 47 0;
#X connect 50 0 49 0;
#X connect 51 0 19 0;
#X connect 52 0 16 0;
#X connect 53 0 32 0;
#X connect 54 0 55 0;
#X connect 55 0 56 0;
#X connect 56 0 53 0;
#X restore 250 112 pd \$0-clap;
#X obj 261 164 outlet~;
#X obj 331 164 outlet~;
#X connect 0 0 1 0;
#X connect 1 0 2 0;
#X connect 1 1 7 0;
#X connect 1 2 10 0;
#X connect 1 3 4 0;
#X connect 2 0 3 0;
#X connect 4 0 5 0;
#X connect 4 1 6 0;
#X connect 7 0 8 0;
#X connect 7 1 9 0;
#X connect 10 0 11 0;
#X connect 10 1 12 0;
#X restore 40 313 pd \$0-drummer;
#X obj 288 73 table \$0-accent 32;
#N canvas 0 0 450 300 \$0-accentuate 0;
#X obj 18 14 inlet;
#X obj 18 56 tabread \$0-accent;
#X obj 18 77 sel 0 1 2 3 4 5 6 7;
#X obj 18 98 outlet;
#X obj 25 119 outlet;
#X obj 33 142 outlet;
#X obj 45 167 outlet;
#X obj 68 98 outlet;
```

```
#X obj 75 119 outlet;
#X obj 83 142 outlet;
#X obj 95 167 outlet;
#X obj 18 35 mod 32;
#X connect 0 0 11 0;
#X connect 1 0 2 0;
#X connect 2 0 3 0;
#X connect 2 1 4 0;
#X connect 2 2 5 0;
#X connect 2 3 6 0;
#X connect 2 4 7 0;
#X connect 2 5 8 0;
#X connect 2 6 9 0;
#X connect 2 7 10 0;
#X connect 11 0 1 0;
#X restore 171 164 pd \$0-accentuate;
#N canvas 0 0 450 406 \$0-dubpad 0;
#X obj 342 26 inlet~;
#X obj 24 29 inlet;
#X obj 16 383 delwrite~ \$0-dubpad-l 10000;
#X obj 204 384 delwrite~ \$0-dubpad-r 10000;
#X obj 203 228 vd~ \$0-dubpad-r;
#X obj 15 231 vd~ \$0-dubpad-l;
#X obj 54 355 outlet~;
#X obj 140 357 outlet~;
#X obj 24 53 t b b b;
#X obj 22 110 v \$0-dubpad-l-del;
#X obj 46 91 v \$0-dubpad-r-del;
#X obj 16 176 *;
#X obj 63 72 v \$0-tempo;
#X obj 16 197 sig~;
#X obj 204 178 *;
#X obj 204 199 sig~;
#X obj 15 324 expr~ tanh($v1);
#X obj 203 325 expr~ tanh($v1);
#X obj 203 274 lop~ 3000;
#X obj 15 273 lop~ 3000;
#X obj 203 253 hip~ 300;
#X obj 15 252 hip~ 300;
#X obj 229 25 inlet;
#X obj 229 46 v \$0-tempo;
#X msg 235 140 1 \, 0 \$1;
#X obj 235 161 vline~;
#X obj 235 182 *~;
#X obj 235 203 *~;
#X obj 159 25 inlet;
#X obj 159 46 v \$0-tempo;
#X msg 51 149 1 \, 0 \$1;
#X obj 51 170 vline~;
#X obj 51 191 *~;
#X obj 51 212 *~;
#X obj 181 299 *~ 0.25;
#X obj 65 299 *~ 0.25;
#X obj 111 150 phasor~;
#X obj 111 171 -~ 0.25;
#X obj 111 192 cos~;
#X obj 151 192 cos~;
```

```
#X obj 158 115 /;
#X obj 158 91 swap 1000;
#X obj 158 72 * 8;
#X obj 111 213 *~ 2;
#X obj 151 213 *~ 2;
#X obj 235 120 * 2;
#X obj 51 128 * 2;
#X obj 2 8 inlet;
#X obj 329 208 route dubfb;
#X obj 329 229 vline~;
#X obj 231 299 *~;
#X obj 14 298 *~;
#X obj 329 250 *~ 0.85;
#X connect 0 0 27 1;
#X connect 0 0 33 1;
#X connect 1 0 8 0;
#X connect 4 0 20 0;
#X connect 5 0 21 0;
#X connect 8 0 9 0;
#X connect 8 1 10 0;
#X connect 8 2 12 0;
#X connect 9 0 11 0;
#X connect 10 0 14 0;
#X connect 11 0 13 0;
#X connect 12 0 11 1;
#X connect 12 0 14 1;
#X connect 12 0 42 0;
#X connect 13 0 5 0;
#X connect 14 0 15 0;
#X connect 15 0 4 0;
#X connect 16 0 2 0;
#X connect 16 0 6 0;
#X connect 17 0 3 0;
#X connect 17 0 7 0;
#X connect 18 0 34 0;
#X connect 18 0 50 0;
#X connect 19 0 35 0;
#X connect 19 0 51 0;
#X connect 20 0 18 0;
#X connect 21 0 19 0;
#X connect 22 0 23 0;
#X connect 23 0 45 0;
#X connect 24 0 25 0;
#X connect 25 0 26 0;
#X connect 25 0 26 1;
#X connect 26 0 27 0;
#X connect 27 0 20 0;
#X connect 28 0 29 0;
#X connect 29 0 46 0;
#X connect 30 0 31 0;
#X connect 31 0 32 0;
#X connect 31 0 32 1;
#X connect 32 0 33 0;
#X connect 33 0 21 0;
#X connect 34 0 16 0;
#X connect 35 0 17 0;
#X connect 36 0 37 0;
```

```
#X connect 36 0 39 0;
#X connect 37 0 38 0;
#X connect 38 0 43 0;
#X connect 39 0 44 0;
#X connect 40 0 36 0;
#X connect 41 0 40 0;
#X connect 41 1 40 1;
#X connect 42 0 41 0;
#X connect 43 0 5 0;
#X connect 44 0 4 0;
#X connect 45 0 24 0;
#X connect 46 0 30 0;
#X connect 47 0 48 0;
#X connect 48 0 49 0;
#X connect 49 0 52 0;
#X connect 50 0 17 0;
#X connect 51 0 16 0;
#X connect 52 0 50 1;
#X connect 52 0 51 1;
#X restore 173 248 pd \$0-dubpad;
#X obj 289 124 value \$0-dubpad-l-del;
#X obj 289 145 value \$0-dubpad-r-del;
#X obj 85 71 t b b;
#N canvas 481 95 450 567 \$0-trichords 0;
#X obj 22 14 inlet;
#X obj 22 35 + 32;
#X obj 22 56 mod 64;
#X obj 22 77 moses 47.5;
#X obj 22 98 mod 16;
#X obj 80 98 - 32;
#X obj 22 127 tabread \$0-triline;
#X obj 22 148 sel -1;
#X obj 70 149 tabread \$0-scale;
#X obj 70 190 + 12;
#X obj 70 211 mtof;
#X obj 70 232 vline~;
#X obj 70 274 phasor~;
#X obj 70 335 expr~ if($v1<$v2 \, $v1/$v2 \, (1-$v1)/(1-$v2));
#X obj 325 229 vline~;
#X obj 325 274 clip~ 0.001 0.999;
#X msg 325 208 0 \, 0.707 \$1;
#X obj 326 253 *~;
#X obj 80 355 expr~ if($v1<$v2 \, $v1/$v2 \, (1-$v1)/(1-$v2));
#X obj 90 375 expr~ if($v1<$v2 \, $v1/$v2 \, (1-$v1)/(1-$v2));
#X obj 80 294 phasor~;
#X obj 90 314 phasor~;
#X obj 120 211 + 1;
#X obj 190 194 v \$0-tempo;
#X obj 120 232 *;
#X obj 69 469 -~ 0.25;
#X obj 69 490 cos~;
#X obj 69 422 hip~ 5;
#X obj 69 511 outlet~;
#X obj 69 445 *~;
#X msg 151 293 1 \, 0 \$1;
#X obj 151 314 vline~;
#X obj 120 190 random 8;
```

```
#X obj 69 169 t f b b;
#X obj 129 469 -~ 0.25;
#X obj 129 490 cos~;
#X obj 129 511 outlet~;
#X obj 129 445 *~;
#X obj 200 251 + 1;
#X obj 200 272 *;
#X obj 200 230 random 8;
#X msg 205 293 1 \, 0 \$1;
#X obj 205 314 vline~;
#X obj 227 13 inlet;
#X obj 227 105 v \$0-tempo;
#X msg 227 147 1 \, 0 \$1;
#X obj 227 168 vline~;
#X obj 70 253 lop~ 8;
#X obj 163 254 -~ 3;
#X obj 124 253 +~ 3;
#X obj 325 187 * 4;
#X connect 0 0 1 0;
#X connect 1 0 2 0;
#X connect 2 0 3 0;
#X connect 3 0 4 0;
#X connect 3 1 5 0;
#X connect 4 0 6 0;
#X connect 5 0 6 0;
#X connect 6 0 7 0;
#X connect 7 1 8 0;
#X connect 8 0 33 0;
#X connect 9 0 10 0;
#X connect 10 0 11 0;
#X connect 11 0 47 0;
#X connect 12 0 13 0;
#X connect 13 0 27 0;
#X connect 14 0 17 0;
#X connect 14 0 17 1;
#X connect 15 0 13 1;
#X connect 15 0 18 1;
#X connect 15 0 19 1;
#X connect 16 0 14 0;
#X connect 17 0 15 0;
#X connect 18 0 27 0;
#X connect 19 0 27 0;
#X connect 20 0 18 0;
#X connect 21 0 19 0;
#X connect 22 0 24 0;
#X connect 23 0 24 1;
#X connect 23 0 39 1;
#X connect 23 0 50 0;
#X connect 24 0 30 0;
#X connect 25 0 26 0;
#X connect 26 0 28 0;
#X connect 27 0 29 0;
#X connect 27 0 37 0;
#X connect 29 0 25 0;
#X connect 30 0 31 0;
#X connect 31 0 29 1;
#X connect 32 0 22 0;
```

```
#X connect 33 0 9 0;
#X connect 33 1 32 0;
#X connect 33 1 40 0;
#X connect 33 2 23 0;
#X connect 34 0 35 0;
#X connect 35 0 36 0;
#X connect 37 0 34 0;
#X connect 38 0 39 0;
#X connect 39 0 41 0;
#X connect 40 0 38 0;
#X connect 41 0 42 0;
#X connect 42 0 37 1;
#X connect 43 0 44 0;
#X connect 44 0 45 0;
#X connect 45 0 46 0;
#X connect 46 0 29 1;
#X connect 46 0 37 1;
#X connect 47 0 12 0;
#X connect 47 0 48 0;
#X connect 47 0 49 0;
#X connect 48 0 21 0;
#X connect 49 0 20 0;
#X connect 50 0 16 0;
#X restore 259 245 pd \$0-trichords;
#X obj 288 7 table \$0-triline 32;
#N canvas 0 207 1001 490 \$0-mixer 0;
#X obj 68 41 inlet~;
#X obj 486 45 inlet~;
#X obj 143 376 outlet~;
#X obj 224 375 outlet~;
#X obj 304 42 inlet~;
#X obj 349 43 inlet~;
#X obj 534 46 inlet~;
#X obj 594 46 inlet~;
#X obj 662 42 inlet~;
#X obj 722 42 inlet~;
#X obj 113 43 inlet~;
#X obj 163 43 inlet~;
#X obj 792 42 inlet~;
#X obj 852 42 inlet~;
#N canvas 0 0 450 643 \$0-compress 0;
#X obj 28 28 inlet~;
#X obj 176 20 inlet~;
#X obj 27 543 outlet~;
#X obj 177 544 outlet~;
#X obj 58 71 hip~ 5;
#X obj 121 71 hip~ 5;
#X obj 58 97 *~;
#X obj 120 98 *~;
#X obj 87 158 +~;
#X obj 87 281 rmstodb~;
#X obj 86 353 dbtorms~;
#X obj 27 484 *~;
#X obj 177 484 *~;
#X obj 25 272 /~;
#X obj 178 270 /~;
#X obj 201 390 dbtorms;
```

```
#X obj 86 417 /~ 0;
#X obj 86 197 sqrt~;
#X obj 86 442 *~ 0.25;
#X obj 310 55 loadbang;
#X obj 27 514 expr~ tanh($v1);
#X obj 177 514 expr~ tanh($v1);
#X obj 310 79 f 48;
#X obj 87 233 lop~ 10;
#X obj 120 123 lop~ 10;
#X obj 57 124 lop~ 10;
#X obj 86 314 expr~ if($v1>$f2 \, $f2+($v1-$f2)/4 \, $f2);
#X obj 202 367 expr (100-$f1)/4+$f1;
#X connect 0 0 4 0;
#X connect 0 0 13 0;
#X connect 1 0 5 0;
#X connect 1 0 14 0;
#X connect 4 0 6 0;
#X connect 4 0 6 1;
#X connect 5 0 7 0;
#X connect 5 0 7 1;
#X connect 6 0 25 0;
#X connect 7 0 24 0;
#X connect 8 0 17 0;
#X connect 9 0 26 0;
#X connect 10 0 16 0;
#X connect 11 0 20 0;
#X connect 12 0 21 0;
#X connect 13 0 11 0;
#X connect 14 0 12 0;
#X connect 15 0 16 1;
#X connect 16 0 18 0;
#X connect 17 0 23 0;
#X connect 18 0 11 1;
#X connect 18 0 12 1;
#X connect 19 0 22 0;
#X connect 20 0 2 0;
#X connect 21 0 3 0;
#X connect 22 0 26 1;
#X connect 22 0 27 0;
#X connect 23 0 13 1;
#X connect 23 0 14 1;
#X connect 23 0 9 0;
#X connect 24 0 8 1;
#X connect 25 0 8 0;
#X connect 26 0 10 0;
#X connect 27 0 15 0;
#X restore 143 301 pd \$0-compress;
#X obj 203 43 inlet~;
#X obj 253 43 inlet~;
#X obj 83 150 vline~;
#X obj 66 205 *~;
#X obj 113 207 *~;
#X obj 163 207 *~;
#X obj 141 150 vline~;
#X obj 211 205 *~;
#X obj 261 205 *~;
#X obj 238 152 vline~;
```

```
#X obj 334 155 vline~;
#X obj 304 206 *~;
#X obj 349 207 *~;
#X obj 479 203 *~;
#X obj 492 157 vline~;
#X obj 682 145 vline~;
#X obj 663 196 *~;
#X obj 723 196 *~;
#X obj 793 196 *~;
#X obj 853 196 *~;
#X obj 816 146 vline~;
#X obj 568 157 vline~;
#X obj 533 200 *~;
#X obj 595 198 *~;
#X obj 8 44 inlet;
#X obj 148 333 *~;
#X obj 492 277 vline~;
#X obj 220 335 *~;
#X obj 394 42 inlet~;
#X obj 439 43 inlet~;
#X obj 188 77 route kick snare clap hihat bass dubpad trichords syncer
master breaks;
#X obj 380 206 *~ 1.25;
#X obj 429 205 *~ 1.25;
#X connect 0 0 18 0;
#X connect 1 0 28 0;
#X connect 4 0 26 0;
#X connect 5 0 27 0;
#X connect 6 0 37 0;
#X connect 7 0 38 0;
#X connect 8 0 31 0;
#X connect 9 0 32 0;
#X connect 10 0 19 0;
#X connect 11 0 20 0;
#X connect 12 0 33 0;
#X connect 13 0 34 0;
#X connect 14 0 40 0;
#X connect 14 1 42 0;
#X connect 15 0 22 0;
#X connect 16 0 23 0;
#X connect 17 0 18 1;
#X connect 18 0 14 0;
#X connect 18 0 14 1;
#X connect 19 0 14 0;
#X connect 20 0 14 1;
#X connect 21 0 19 1;
#X connect 21 0 20 1;
#X connect 22 0 14 0;
#X connect 23 0 14 1;
#X connect 24 0 22 1;
#X connect 24 0 23 1;
#X connect 25 0 26 1;
#X connect 25 0 27 1;
#X connect 26 0 14 0;
#X connect 27 0 14 1;
#X connect 28 0 14 0;
#X connect 28 0 14 1;
```

```
#X connect 29 0 28 1;
#X connect 30 0 31 1;
#X connect 30 0 32 1;
#X connect 31 0 14 0;
#X connect 32 0 14 1;
#X connect 33 0 14 0;
#X connect 34 0 14 1;
#X connect 35 0 33 1;
#X connect 35 0 34 1;
#X connect 36 0 37 1;
#X connect 36 0 38 1;
#X connect 37 0 14 0;
#X connect 38 0 14 1;
#X connect 39 0 45 0;
#X connect 40 0 2 0;
#X connect 41 0 40 1;
#X connect 41 0 42 1;
#X connect 42 0 3 0;
#X connect 43 0 46 0;
#X connect 44 0 47 0;
#X connect 45 0 17 0;
#X connect 45 1 21 0;
#X connect 45 2 24 0;
#X connect 45 3 25 0;
#X connect 45 4 29 0;
#X connect 45 5 36 0;
#X connect 45 6 30 0;
#X connect 45 7 35 0;
#X connect 45 8 41 0;
#X connect 46 0 14 0;
#X connect 47 0 14 1;
#X restore 23 337 pd \$0-mixer ---------------------------;
#X obj 288 173 table \$0-syncline 32;
#N canvas 0 0 450 654 \$0-syncer 0;
#X obj 15 42 inlet;
#X obj 15 84 mod 64;
#X obj 15 105 moses 47.5;
#X obj 73 126 - 32;
#X obj 15 176 sel -1;
#X obj 18 226 tabread \$0-scale;
#X obj 18 288 mtof;
#X obj 18 309 vline~;
#X obj 18 351 phasor~;
#X obj 224 377 + 1;
#X obj 271 316 v \$0-tempo;
#X obj 224 398 *;
#X obj 62 450 hip~ 5;
#X obj 62 539 outlet~;
#X obj 62 473 *~;
#X msg 231 420 1 \, 0 \$1;
#X obj 231 441 vline~;
#X obj 17 246 t f b b;
#X obj 164 540 outlet~;
#X obj 165 478 *~;
#X obj 280 378 + 1;
#X obj 280 399 *;
#X msg 285 420 1 \, 0 \$1;
```

```
#X obj 285 441 vline~;
#X obj 220 41 inlet;
#X obj 220 133 v \$0-tempo;
#X obj 220 154 * 8;
#X obj 220 196 vline~;
#X obj 220 73 t b b;
#X obj 261 72 random 4;
#X obj 261 93 + 4;
#X obj 15 63 + 16;
#X obj 59 381 *~;
#X obj 61 404 wrap~;
#X obj 118 374 +~ 1;
#X obj 118 334 vline~;
#X obj 118 353 *~;
#X obj 15 155 tabread \$0-syncline;
#X obj 18 203 mod 5;
#X obj 123 179 div 5;
#X obj 162 249 *;
#X obj 123 199 + 1;
#X obj 178 229 v \$0-tempo;
#X obj 61 517 expr~ tanh($v1);
#X obj 164 519 expr~ tanh($v1);
#X obj 117 295 pack f f;
#X msg 117 316 \$1 \, 0 \$2;
#X obj 123 226 t f f b;
#X obj 117 252 sqrt;
#X obj 61 427 *~ 16;
#X obj 18 330 lop~ 25;
#X obj 160 273 / 3;
#X msg 220 175 2 \, 0 \$1;
#X obj 280 357 random 4;
#X obj 224 356 random 4;
#X obj 15 126 mod 16;
#X connect 0 0 31 0;
#X connect 1 0 2 0;
#X connect 2 0 55 0;
#X connect 2 1 3 0;
#X connect 3 0 37 0;
#X connect 4 1 38 0;
#X connect 4 1 39 0;
#X connect 5 0 17 0;
#X connect 6 0 7 0;
#X connect 7 0 50 0;
#X connect 8 0 32 0;
#X connect 9 0 11 0;
#X connect 10 0 11 1;
#X connect 10 0 21 1;
#X connect 11 0 15 0;
#X connect 12 0 14 0;
#X connect 12 0 19 0;
#X connect 14 0 43 0;
#X connect 15 0 16 0;
#X connect 16 0 14 1;
#X connect 17 0 6 0;
#X connect 17 1 53 0;
#X connect 17 1 54 0;
#X connect 17 2 10 0;
```

```
#X connect 19 0 44 0;
#X connect 20 0 21 0;
#X connect 21 0 22 0;
#X connect 22 0 23 0;
#X connect 23 0 19 1;
#X connect 24 0 28 0;
#X connect 25 0 26 0;
#X connect 26 0 52 0;
#X connect 27 0 34 0;
#X connect 28 0 25 0;
#X connect 28 1 29 0;
#X connect 29 0 30 0;
#X connect 30 0 26 1;
#X connect 31 0 1 0;
#X connect 32 0 33 0;
#X connect 33 0 49 0;
#X connect 34 0 32 1;
#X connect 35 0 36 0;
#X connect 35 0 36 1;
#X connect 36 0 34 0;
#X connect 37 0 4 0;
#X connect 38 0 5 0;
#X connect 39 0 41 0;
#X connect 40 0 51 0;
#X connect 41 0 47 0;
#X connect 42 0 40 1;
#X connect 43 0 13 0;
#X connect 44 0 18 0;
#X connect 45 0 46 0;
#X connect 46 0 35 0;
#X connect 47 0 48 0;
#X connect 47 1 40 0;
#X connect 47 2 42 0;
#X connect 48 0 45 0;
#X connect 49 0 12 0;
#X connect 50 0 8 0;
#X connect 51 0 45 1;
#X connect 52 0 27 0;
#X connect 53 0 20 0;
#X connect 54 0 9 0;
#X connect 55 0 37 0;
#X restore 363 247 pd \$0-syncer;
#X obj 290 193 value \$0-swing;
#N canvas 13 38 929 734 \$0-sequencer 0;
#X obj 68 5 inlet;
#X obj 122 666 outlet;
#X obj 68 26 div 64;
#X obj 68 47 change;
#X obj 15 23 inlet;
#X obj 68 90 sel -1;
#X obj 68 69 - 1;
#X obj 68 158 sel 0;
#X obj 67 196 v \$0-tempo;
#X obj 67 217 * 128;
#X obj 147 163 sel 1;
#X obj 147 196 v \$0-tempo;
#X obj 147 217 * 128;
```

```
#X msg 147 259 clap 3 \$1;
#X obj 217 196 v \$0-tempo;
#X obj 217 217 * 128;
#X obj 217 163 sel 2;
#X msg 217 260 snare 2.5 \$1;
#X obj 307 163 sel 3;
#X msg 307 211 kick 0 \, clap 0;
#X obj 64 277 sel 4;
#X obj 64 297 v \$0-tempo;
#X obj 64 318 * 128;
#X obj 291 295 v \$0-tempo;
#X obj 291 316 * 128;
#X obj 291 275 sel 5;
#X msg 291 339 hihat 2 \$1;
#X obj 371 295 v \$0-tempo;
#X obj 371 316 * 128;
#X obj 371 275 sel 6;
#X msg 371 339 trichords 2 \$1;
#X obj 476 276 sel 7;
#X obj 481 295 v \$0-tempo;
#X obj 481 316 * 64;
#X msg 481 339 bass 0 \$1;
#X obj 62 369 sel 8;
#X obj 267 370 sel 9;
#X obj 268 392 v \$0-tempo;
#X obj 268 413 * 64;
#X obj 359 372 sel 10;
#X obj 358 392 v \$0-tempo;
#X obj 358 413 * 128;
#X obj 477 374 sel 11;
#X obj 477 395 v \$0-tempo;
#X obj 477 416 * 64;
#X msg 477 454 snare 2.5 \$1;
#X obj 13 464 sel 12;
#X obj 319 495 v \$0-tempo;
#X obj 340 520 * 256;
#X msg 321 543 trichords 2 \, dubpad 6 \$1;
#X msg 405 498 syncer 1;
#X obj 52 571 v \$0-tempo;
#X msg 26 621 bass 0 \$1;
#X obj 342 661 outlet;
#X msg 309 626 bassfx 0 \, dubfb 0 \$1;
#X obj 180 569 v \$0-tempo;
#X obj 6 662 outlet;
#X obj 67 177 t b b;
#X obj 180 589 * 256;
#X msg 25 258 bassfx 1 \, dubfb 1;
#X msg 166 91 dubfb 1;
#X msg 64 341 kick 8 \, clap 3 \, bass 3 \, syncer 1 \$1;
#X msg 359 434 kick 8 \, clap 3 \, hihat 0 \, bass 3 \$1;
#X obj 484 504 v \$0-tempo;
#X obj 484 525 * 256;
#X obj 63 401 v \$0-tempo;
#X obj 63 422 * 256;
#X msg 269 432 hihat 2 \$1;
#X msg 482 551 breaks 1.5 \$1 \, kick 0 \, snare 0 \, clap 0 \, hihat
0 \, trichords 0 \, syncer 0;
```

```
#X msg 63 447 kick 0 \, clap 0 \, snare 0 \, hihat 0 \, breaks 1.5
\$1;
#X msg 67 237 master 0.9 \, kick 8 \, dubpad 6;
#X msg 15 127 kick 0 \, snare 0 \, clap 0 \, hihat 0 \, breaks 0 \,
bass 0 \, dubpad 0 \, trichords 0 \, syncer 0 \, master 0.9;
#X obj 14 483 v \$0-tempo;
#X obj 14 504 * 256;
#X msg 92 623 breaks 0 \$1;
#X msg 13 523 hihat 2 \, trichords 0 \, syncer 0 \, dubpad 0 \, breaks
0.1 \$1;
#X obj 558 659 outlet;
#X msg 598 358 dthrow -1;
#X msg 632 405 dthrow 4;
#X msg 659 502 balls 2 \, throw 2;
#X msg 627 614 balls 2;
#X msg 599 325 dthrow 1;
#X msg 628 590 balls 2;
#X obj 115 482 sel 13;
#X floatatom 491 147 5 0 0 0 - - -;
#X msg 439 620 dballs 1 \, dthrow 1;
#X obj 316 469 sel 14;
#X obj 405 477 sel 15;
#X obj 481 477 sel 16;
#X obj 53 550 sel 18;
#X obj 179 550 sel 23;
#X obj 27 600 * 192;
#X obj 101 108 mod 24;
#X obj 87 600 * 320;
#X obj 121 573 v \$0-tempo;
#X obj 119 551 sel 20;
#X obj 123 596 * 256;
#X msg 215 626 dubfb 2 \$1;
#X connect 0 0 2 0;
#X connect 2 0 3 0;
#X connect 3 0 6 0;
#X connect 4 0 71 0;
#X connect 4 0 60 0;
#X connect 5 0 71 0;
#X connect 5 0 60 0;
#X connect 5 1 92 0;
#X connect 6 0 5 0;
#X connect 7 0 57 0;
#X connect 7 0 79 0;
#X connect 7 1 10 0;
#X connect 8 0 9 0;
#X connect 9 0 70 0;
#X connect 10 0 11 0;
#X connect 10 0 81 0;
#X connect 10 1 16 0;
#X connect 11 0 12 0;
#X connect 12 0 13 0;
#X connect 13 0 1 0;
#X connect 14 0 15 0;
#X connect 15 0 17 0;
#X connect 16 0 14 0;
#X connect 16 0 85 0;
#X connect 16 1 18 0;
```

```
#X connect 17 0 1 0;
#X connect 18 0 19 0;
#X connect 18 0 81 0;
#X connect 18 1 20 0;
#X connect 19 0 1 0;
#X connect 20 0 21 0;
#X connect 20 0 78 0;
#X connect 20 0 85 0;
#X connect 20 1 25 0;
#X connect 21 0 22 0;
#X connect 22 0 61 0;
#X connect 23 0 24 0;
#X connect 24 0 26 0;
#X connect 25 0 23 0;
#X connect 25 1 29 0;
#X connect 26 0 1 0;
#X connect 27 0 28 0;
#X connect 28 0 30 0;
#X connect 29 0 27 0;
#X connect 29 0 77 0;
#X connect 29 0 85 0;
#X connect 29 1 31 0;
#X connect 30 0 1 0;
#X connect 31 0 32 0;
#X connect 31 0 77 0;
#X connect 31 1 35 0;
#X connect 32 0 33 0;
#X connect 33 0 34 0;
#X connect 34 0 1 0;
#X connect 35 0 65 0;
#X connect 35 0 77 0;
#X connect 35 1 36 0;
#X connect 36 0 37 0;
#X connect 36 0 77 0;
#X connect 36 1 39 0;
#X connect 37 0 38 0;
#X connect 38 0 67 0;
#X connect 39 0 40 0;
#X connect 39 0 78 0;
#X connect 39 1 42 0;
#X connect 40 0 41 0;
#X connect 41 0 62 0;
#X connect 42 0 43 0;
#X connect 42 1 46 0;
#X connect 43 0 44 0;
#X connect 44 0 45 0;
#X connect 45 0 1 0;
#X connect 46 0 72 0;
#X connect 46 0 82 0;
#X connect 46 1 83 0;
#X connect 47 0 48 0;
#X connect 48 0 49 0;
#X connect 49 0 1 0;
#X connect 50 0 1 0;
#X connect 51 0 91 0;
#X connect 51 0 93 0;
#X connect 52 0 1 0;
```

```
#X connect 54 0 53 0;
#X connect 55 0 58 0;
#X connect 57 0 8 0;
#X connect 57 0 59 0;
#X connect 57 1 56 0;
#X connect 57 1 71 0;
#X connect 58 0 54 0;
#X connect 59 0 53 0;
#X connect 60 0 53 0;
#X connect 61 0 1 0;
#X connect 62 0 1 0;
#X connect 63 0 64 0;
#X connect 64 0 68 0;
#X connect 65 0 66 0;
#X connect 66 0 69 0;
#X connect 67 0 1 0;
#X connect 68 0 1 0;
#X connect 69 0 1 0;
#X connect 70 0 1 0;
#X connect 71 0 1 0;
#X connect 72 0 73 0;
#X connect 73 0 75 0;
#X connect 74 0 1 0;
#X connect 75 0 1 0;
#X connect 77 0 76 0;
#X connect 78 0 76 0;
#X connect 79 0 76 0;
#X connect 80 0 76 0;
#X connect 81 0 76 0;
#X connect 82 0 76 0;
#X connect 83 0 85 0;
#X connect 83 1 86 0;
#X connect 85 0 76 0;
#X connect 86 0 47 0;
#X connect 86 0 85 0;
#X connect 86 1 87 0;
#X connect 87 0 50 0;
#X connect 87 0 85 0;
#X connect 87 1 88 0;
#X connect 88 0 63 0;
#X connect 88 0 77 0;
#X connect 88 1 89 0;
#X connect 89 0 51 0;
#X connect 89 0 77 0;
#X connect 89 1 95 0;
#X connect 90 0 55 0;
#X connect 90 0 80 0;
#X connect 91 0 52 0;
#X connect 92 0 7 0;
#X connect 92 0 84 0;
#X connect 93 0 74 0;
#X connect 94 0 96 0;
#X connect 95 0 77 0;
#X connect 95 0 94 0;
#X connect 95 1 90 0;
#X connect 96 0 97 0;
#X connect 97 0 53 0;
```

```
#X restore 12 187 pd \$0-sequencer;
#X obj 48 70 t b b;
#X obj 142 461 writesf~ 2;
#X msg 208 82 \; pd dsp 1;
#X obj 207 61 delay 1000;
#X obj 198 118 delay 1000;
#X msg 127 394 stop;
#X obj 150 488 delay 1000;
#X msg 150 528 \; pd quit;
#X obj 150 507 spigot;
#X obj 145 7 table \$0-breakbeat-l;
#X obj 145 28 table \$0-breakbeat-r;
#N canvas 0 0 450 590 \$0-breaks 0;
#X obj 105 17 inlet;
#X obj 24 495 outlet~;
#X obj 123 498 outlet~;
#X obj 21 67 mod 2;
#X obj 21 88 sel 0;
#X obj 20 138 v \$0-tempo;
#X obj 20 159 * 2;
#X msg 20 180 0 \, 1 \$1;
#X obj 20 201 vline~;
#X obj 20 222 /~ 16;
#X obj 20 243 +~;
#X obj 20 264 wrap~;
#X obj 20 285 *~ 0;
#X obj 21 37 t f f;
#X obj 97 96 * 5;
#X obj 97 117 mod 16;
#X obj 97 75 mod 32;
#X obj 97 204 vline~;
#X obj 97 178 / 16;
#X obj 179 202 v \$0-breakbeat-len;
#X obj 21 115 t b b;
#X obj 33 331 tabread4~ \$0-breakbeat-l;
#X obj 133 351 tabread4~ \$0-breakbeat-r;
#X obj 80 263 +~;
#X obj 80 284 wrap~;
#X obj 80 305 *~ 0;
#X obj 25 462 expr~ tanh($v1);
#X obj 125 462 expr~ tanh($v1);
#X obj 23 371 tabread4~ \$0-breakbeat-r;
#X obj 123 391 tabread4~ \$0-breakbeat-l;
#X obj 80 242 /~ 8;
#X obj 126 413 *~ 32;
#X obj 26 413 *~ 32;
#X obj 64 15 inlet;
#X obj 64 36 route breaks;
#X obj 81 436 vline~;
#X obj 124 479 *~;
#X obj 24 480 *~;
#X obj 126 437 hip~ 100;
#X obj 24 437 hip~ 100;
#X connect 0 0 13 0;
#X connect 3 0 4 0;
#X connect 4 0 20 0;
#X connect 5 0 6 0;
```

```
#X connect 6 0 7 0;
#X connect 7 0 8 0;
#X connect 8 0 9 0;
#X connect 8 0 30 0;
#X connect 9 0 10 0;
#X connect 10 0 11 0;
#X connect 11 0 12 0;
#X connect 12 0 28 0;
#X connect 12 0 29 0;
#X connect 13 0 3 0;
#X connect 13 1 16 0;
#X connect 14 0 15 0;
#X connect 15 0 18 0;
#X connect 16 0 14 0;
#X connect 17 0 10 1;
#X connect 17 0 23 1;
#X connect 18 0 17 0;
#X connect 19 0 12 1;
#X connect 19 0 25 1;
#X connect 20 0 5 0;
#X connect 20 1 19 0;
#X connect 21 0 32 0;
#X connect 22 0 31 0;
#X connect 23 0 24 0;
#X connect 24 0 25 0;
#X connect 25 0 21 0;
#X connect 25 0 22 0;
#X connect 26 0 37 0;
#X connect 27 0 36 0;
#X connect 28 0 32 0;
#X connect 29 0 31 0;
#X connect 30 0 23 0;
#X connect 31 0 38 0;
#X connect 32 0 39 0;
#X connect 33 0 34 0;
#X connect 34 0 35 0;
#X connect 35 0 37 1;
#X connect 35 0 36 1;
#X connect 36 0 2 0;
#X connect 37 0 1 0;
#X connect 38 0 27 0;
#X connect 39 0 26 0;
#X restore 78 241 pd \$0-breaks;
#N canvas 0 0 450 300 \$0-flanger 0;
#X obj 22 16 inlet;
#X obj 83 14 inlet~;
#X obj 258 13 inlet~;
#X obj 80 241 delwrite~ \$0-flange-l 1000;
#X obj 243 234 delwrite~ \$0-flange-r 1000;
#X obj 95 174 vd~ \$0-flange-l;
#X obj 79 217 +~;
#X obj 245 211 +~;
#X obj 260 163 vd~ \$0-flange-r;
#X obj 159 125 vline~;
#X obj 158 48 mtof;
#X obj 158 85 swap 1000;
#X obj 158 106 /;
```

```
#X obj 19 44 mod 32;
#X obj 19 103 tabread \$0-flangeline;
#X obj 19 124 sel -1;
#X obj 158 29 tabread \$0-scale;
#X obj 99 264 outlet~;
#X obj 284 269 outlet~;
#X obj 157 146 lop~ 25;
#X obj 260 89 v \$0-tempo;
#X obj 284 132 vline~;
#X obj 260 186 *~;
#X obj 95 195 *~;
#X obj 88 49 div 16;
#X obj 88 70 + 1;
#X obj 20 71 t f f;
#X obj 159 67 *;
#X obj 262 66 bang;
#X msg 259 109 -0.95 \, -0.85 \$1;
#X connect 0 0 13 0;
#X connect 0 0 28 0;
#X connect 1 0 6 0;
#X connect 2 0 7 0;
#X connect 5 0 23 0;
#X connect 6 0 3 0;
#X connect 6 0 17 0;
#X connect 7 0 4 0;
#X connect 7 0 18 0;
#X connect 8 0 22 0;
#X connect 9 0 19 0;
#X connect 10 0 27 0;
#X connect 11 0 12 0;
#X connect 11 1 12 1;
#X connect 12 0 9 0;
#X connect 13 0 26 0;
#X connect 14 0 15 0;
#X connect 15 1 16 0;
#X connect 16 0 10 0;
#X connect 19 0 8 0;
#X connect 19 0 5 0;
#X connect 20 0 29 0;
#X connect 21 0 22 1;
#X connect 21 0 23 1;
#X connect 22 0 7 1;
#X connect 23 0 6 1;
#X connect 24 0 25 0;
#X connect 25 0 27 1;
#X connect 26 0 14 0;
#X connect 26 1 24 0;
#X connect 27 0 11 0;
#X connect 28 0 20 0;
#X connect 29 0 21 0;
#X restore 69 268 pd \$0-flanger;
#X obj 289 215 table \$0-flangeline 32;
#X obj 149 429 spigot;
#X obj 227 422 loadbang;
#X obj 227 443 v render;
#N canvas 0 0 679 421 \$0-cuesheet 0;
#X obj 92 17 inlet;
```

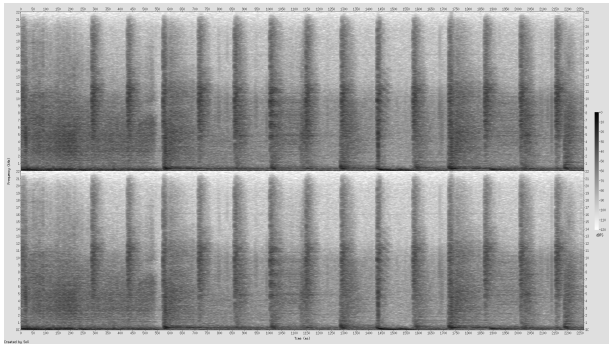
```
#X obj 188 15 inlet;
#X obj 15 15 inlet;
#X obj 188 75 f 0;
#X obj 224 74 + 1;
#X obj 279 14 loadbang;
#X msg 279 35 1000 75;
#X obj 279 55 /;
#X obj 188 54 metro 13.3333;
#X obj 188 96 t f f f f;
#X obj 239 125 mod 75;
#X obj 239 146 makefilename %02d;
#X obj 197 124 div 75;
#X obj 197 145 mod 60;
#X obj 197 166 makefilename %02d;
#X obj 142 124 div 4500;
#X obj 142 189 makefilename %02d;
#X obj 140 355 textfile;
#X obj 401 33 loadbang;
#X msg 230 287 clear \, rewind \, add PERFORMER "dynamo.pd" \, add
TITLE "dynamo.pd #\$1" \, add FILE "dynamo.pd.\$1.wav" WAVE;
#X obj 401 55 v seed;
#X obj 401 76 makefilename %02d;
#X obj 92 41 bang;
#X obj 188 36 bang;
#X obj 92 62 f 1;
#X obj 147 63 + 1;
#X obj 86 168 makefilename %02d;
#X msg 12 296 write dynamo.pd.\$1.cue cr;
#X obj 15 43 bang;
#X obj 13 251 symbol;
#X obj 29 348 outlet;
#X obj 23 140 print track;
#X obj 28 99 sel 16;
#X obj 418 135 makefilename %02d;
#X msg 419 112 1;
#X msg 62 273 add INDEX \$6 \$2:\$3:\$4;
#X obj 93 93 t f f;
#X obj 176 234 == 1;
#X obj 120 234 t a a a;
#X obj 141 254 spigot;
#X obj 119 214 pack s s s s s s s;
#X obj 418 185 makefilename %02d;
#X msg 419 162 0;
#X msg 141 315 add INDEX \$7 00:00:00;
#X msg 198 252 add TRACK \$1 AUDIO \, add FLAGS DCP \, add PERFORMER
"dynamo.pd" \, add TITLE "dynamo.pd #\$5/\$1";
#X connect 0 0 22 0;
#X connect 1 0 23 0;
#X connect 2 0 28 0;
#X connect 3 0 4 0;
#X connect 3 0 9 0;
#X connect 4 0 3 1;
#X connect 5 0 6 0;
#X connect 6 0 7 0;
#X connect 7 0 8 1;
#X connect 8 0 3 0;
#X connect 9 1 15 0;
```

```
#X connect 9 2 12 0;
#X connect 9 3 10 0;
#X connect 10 0 11 0;
#X connect 11 0 40 3;
#X connect 12 0 13 0;
#X connect 13 0 14 0;
#X connect 14 0 40 2;
#X connect 15 0 16 0;
#X connect 16 0 40 1;
#X connect 18 0 20 0;
#X connect 18 0 34 0;
#X connect 18 0 42 0;
#X connect 19 0 17 0;
#X connect 20 0 21 0;
#X connect 21 0 19 0;
#X connect 21 0 29 1;
#X connect 21 0 40 4;
#X connect 22 0 24 0;
#X connect 23 0 8 0;
#X connect 24 0 25 0;
#X connect 24 0 31 0;
#X connect 24 0 32 0;
#X connect 24 0 36 0;
#X connect 25 0 24 1;
#X connect 26 0 40 0;
#X connect 27 0 17 0;
#X connect 28 0 29 0;
#X connect 29 0 27 0;
#X connect 32 0 30 0;
#X connect 33 0 40 5;
#X connect 34 0 33 0;
#X connect 35 0 17 0;
#X connect 36 0 26 0;
#X connect 36 1 37 0;
#X connect 37 0 39 1;
#X connect 38 0 35 0;
#X connect 38 1 39 0;
#X connect 38 2 44 0;
#X connect 39 0 43 0;
#X connect 40 0 38 0;
#X connect 41 0 40 6;
#X connect 42 0 41 0;
#X connect 43 0 17 0;
#X connect 44 0 17 0;
#X restore 12 449 pd \$0-cuesheet;
#X obj 12 381 list prepend 0;
#X obj 14 402 route 0 1;
#X msg 11 479 1;
#X obj 39 423 t b b;
#X obj 272 342 v seed;
#X obj 272 363 makefilename %02d;
#X msg 272 385 open -bytes 2 dynamo.pd.\$1.wav \, start;
#X obj 7 424 bang;
#X obj 12 501 spigot;
#X obj 12 360 spigot;
#X obj 93 428 spigot;
#X obj 487 520 outlet;
```

```
#X obj 64 143 t f f f f;
#X obj 602 520 outlet;
#X obj 662 521 outlet;
#X connect 1 0 2 1;
#X connect 1 0 26 0;
#X connect 2 0 23 0;
#X connect 2 1 15 0;
#X connect 3 0 2 1;
#X connect 6 0 53 0;
#X connect 7 0 18 10;
#X connect 7 1 12 4;
#X connect 9 0 18 1;
#X connect 9 1 18 2;
#X connect 9 2 18 3;
#X connect 9 3 18 4;
#X connect 9 4 18 5;
#X connect 9 5 18 6;
#X connect 9 6 18 7;
#X connect 11 1 20 1;
#X connect 11 2 16 1;
#X connect 11 3 12 2;
#X connect 11 4 12 3;
#X connect 11 5 7 2;
#X connect 11 6 7 3;
#X connect 11 7 7 4;
#X connect 12 0 18 11;
#X connect 12 1 18 12;
#X connect 15 0 6 1;
#X connect 15 1 12 1;
#X connect 15 1 22 0;
#X connect 16 0 18 13;
#X connect 16 1 18 14;
#X connect 18 0 8 0;
#X connect 18 0 24 0;
#X connect 18 1 8 1;
#X connect 18 1 24 1;
#X connect 20 0 18 15;
#X connect 20 1 18 16;
#X connect 22 0 2 0;
#X connect 22 0 50 0;
#X connect 22 1 18 0;
#X connect 22 1 34 0;
#X connect 22 2 7 0;
#X connect 22 2 12 0;
#X connect 22 3 54 0;
#X connect 23 0 6 0;
#X connect 23 1 12 1;
#X connect 26 0 25 0;
#X connect 26 0 27 0;
#X connect 27 0 45 0;
#X connect 27 0 51 0;
#X connect 28 0 37 0;
#X connect 28 0 55 0;
#X connect 29 0 31 0;
#X connect 31 0 30 0;
#X connect 34 0 35 1;
#X connect 34 1 35 2;
```

```
#X connect 35 0 18 8;
#X connect 35 1 18 9;
#X connect 37 0 24 0;
#X connect 38 0 39 0;
#X connect 39 0 37 1;
#X connect 39 0 31 1;
#X connect 39 0 49 1;
#X connect 39 0 50 1;
#X connect 39 0 51 1;
#X connect 40 0 43 0;
#X connect 41 0 42 0;
#X connect 42 0 48 0;
#X connect 42 1 44 0;
#X connect 43 0 49 0;
#X connect 44 0 40 0;
#X connect 44 1 29 0;
#X connect 44 1 28 0;
#X connect 45 0 46 0;
#X connect 46 0 47 0;
#X connect 47 0 37 0;
#X connect 47 0 55 0;
#X connect 48 0 40 1;
#X connect 49 0 41 1;
#X connect 50 0 41 0;
#X connect 51 0 40 2;
#X connect 53 0 16 0;
#X connect 53 0 7 1;
#X connect 53 0 20 0;
#X connect 53 0 9 0;
#X connect 53 0 34 1;
#X connect 53 0 35 0;
#X connect 53 1 11 0;
#X connect 53 2 22 1;
#X connect 53 3 52 0;
```

3 pdgem/breakbeat.wav



4 pdgem/complex-mod~.pd

```
#N canvas 206 108 428 341 12;
#X obj 142 87 inlet~;
#X obj 315 166 cos~;
#X obj 351 144 +~ -0.25;
```

```
#X obj 351 166 cos~;
#X obj 225 87 inlet~;
#X obj 142 215 *~;
#X obj 225 216 *~;
#X obj 142 251 -~;
#X obj 142 284 outlet~;
#X obj 212 285 outlet~;
#X obj 212 252 +~;
#X text 140 310 positive;
#X text 213 311 negative;
#X obj 315 114 phasor~;
#X obj 315 88 inlet~;
#X connect 0 0 5 0;
#X connect 1 0 5 1;
#X connect 2 0 3 0;
#X connect 3 0 6 1;
#X connect 4 0 6 0;
#X connect 5 0 7 0;
#X connect 5 0 10 0;
#X connect 6 0 7 1;
#X connect 6 0 10 1;
#X connect 7 0 8 0;
#X connect 10 0 9 0;
#X connect 13 0 2 0;
#X connect 13 0 1 0;
#X connect 14 0 13 0;
```

5 pdgem/fractaloids.frag

```
uniform vec2 roots[8];
uniform vec2 powers[8];
uniform float hues[8];
uniform int activeRoots;
uniform float epsSquared;
uniform sampler2D hatch;

float cmag2(vec2 z) { return dot(z,z); }
vec2 crecip(vec2 z) { float d = cmag2(z); return z / vec2(d, -d); }
vec2 cmul(vec2 a, vec2 b) { return vec2(a.x * b.x - a.y * b.y, a.x * b.y + a.y * b.x); }

vec2 newtonStep(vec2 z) {
  vec2 s = vec2(0.0);
  for (int i = 0; i < 8; ++i) {
    if (i < activeRoots) {
      s += cmul(powers[i], crecip(z - roots[i]));
    }
  }
  return z - crecip(s);
}

vec4 newtonDelta(vec2 z00) {
  vec2 z0 = z00;
  bool done = false;
  int count = -1;
  float root = -1.0;
  float d0 = 1000.0;
```

```
  float d1 = 1000.0;
  vec2 z1 = z00;
  for (int n = 0; n < 64; ++n) {
    d1 = d0;
    z0 = newtonStep(z0);
    d0 = 1000.0;
    for (int i = 0; i < 8; ++i) {
      if (i < activeRoots) {
        float d = cmag2(z0 - roots[i].xy);
        if (d < d0) { d0 = d; root = hues[i]; z1 = roots[i]; }
      }
    }
    if (d0 < epsSquared) {
      done = true;
      count = n;
      break;
    }
  }
  float delta = -1.0;
  if (done) {
    delta = float(count);
    if (d0 > 0.0) {
      delta += clamp(log(epsSquared / d1) / log(d0 / d1), 0.0, 1.0);
    }
  }
  return vec4(root, delta, z1);
}

vec3 yuv2rgb(vec3 yuv) {
  return yuv * mat3(1.0, 1.407, 0.0, 1.0, -0.677, -0.236, 1.0, 0.0, 1.848);
}

vec3 splat(vec3 rgb) {
  float m = max(max(max(rgb.x, rgb.y), rgb.z), 1.0);
  return clamp(rgb / m, 0.0, 1.0);
}

void main(void) {
  vec2 z = gl_TexCoord[0].xy;
  vec4 d = newtonDelta(z);
  vec3 c = vec3(0.0);
  if (!(d.x < 0.0 || d.y < 0.0)) {
    vec2 w = (z - d.zw) * 2.0 + vec2(0.5);
    w -= floor(w);
    w /= vec2(4.0, 2.0);
    float p = d.x / 8.0;
    p -= floor(p);
    vec2 k = vec2(2.0 * p, p < 0.5 ? 0.0 : 0.5);
    k -= floor(k);
    float ds = texture2D(hatch, w + k).g;
    float y = pow(clamp(2.0 / (1.0 + pow(d.y, 0.5)) - 2.0/pow(64.0, 0.5), 0.0, ↙
        ↳ 1.0), 0.5);
    float r = clamp(0.2 - 0.05 * ds, 0.0, 1.0) * y;
    float t = d.x * 3.883222077450933;
    c = splat(yuv2rgb(vec3((0.9 + 0.1 * ds) * y, r * cos(t), r * sin(t))));
  }
  gl_FragColor = vec4(c, 1.0);
```

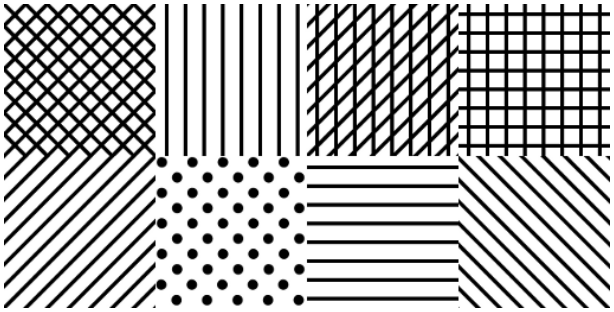
```
}
```

6 pdgem/fractaloids.pd

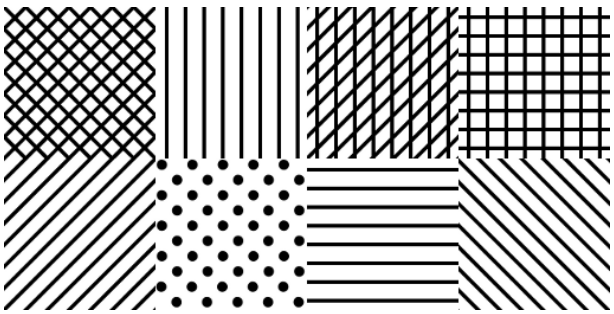
```
#N canvas 0 0 714 270 10;
#X obj 13 41 inlet;
#X obj 13 190 glsl_program;
#X obj 323 43 inlet;
#X text 319 25 roots;
#X obj 321 65 t a a;
#X msg 49 139 link \$1;
#X obj 13 216 outlet;
#X text 14 234 gemlist;
#X text 12 17 gemlist;
#X text 115 19 epsSquared;
#X obj 115 38 inlet;
#X msg 115 137 epsSquared \$1;
#X obj 344 91 list length;
#X msg 331 151 activeRoots \$1;
#X obj 13 96 glsl_fragment;
#X msg 24 72 open fractaloids.frag;
#X obj 211 29 loadbang;
#X msg 169 75 1e-06;
#X msg 253 73 0;
#X obj 299 193 list trim;
#X obj 299 171 list prepend roots[0];
#X obj 115 112 f;
#X obj 331 131 f;
#X obj 299 151 list;
#X obj 13 117 t a b;
#X obj 344 112 div 2;
#X obj 462 46 inlet;
#X text 457 25 powers;
#X obj 441 196 list trim;
#X obj 441 154 list;
#X obj 437 175 list prepend powers[0];
#X obj 602 46 inlet;
#X obj 581 196 list trim;
#X obj 581 154 list;
#X obj 577 175 list prepend hues[0];
#X msg 483 118 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0;
#X text 597 25 hues;
#X connect 0 0 14 0;
#X connect 1 0 6 0;
#X connect 2 0 4 0;
#X connect 4 0 23 1;
#X connect 4 1 12 0;
#X connect 5 0 1 0;
#X connect 10 0 21 1;
#X connect 11 0 1 0;
#X connect 12 0 25 0;
#X connect 13 0 1 0;
#X connect 14 0 24 0;
#X connect 14 1 5 0;
#X connect 15 0 14 0;
#X connect 16 0 15 0;
#X connect 16 0 17 0;
```

```
#X connect 16 0 18 0;
#X connect 16 0 35 0;
#X connect 17 0 21 1;
#X connect 18 0 22 1;
#X connect 19 0 1 0;
#X connect 20 0 19 0;
#X connect 21 0 11 0;
#X connect 22 0 13 0;
#X connect 23 0 20 0;
#X connect 24 0 1 0;
#X connect 24 1 21 0;
#X connect 24 1 22 0;
#X connect 24 1 23 0;
#X connect 24 1 29 0;
#X connect 24 1 33 0;
#X connect 25 0 22 1;
#X connect 26 0 29 1;
#X connect 28 0 1 0;
#X connect 29 0 30 0;
#X connect 30 0 28 0;
#X connect 31 0 33 1;
#X connect 32 0 1 0;
#X connect 33 0 34 0;
#X connect 34 0 32 0;
#X connect 35 0 29 1;
```

7 pdgem/hatching.png



8 pdgem/hatching.tif



9 pdgem/hilbert~.pd

```
#N canvas 269 0 593 306 12;
#X obj 105 92 biquad~ 0.83774 -0.06338 0.06338 -0.83774 1;
#X obj 105 66 biquad~ 1.94632 -0.94657 0.94657 -1.94632 1;
#X obj 86 149 biquad~ -0.02569 0.260502 -0.260502 0.02569 1;
#X obj 86 175 biquad~ 1.8685 -0.870686 0.870686 -1.8685 1;
#X obj 86 39 inlet~;
#X obj 105 121 outlet~;
#X obj 86 202 outlet~;
#X text 34 232 This is a pair of 4th-order all-pass filters whose outputs
somehow manage to be about 90 degrees out of phase from each other.
Both have different phases from the original. Adapted from a 4X patch
by Emmanuel Favreau \, circa 1982;
#X obj 502 39 inlet;
#X text 461 13 bang to clear;
#X text 80 16 signal in;
#X msg 502 112 clear;
#X connect 0 0 5 0;
#X connect 1 0 0 0;
#X connect 2 0 3 0;
#X connect 3 0 6 0;
#X connect 4 0 1 0;
#X connect 4 0 2 0;
#X connect 8 0 11 0;
#X connect 11 0 1 0;
#X connect 11 0 0 0;
#X connect 11 0 2 0;
#X connect 11 0 3 0;
```

10 pdgem/jugglee-help.pd

```
#N canvas 0 0 450 414 10;
#X obj 242 126 jugglee;
#X obj 242 79 juggler;
#X obj 324 137 print juggler;
#X obj 242 41 t b b;
#X obj 300 22 loadbang;
#X obj 77 23 f 0;
#X obj 108 40 moses 1;
#X obj 148 83 - 1;
#X obj 148 61 t f b;
#X msg 296 183 create \, 1;
#X msg 311 210 0 \, destroy;
#X obj 16 23 gemhead;
#X obj 17 95 t b a;
#X obj 13 134 gemlist;
#X obj 242 150 route ball;
#X obj 242 171 t b a;
#X obj 111 197 unpack f f f;
#X obj 109 282 pack f 1 1;
#X obj 109 305 Gem/hsv2rgb;
#X obj 111 239 mod 360;
#X obj 111 260 / 360;
#X obj 180 246 pack f f 0;
#X obj 15 239 translate;
#X obj 15 336 translate;
#X obj 49 163 loadbang;
#X msg 48 307 -1;
```

```
#X msg 41 187 1;
#X obj 15 260 color;
#X obj 15 281 sphere 0.1;
#X obj 111 218 * 222.492;
#X obj 201 223 - 3;
#X obj 324 112 spigot;
#X obj 356 86 tgl 15 0 empty empty empty 17 7 0 10 -262144 -1 -1 0
1;
#X obj 172 224 * 2;
#X obj 199 203 moses 0;
#X msg 300 47 balls 5 \, throw 7;
#X obj 234 225 / 2;
#X obj 107 20 + 0.1;
#X obj 296 236 gemwin 25;
#X connect 0 0 14 0;
#X connect 1 0 0 0;
#X connect 1 0 31 0;
#X connect 3 0 1 0;
#X connect 3 1 0 0;
#X connect 4 0 35 0;
#X connect 5 0 37 0;
#X connect 6 0 5 1;
#X connect 6 0 0 0;
#X connect 6 1 8 0;
#X connect 7 0 5 0;
#X connect 8 0 7 0;
#X connect 8 1 3 0;
#X connect 9 0 38 0;
#X connect 10 0 38 0;
#X connect 11 0 12 0;
#X connect 12 0 5 0;
#X connect 12 1 13 1;
#X connect 13 0 22 0;
#X connect 14 0 15 0;
#X connect 15 0 13 0;
#X connect 15 1 16 0;
#X connect 16 0 29 0;
#X connect 16 1 33 0;
#X connect 16 2 34 0;
#X connect 17 0 18 0;
#X connect 18 0 27 1;
#X connect 19 0 20 0;
#X connect 20 0 17 0;
#X connect 21 0 22 2;
#X connect 21 0 23 2;
#X connect 22 0 27 0;
#X connect 24 0 25 0;
#X connect 24 0 26 0;
#X connect 25 0 23 1;
#X connect 26 0 22 1;
#X connect 27 0 28 0;
#X connect 28 0 23 0;
#X connect 29 0 19 0;
#X connect 30 0 21 1;
#X connect 31 0 2 0;
#X connect 32 0 31 1;
#X connect 33 0 21 0;
```

```
#X connect 34 0 30 0;
#X connect 34 1 36 0;
#X connect 35 0 1 0;
#X connect 36 0 30 0;
#X connect 37 0 6 0;
```

11 pdgem/jugglee.pd_lua

```
local J = pd.Class:new():register("jugglee")

function J:initialize()
  self.balls = { }
  self.currentBeat = 0
  self.currentHand = 1
  self.throwTime = 0.2
  self.catchTime = 0.2
  self.handRadius = 0.2
  self.inlets = 1
  self.outlets = 1
  return true
end

local function position(hands, ball, t)
  if (ball.t1 - ball.t0) == 1 then
    if     t < ball.t0 then
      return { 0, -1 }
    elseif t < ball.t1 then
      return { ball.x0 + (ball.x1 - ball.x0) * (t - ball.t0), -hands.handRadius }
    else
      return { 0, -1 }
    end
  else
    if     t < ball.t0 then
      return { 0, -1 }
    elseif t < ball.t0 + hands.throwTime then
      local d = 1
      if ball.x0 < 0 then d = -1 end
      local r = hands.handRadius
      local a = math.pi/2 * (t - ball.t0) / hands.throwTime
      return { ball.x0 + d * r * (1 - math.sin(a)), -r * math.cos(a) }
    elseif t < ball.t1 - hands.catchTime then
      local throw = (ball.t1 - hands.catchTime) - (ball.t0 + hands.throwTime)
      local dt = (t - (ball.t0 + hands.throwTime))
      return { ball.x0 + (ball.x1 - ball.x0) * dt / throw, throw * dt - dt * dt }
    elseif t < ball.t1 then
      local d = 1
      if ball.x1 < 0 then d = -1 end
      local r = hands.handRadius
      local a = math.pi/2 * (ball.t1 - t) / hands.catchTime
      return { ball.x1 - d * r * (1 - math.sin(a)), -r * math.cos(a) }
    else
      return { 0, -1 }
    end
  end
end
```

```
function J:in_1_bang()
  self.currentBeat = self.currentBeat + 1
  self.currentHand = -self.currentHand
  local t = self.currentBeat + 0.5
  local balls = { }
  local i
  for i = 1,#(self.balls) do
    local ball = self.balls[i]
    local p = position(self, ball, t)
    if p[2] > 0 then
      table.insert(balls, ball)
    end
  end
  self.balls = balls
end

function J:in_1_float(phase)
  local t = self.currentBeat + phase
  local i
  for i = 1,#(self.balls) do
    local ball = self.balls[i]
    local p = position(self, ball, t)
    local x = p[1]
    local y = p[2]
    self:outlet(1, "ball", { ball.id, x, y })
  end
end

function J:in_1_throw(atoms)
  local id        = atoms[1]
  local direction = atoms[2]
  local speed     = atoms[3]
  local fhand = self.currentHand
  local thand = self.currentHand + 2 * direction
  local dx = -direction
  local ball = { }
  ball.id = id
  ball.x0 = fhand * (1 - self.handRadius)
  ball.x1 = thand * (1 + self.handRadius)
  ball.t0 = self.currentBeat
  ball.t1 = self.currentBeat + speed
  table.insert(self.balls, ball)
end

function J:in_1_pick() end
function J:in_1_drop() end
function J:in_1_wait() end
function J:in_1_pause() end
function J:in_1_remove() end
```

12 pdgem/juggler-help.pd

```
#N canvas 0 0 450 300 10;
#X obj 105 131 juggler;
#X msg 139 75 balls 3 \, throw 5;
#X obj 107 152 print juggle;
```

```
#X obj 104 75 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
-1;
#X msg 166 96 balls 4;
#X msg 184 120 balls 8;
#X msg 260 102 throw 3;
#X msg 260 134 throw 10;
#X msg 204 163 balls 0;
#X connect 0 0 2 0;
#X connect 1 0 0 0;
#X connect 3 0 0 0;
#X connect 4 0 0 0;
#X connect 5 0 0 0;
#X connect 6 0 0 0;
#X connect 7 0 0 0;
#X connect 8 0 0 0;
```

13 pdgem/juggler.pd_lua

```
local J = pd.Class:new():register("juggler")

function J:initialize()
  self.currentHand = -1
  self.nextBallID = 0
  self.currentBallCount = 0
  self.targetBallCount = 0
  self.maximumThrow = 0
  self.longestThrow = 0
  self.futureLandings = { }
  self.inlets = 1
  self.outlets = 1
  return true
end

function J:in_1_balls(atoms)
  if type(atoms[1] == "number") then
    self.targetBallCount = math.max(0, math.floor(atoms[1]))
  end
end

function J:in_1_throw(atoms)
  if type(atoms[1] == "number") then
    self.maximumThrow = math.max(0, math.floor(atoms[1]))
    self.longestThrow = math.max(self.longestThrow, self.maximumThrow)
  end
end

function J:in_1_bang()
  -- catch a possible ball
  local caught = self.futureLandings[1]
  -- advance futures
  local n = self.longestThrow
  local i
  for i = 1,n do
    self.futureLandings[i] = self.futureLandings[i + 1]
  end
  -- check what we caught
  if type(caught) == "number" then
```

```
40      -- a ball
        if self.currentBallCount > self.targetBallCount then
          -- remove it
          self.currentBallCount = self.currentBallCount - 1
          self:outlet(1, "remove", { caught })
45      else
          -- collect possible throws without landing collisions
          local choices = { }
          for i = 1,self.maximumThrow do
            if type(self.futureLandings[i]) ~= "number" then
50            table.insert(choices, i)
            end
          end
          -- choose a legal throw
          if #choices == 0 then
55          self.currentBallCount = self.currentBallCount - 1
            self:outlet(1, "drop", { caught })
          else
            local dice = math.random()
--[[--
60          dice = dice * dice
            if self.currentHand < 0 then
              dice = 1 - dice
            end
--]]--
65          dice = math.ceil(dice * #choices)
            dice = math.min(math.max(dice, 1), #choices)
            local throw = choices[dice]
            -- check direction of throw
            local direction
70          if (throw / 2) == math.floor(throw / 2) then
              direction = 0
            elseif self.currentHand < 0 then
              direction = 1
            else
75            direction = -1
            end
            -- schedule future landing
            self.futureLandings[throw] = caught
            self:outlet(1, "throw", { caught, direction, throw })
80        end
        end
      else
        -- caught nothing
        if self.currentBallCount < self.targetBallCount then
85        -- collect possible throws without landing collisions
          local choices = { }
          for i = 1,self.maximumThrow do
            if type(self.futureLandings[i]) ~= "number" then
              table.insert(choices, i)
90          end
          end
          -- choose a legal throw
          if #choices == 0 then
            self:outlet(1, "wait", { })
95        else
            local choice = math.random(1, #choices)
```

```
        local throw = choices[choice]
        -- check direction of throw
        local direction
        if (throw / 2) == math.floor(throw / 2) then
          direction = 0
        elseif self.currentHand < 0 then
          direction = 1
        else
          direction = -1
        end
        -- introduce a new ball
        local ball = self.nextBallID
        self.nextBallID = self.nextBallID + 1
        self.currentBallCount = self.currentBallCount + 1
        -- schedule future landing
        self.futureLandings[throw] = ball
        self:outlet(1, "pick", { ball })
        self:outlet(1, "throw", { ball, direction, throw })
      end
    else
      -- no action needed
      self:outlet(1, "pause", { })
    end
  end
  -- swap hands
  self.currentHand = -self.currentHand
end
```

14 pdgem/lissajous-help.pd

```
#N canvas 0 0 450 300 10;
#X obj 26 68 gemhead;
#X obj 26 208 fractaloids;
#X obj 26 230 square 3;
#X msg 58 180 1e-06;
#X obj 58 155 loadbang;
#X obj 26 93 t a b;
#X obj 130 177 lissajous;
#X obj 197 147 + 1;
#X obj 197 121 hradio 15 1 0 8 empty empty empty 0 -8 0 10 -262144
-1 -1 0;
#X obj 197 101 hradio 15 1 0 8 empty empty empty 0 -8 0 10 -262144
-1 -1 0;
#X obj 166 146 + 1;
#X obj 197 81 hradio 15 1 0 8 empty empty empty 0 -8 0 10 -262144 -1
-1 0;
#X obj 137 149 + 1;
#X obj 71 94 f;
#X obj 245 233 gemwin;
#X msg 245 182 create \, 1;
#X msg 271 208 0 \, destroy;
#X obj 108 96 + 0.001;
#X connect 0 0 5 0;
#X connect 1 0 2 0;
#X connect 3 0 1 1;
#X connect 4 0 3 0;
#X connect 5 0 1 0;
```

```
#X connect 5 1 13 0;
#X connect 6 0 1 2;
#X connect 7 0 6 3;
#X connect 8 0 7 0;
#X connect 9 0 10 0;
#X connect 10 0 6 2;
#X connect 11 0 12 0;
#X connect 12 0 6 1;
#X connect 13 0 6 0;
#X connect 13 0 17 0;
#X connect 15 0 14 0;
#X connect 16 0 14 0;
#X connect 17 0 13 1;
```

15 pdgem/lissajous.pd

```
#N canvas 0 0 347 362 10;
#X obj 26 40 inlet;
#X text 27 19 phase;
#X obj 217 32 inlet;
#X text 218 9 count;
#X obj 92 33 inlet;
#X text 94 11 A;
#X obj 160 32 inlet;
#X text 164 12 B;
#X obj 122 229 / 3;
#X obj 121 250 + 0.5;
#X obj 122 207 sin;
#X obj 122 183 * 6.28319;
#X obj 154 83 /;
#X obj 139 106 +;
#X obj 26 61 t b f;
#X obj 26 89 t b b;
#X msg 50 115 set;
#X obj 26 137 t b b;
#X obj 217 53 max 1;
#X obj 261 18 loadbang;
#X msg 258 35 3;
#X obj 53 160 f;
#X obj 53 181 until;
#X obj 53 202 f;
#X obj 85 202 + 1;
#X obj 79 57 t b b f;
#X msg 120 275 add2 \$1;
#X msg 28 299 0.689704 0.799453 0.83257 0.572195 0.725009 0.290624
0.447979 0.166673 0.210171 0.293652 0.190578 0.57613 0.40396 0.801134
;
#X obj 28 320 outlet;
#X obj 81 94 f;
#X obj 94 120 *;
#X obj 110 95 f;
#X msg 198 32 2;
#X msg 131 32 1;
#X connect 0 0 14 0;
#X connect 2 0 18 0;
#X connect 4 0 29 1;
#X connect 6 0 31 1;
```

```
#X connect 8 0 9 0;
#X connect 9 0 26 0;
#X connect 10 0 8 0;
#X connect 11 0 10 0;
#X connect 12 0 13 0;
#X connect 13 0 30 1;
#X connect 14 0 15 0;
#X connect 14 1 13 1;
#X connect 15 0 17 0;
#X connect 15 1 16 0;
#X connect 16 0 27 0;
#X connect 17 0 27 0;
#X connect 17 1 21 0;
#X connect 18 0 12 1;
#X connect 18 0 21 1;
#X connect 19 0 20 0;
#X connect 19 0 32 0;
#X connect 19 0 33 0;
#X connect 20 0 18 0;
#X connect 21 0 22 0;
#X connect 22 0 23 0;
#X connect 23 0 24 0;
#X connect 23 0 25 0;
#X connect 24 0 23 1;
#X connect 25 0 29 0;
#X connect 25 1 31 0;
#X connect 25 2 12 0;
#X connect 26 0 27 0;
#X connect 27 0 28 0;
#X connect 29 0 30 0;
#X connect 30 0 11 0;
#X connect 31 0 30 0;
#X connect 32 0 31 1;
#X connect 33 0 29 1;
```

16 pdgem/loader.pd

```
#N canvas 0 0 450 300 10;
#X obj 20 23 r load;
#X obj 20 65 max 1;
#X obj 106 93 until;
#X msg 106 137 obj 10 10 random;
#X obj 20 89 t b f;
#X obj 79 167 s pd-\$0-loader;
#N canvas 0 0 450 300 \$0-loader 0;
#X restore 106 22 pd \$0-loader;
#X obj 20 114 t b b;
#X msg 47 137 loadbang;
#X msg 220 137 clear;
#X msg 20 228 \; pd open main.pd .;
#X obj 20 198 delay 1000;
#X obj 103 45 v render;
#X obj 104 65 v seed;
#X obj 20 44 unpack f f;
#X connect 0 0 14 0;
#X connect 1 0 4 0;
#X connect 1 0 13 0;
```

```
#X connect 2 0 3 0;
#X connect 3 0 5 0;
#X connect 4 0 7 0;
#X connect 4 1 2 0;
#X connect 7 0 11 0;
#X connect 7 1 8 0;
#X connect 8 0 5 0;
#X connect 9 0 5 0;
#X connect 11 0 10 0;
#X connect 14 0 1 0;
#X connect 14 1 12 0;
```

17 pdgem/main.pd

```
#N canvas 0 0 450 300 10;
#X obj 192 227 video;
#X obj 195 58 audio;
#X obj 195 102 sel 0;
#X obj 195 146 timer;
#X msg 195 167 0 \, 1 \$1;
#X obj 195 188 vline~;
#X obj 195 125 t b b b;
#X obj 268 186 tgl 15 0 empty empty empty 17 7 0 10 -262144 -1 -1 1
1;
#X obj 195 81 mod 4;
#X obj 268 157 loadbang;
#X obj 9 106 - 64;
#X obj 10 178 expr pow(0.525 \, pow($f1 \, 3));
#X obj 9 130 mod 1536;
#X obj 10 154 / 1536;
#X obj 14 220 hsl 128 15 0 1 0 0 empty empty empty -2 -8 0 10 -262144
-1 -1 3779 1;
#X connect 1 0 8 0;
#X connect 1 0 10 0;
#X connect 1 1 0 3;
#X connect 1 2 0 4;
#X connect 2 0 6 0;
#X connect 3 0 4 0;
#X connect 4 0 5 0;
#X connect 5 0 0 1;
#X connect 6 0 3 0;
#X connect 6 1 3 1;
#X connect 6 2 0 2;
#X connect 7 0 0 5;
#X connect 8 0 2 0;
#X connect 9 0 7 0;
#X connect 10 0 12 0;
#X connect 11 0 0 0;
#X connect 12 0 13 0;
#X connect 13 0 11 0;
#X connect 13 0 14 0;
```

18 pdgem/README

```
note: requires patched Gem to support uniform arrays
```

19 pdgem/start.sh

```
#!/bin/sh
r=1
n=`openssl rand 1 | hexdump -e "1/1 \"%1u\n\""`
pd -r 48000 -jack -channels 2 -path "${HOME}/opt/lib/pd/extra/Gem" -lib Gem:↙
    ↳ pdlua -open loader.pd -send "load $n $r"
```

20 pdgem/video.pd

```
#N canvas 34 20 824 443 10;
#X obj 248 113 jugglee;
#X obj 248 86 juggler;
#X obj 251 67 t b b;
#X msg 424 81 0 \, destroy;
#X obj 50 22 gemhead;
#X obj 248 137 route ball;
#X obj 248 157 unpack f f f;
#X obj 426 107 gemwin 25;
#X obj 53 249 fractaloids;
#X msg 84 216 1e-06;
#X obj 84 191 loadbang;
#X obj 50 47 t a b b b;
#X msg 142 194 set;
#X obj 316 193 moses 0;
#X obj 281 239 + 0.5;
#X obj 52 266 square 4;
#X obj 239 25 inlet;
#X obj 195 24 inlet~;
#X obj 195 45 snapshot~;
#X obj 636 21 inlet;
#X obj 424 39 sel 0 1;
#X obj 169 100 loadbang;
#X msg 140 261 4;
#X obj 140 282 / 9;
#X obj 140 303 * 16;
#X obj 316 215 / 4;
#X obj 327 239 + 0.3;
#X obj 282 191 / 4;
#X msg 381 225 balls \$1;
#X msg 446 224 throw \$1;
#X obj 52 149 pix_texture;
#X obj 52 124 pix_image hatching.tif;
#X msg 152 151 rectangle 0;
#X obj 52 170 pix_coordinate;
#X msg 148 173 0 0 1 0 1 1 0 1;
#X obj 360 19 inlet;
#X obj 379 165 +;
#X obj 442 166 +;
#X obj 377 131 route dballs dthrow balls throw;
#X msg 446 59 dimen 1024 576 \, create \, 1;
#X obj 579 17 inlet;
#X obj 580 93 symbol;
#X obj 580 114 sel start stop;
#X obj 52 364 pix_write;
#X obj 52 301 spigot;
```

```
#X obj 580 135 v seed;
#X obj 581 156 makefilename %02d;
#X msg 581 175 file dynamo.pd.\$1. 0 \, auto 1;
#X obj 52 326 t b a;
#X msg 620 198 auto 0;
#X msg 587 196 1;
#X obj 592 300 spigot;
#X obj 627 257 loadbang;
#X obj 627 278 v render;
#X msg 672 201 0;
#N canvas 0 0 450 300 \$0-collect 0;
#X obj 20 12 inlet;
#X msg 20 33 0.785573 0.343176 0.272566 0.49117 0.388845 0.662158 0.553015
0.685152 0.505124 0.66648 0.436736 0.606141 0.367665 0.31648;
#X obj 19 67 outlet;
#X obj 80 76 inlet;
#X msg 80 97 0.804513 0.142787 0.804513 -0.113717 0.804513 -0.0555773
0.804513 0.0265076 0.804513 0.00256219 0.804513 -0.031632 0.804513
-0.0661673;
#X obj 79 131 outlet;
#X obj 145 138 inlet;
#X msg 145 159 668 670 669 672 664 671 657;
#X obj 144 193 outlet;
#X connect 0 0 1 0;
#X connect 1 0 2 0;
#X connect 3 0 4 0;
#X connect 4 0 5 0;
#X connect 6 0 7 0;
#X connect 7 0 8 0;
#X restore 130 219 pd \$0-collect;
#X msg 279 287 add2 \$1 \$2;
#X obj 281 266 pack f f;
#X msg 278 355 add2 \$1;
#X obj 124 25 inlet;
#X obj 196 68 t f f;
#X obj 275 326 +;
#X obj 372 330 loadbang;
#X obj 372 351 random 1000;
#X obj 379 189 clip 2 8;
#X obj 211 373 pack f f;
#X msg 210 402 add2 \$2 \$1;
#X obj 215 348 * 0.125;
#X obj 444 191 clip 2 10;
#X obj 471 170 + 1;
#X obj 349 215 / 12;
#X connect 0 0 5 0;
#X connect 1 0 0 0;
#X connect 2 0 1 0;
#X connect 2 1 0 0;
#X connect 3 0 7 0;
#X connect 4 0 11 0;
#X connect 5 0 6 0;
#X connect 6 0 61 0;
#X connect 6 1 27 0;
#X connect 6 1 67 0;
#X connect 6 2 13 0;
#X connect 8 0 15 0;
```

```
#X connect 9 0 8 1;
#X connect 10 0 9 0;
#X connect 11 0 31 0;
#X connect 11 1 55 0;
#X connect 11 1 55 1;
#X connect 11 1 55 2;
#X connect 11 2 18 0;
#X connect 11 3 12 0;
#X connect 12 0 55 0;
#X connect 12 0 55 1;
#X connect 12 0 55 2;
#X connect 13 0 25 0;
#X connect 13 1 70 0;
#X connect 14 0 57 0;
#X connect 15 0 44 0;
#X connect 16 0 2 0;
#X connect 17 0 18 0;
#X connect 18 0 60 0;
#X connect 19 0 20 0;
#X connect 20 0 3 0;
#X connect 20 1 39 0;
#X connect 21 0 22 0;
#X connect 21 0 32 0;
#X connect 21 0 34 0;
#X connect 22 0 23 0;
#X connect 23 0 24 0;
#X connect 24 0 15 1;
#X connect 25 0 26 0;
#X connect 26 0 57 1;
#X connect 27 0 14 0;
#X connect 28 0 1 0;
#X connect 29 0 1 0;
#X connect 30 0 33 0;
#X connect 31 0 30 0;
#X connect 32 0 30 0;
#X connect 33 0 8 0;
#X connect 34 0 33 1;
#X connect 35 0 38 0;
#X connect 36 0 64 0;
#X connect 37 0 68 0;
#X connect 38 0 36 0;
#X connect 38 1 37 0;
#X connect 38 2 64 0;
#X connect 38 3 68 0;
#X connect 39 0 7 0;
#X connect 40 0 41 0;
#X connect 41 0 42 0;
#X connect 42 0 45 0;
#X connect 42 0 50 0;
#X connect 42 1 49 0;
#X connect 42 1 54 0;
#X connect 44 0 48 0;
#X connect 45 0 46 0;
#X connect 46 0 47 0;
#X connect 47 0 43 0;
#X connect 48 0 43 0;
#X connect 48 1 43 0;
```

```
#X connect 49 0 43 0;
#X connect 50 0 51 0;
#X connect 51 0 44 1;
#X connect 52 0 53 0;
#X connect 53 0 51 1;
#X connect 54 0 51 0;
#X connect 55 0 8 2;
#X connect 55 1 8 3;
#X connect 55 2 8 4;
#X connect 56 0 55 0;
#X connect 57 0 56 0;
#X connect 58 0 55 2;
#X connect 59 0 65 1;
#X connect 60 0 0 0;
#X connect 61 0 58 0;
#X connect 62 0 63 0;
#X connect 63 0 61 1;
#X connect 64 0 36 1;
#X connect 64 0 28 0;
#X connect 64 0 69 0;
#X connect 65 0 66 0;
#X connect 66 0 55 1;
#X connect 67 0 65 0;
#X connect 68 0 37 1;
#X connect 68 0 29 0;
#X connect 69 0 68 1;
#X connect 70 0 25 0;
```

21 webgl/agpl-3.0.txt

GNU AFFERO GENERAL PUBLIC LICENSE
Version 3, 19 November 2007

Preamble

The GNU Affero General Public License is a free, copyleft license for software and other kinds of works, specifically designed to ensure cooperation with the community in the case of network server software.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, our General Public Licenses are intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

Developers that use our General Public Licenses protect your rights

with two steps: (1) assert copyright on the software, and (2) offer you this License which gives you legal permission to copy, distribute and/or modify the software.

A secondary benefit of defending all users' freedom is that improvements made in alternate versions of the program, if they receive widespread use, become available for other developers to incorporate. Many developers of free software are heartened and encouraged by the resulting cooperation. However, in the case of software used on network servers, this result may fail to come about. The GNU General Public License permits making a modified version and letting the public access it on a server without ever releasing its source code to the public.

The GNU Affero General Public License is designed specifically to ensure that, in such cases, the modified source code becomes available to the community. It requires the operator of a network server to provide the source code of the modified version running there to the users of that server. Therefore, public use of a modified version, on a publicly accessible server, gives the public access to the source code of the modified version.

An older license, called the Affero General Public License and published by Affero, was designed to accomplish similar goals. This is a different license, not a version of the Affero GPL, but Affero has released a new version of the Affero GPL which permits relicensing under this license.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU Affero General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the

public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to

produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply

if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work

occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a

party means to make such an agreement or commitment not to enforce a
485 patent against the party.

If you convey a covered work, knowingly relying on a patent license,
and the Corresponding Source of the work is not available for anyone
to copy, free of charge and under the terms of this License, through a
490 publicly available network server or other readily accessible means,
then you must either (1) cause the Corresponding Source to be so
available, or (2) arrange to deprive yourself of the benefit of the
patent license for this particular work, or (3) arrange, in a manner
consistent with the requirements of this License, to extend the patent
495 license to downstream recipients. "Knowingly relying" means you have
actual knowledge that, but for the patent license, your conveying the
covered work in a country, or your recipient's use of the covered work
in a country, would infringe one or more identifiable patents in that
country that you have reason to believe are valid.
500
If, pursuant to or in connection with a single transaction or
arrangement, you convey, or propagate by procuring conveyance of, a
covered work, and grant a patent license to some of the parties
receiving the covered work authorizing them to use, propagate, modify
505 or convey a specific copy of the covered work, then the patent license
you grant is automatically extended to all recipients of the covered
work and works based on it.

A patent license is "discriminatory" if it does not include within
510 the scope of its coverage, prohibits the exercise of, or is
conditioned on the non-exercise of one or more of the rights that are
specifically granted under this License. You may not convey a covered
work if you are a party to an arrangement with a third party that is
in the business of distributing software, under which you make payment
515 to the third party based on the extent of your activity of conveying
the work, and under which the third party grants, to any of the
parties who would receive the covered work from you, a discriminatory
patent license (a) in connection with copies of the covered work
conveyed by you (or copies made from those copies), or (b) primarily
520 for and in connection with specific products or compilations that
contain the covered work, unless you entered into that arrangement,
or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting
525 any implied license or other defenses to infringement that may
otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

530 If conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License. If you cannot convey a
covered work so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you may
535 not convey it at all. For example, if you agree to terms that obligate you
to collect a royalty for further conveying from those to whom you convey
the Program, the only way you could satisfy both those terms and this
License would be to refrain entirely from conveying the Program.

540 13. Remote Network Interaction; Use with the GNU General Public License.

Notwithstanding any other provision of this License, if you modify the Program, your modified version must prominently offer all users interacting with it remotely through a computer network (if your version supports such interaction) an opportunity to receive the Corresponding Source of your version by providing access to the Corresponding Source from a network server at no charge, through some standard or customary means of facilitating copying of software. This Corresponding Source shall include the Corresponding Source for any work covered by version 3 of the GNU General Public License that is incorporated pursuant to the following paragraph.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the work with which it is combined will remain governed by version 3 of the GNU General Public License.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU Affero General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU Affero General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU Affero General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU Affero General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.>
> Copyright (C) <year> <name of author>
>
> This program is free software: you can redistribute it and/or modify it under the terms of the GNU Affero General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Affero General Public License for more details.
>
> You should have received a copy of the GNU Affero General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If your software can interact with users remotely through a computer network, you should also make sure that it provides a way for users to get its source. For example, if your program is a web application, its interface could display a "Source" link that leads users to an archive of the code. There are many ways you could offer source, and different

```
655 solutions will be better for different programs; see section 13 for the
    specific requirements.

      You should also get your employer (if you work as a programmer) or school,
    if any, to sign a "copyright disclaimer" for the program, if necessary.
660 For more information on this, and how to apply and follow the GNU AGPL, see
    <http://www.gnu.org/licenses/>.
```

22 webgl/fractaloids.html

```
    <!DOCTYPE html>
    <html><head><meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>fractaloids</title>
    <!--
 5
    fractaloids.html -- trippiness in your browser
    Copyright (C) 2012 Claude Heiland-Allen

    This program is free software: you can redistribute it and/or modify
10  it under the terms of the GNU Affero General Public License as
    published by the Free Software Foundation, either version 3 of the
    License, or (at your option) any later version.

    This program is distributed in the hope that it will be useful,
15  but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.

    You should have received a copy of the GNU Affero General Public License
20  along with this program.  If not, see <http://www.gnu.org/licenses/>.

    -->
    <style type="text/css">* { border: 0; padding: 0; margin: 0; } #framerate { text↙
        ↘ -align: right; }</style>
    <script type="text/javascript" src="webgl-utils.js"></script>
25  <script type="text/javascript" src="webgl-debug.js"></script>
    <script type="text/javascript" src="J3DI.js"></script>
    <script type="text/javascript" src="J3DIMath.js"></script>
    <script id="vert" type="x-shader/x-vertex">//<![CDATA[

30  uniform mat4 vMatrix;
    attribute vec2 vPosition;
    varying vec2 v_texCoord;
    void main() {
      gl_Position = vec4(vPosition, 0.0, 1.0);
35    v_texCoord = vec2(vMatrix * vec4(vPosition, 0.0, 1.0));
    }

    //]]></script>
    <script id="frag" type="x-shader/x-fragment">//<![CDATA[
40  precision mediump float;

    uniform vec2 roots[8];
    uniform float epsSquared;
    uniform int maxIters;
45  uniform int activeRoots;
```

```
varying vec2 v_texCoord;

float cmag2(vec2 z) { return dot(z,z); }
vec2 crecip(vec2 z) { float d = cmag2(z); return z / vec2(d, -d); }

vec2 newtonStep(vec2 z) {
  vec2 s = vec2(0.0);
  for (int i = 0; i < 8; ++i) {
    if (i < activeRoots) {
      s += crecip(z - roots[i]);
    }
  }
  return z - crecip(s);
}

vec2 newtonDelta(vec2 z00) {
  vec2 z0 = z00;
  bool done = false;
  int count = -1;
  int root = -1;
  float d0 = 1000.0;
  float d1 = 1000.0;
  for (int n = 0; n < 64; ++n) {
    d1 = d0;
    z0 = newtonStep(z0);
    d0 = 1000.0;
    for (int i = 0; i < 8; ++i) {
      if (i < activeRoots) {
        float d = cmag2(z0 - roots[i]);
        if (d < d0) { d0 = d; root = i; }
      }
    }
    if (d0 < epsSquared) {
      done = true;
      count = n;
      break;
    }
  }
  float delta = -1.0;
  if (done) {
    delta = float(count);
    if (d0 > 0.0) {
      delta += clamp(log(epsSquared / d1) / log(d0 / d1), 0.0, 1.0);
    }
  }
  return vec2(float(root),delta);
}

vec3 yuv2rgb(vec3 yuv) {
  return yuv * mat3(1.0, 1.407, 0.0, 1.0, -0.677, -0.236, 1.0, 0.0, 1.848);
}

vec3 splat(vec3 rgb) {
  float m = max(max(max(rgb.x, rgb.y), rgb.z), 1.0);
  return clamp(rgb / m, 0.0, 1.0);
}
```

```
    void main(void) {
105   vec2 z = v_texCoord.xy;
      vec2 d = newtonDelta(z);
      vec3 c = vec3(0.0);
      if (d.x >= 0.0 && d.y >= 0.0) {
        float y = clamp(1.0 / (1.0 + d.y), 0.0, 1.0);
110     float r = 0.7 * y;
        float t = d.x * 3.883222077450933;
        c = splat(yuv2rgb(vec3(y, r * cos(t), r * sin(t))));
      }
      gl_FragColor = vec4(c, 1.0);
115 }

    //]]></script><script id="main" type="text/javascript">//<![CDATA[

    // global state
120 var g = { };

    function now() { return new Date().getTime(); }

    function shader(gl) {
125   var vert = loadShader(gl, "vert");
      var frag = loadShader(gl, "frag");
      var prog = gl.createProgram();
      gl.attachShader(prog, vert);
      gl.attachShader(prog, frag);
130   gl.bindAttribLocation(prog, 0, "vPosition");
      gl.linkProgram(prog);
      var linked = gl.getProgramParameter(prog, gl.LINK_STATUS);
      if (!linked && !gl.isContextLost()) {
        gl.deleteProgram(prog);
135     gl.deleteProgram(frag);
        gl.deleteProgram(vert);
        return null;
      }
      gl.useProgram(prog);
140   return prog;
    }

    function init() {
      var gl = initWebGL("fractaloids");
145   WebGLDebugUtils.makeDebugContext(gl);
      if (gl) {
        g.maxRoots = 8;
        g.program = shader(gl);
        gl.uniform1f(gl.getUniformLocation(g.program, "epsSquared"), 1.0);
150     gl.uniform1i(gl.getUniformLocation(g.program, "maxIters"), 64);
        gl.uniform1i(gl.getUniformLocation(g.program, "activeRoots"), 0);
        g.points = new Float32Array([1,1, -1,1, -1,-1, 1,-1]);
        g.obj = gl.createBuffer()
        gl.enableVertexAttribArray(0);
155     gl.bindBuffer(gl.ARRAY_BUFFER, g.obj);
        gl.bufferData(gl.ARRAY_BUFFER, g.points, gl.STATIC_DRAW);
        gl.vertexAttribPointer(0, 2, gl.FLOAT, false, 0, 0);
        g.vMatrix = new J3DIMatrix4();
        return gl
160   }
```

```
  return null;
}

function reshape(gl) {
  var canvas = document.getElementById("fractaloids");
  var winW = window.innerWidth;
  var winH = window.innerHeight;
  if (winW == g.width && winH == g.height) {
    return;
  }
  g.width = winW;
  g.height = winH - 60;
  canvas.width = g.width;
  canvas.height = g.height;
  gl.viewport(0,0, g.width, g.height);
  g.vMatrix.makeIdentity();
  g.vMatrix.ortho(-1.0/g.width, 1.0/g.width, -1.0/g.height, 1.0/g.height, -1, 1)↙
    ↳ ;
  g.vMatrix.setUniform(gl, gl.getUniformLocation(g.program, "vMatrix"), false);
}

function render(gl) {
  reshape(gl);
  var t = now();
  var dt = t - g.frameTime;
  g.frameTime = t;
  rs = advance(dt);
  gl.uniform1i(gl.getUniformLocation(g.program, "activeRoots"), Math.min(rs.↙
    ↳ length / 2, 8));
  if (rs.length > 0) {
    gl.uniform2fv(gl.getUniformLocation(g.program, "roots"), rs);
  }
  gl.drawArrays(gl.TRIANGLE_FAN, 0, 4);
  g.framerate.snapshot();
}

function advance(dt) {
  var rs = [];
  for (var i = 0; i < g.loops.length; ++i) {
    var l = g.loops[i];
    if (l) {
      l.offset = l.offset + dt;
      while (l.offset > l.events[l.current].dt) {
        l.offset -= l.events[l.current].dt;
        l.current = (l.current + 1) % l.events.length;
      }
      var e0 = l.events[l.current];
      var e1 = l.events[(l.current + 1) % l.events.length];
      var t = l.offset / e0.dt;
      var x = e0.x * (1 - t) + t * e1.x;
      var y = e0.y * (1 - t) + t * e1.y;
      rs[rs.length] = x;
      rs[rs.length] = y;
    }
  }
  return rs;
}
```

```
function handleContextLost(e) {
  e.preventDefault();
  if (g.requestId !== undefined) {
    window.cancelAnimFrame(g.requestId);
    g.requestId = undefined;
  }
}

function handleContextRestored() {
  init();
  g.f();
}

function handleMouseDown(event) {
  event.preventDefault();
  g.mouseDown = true;
  g.mouseTime = now();
  g.mouseLoop = [];
}

function handleMouseUp(event) {
  event.preventDefault();
  g.mouseDown = false;
  if (g.mouseLoop.length > 1) {
    g.loops[g.nextloop] = { offset: 0, current: 0, events: g.mouseLoop };
    g.nextloop = (g.nextloop + 1) % g.maxloops;
  }
  g.mouseLoop = [];
}

function handleMouseMove(event) {
  event.preventDefault();
  if (! g.mouseDown) { return; }
  var t = now();
  var dt = t - g.mouseTime;
  g.mouseTime = t;
  var x = g.width -  2 * event.clientX;
  var y = g.height - 2 * event.clientY;
  var e = { x: x, y: y, dt: dt };
  g.mouseLoop[g.mouseLoop.length] = e;
}

function start() {
  var canvas = document.getElementById("fractaloids");
  var gl = init();
  if (!gl) { return; }
  g.mouseDown = false;
  g.frameTime = now();
  g.mouseloop = [];
  g.loops = [];
  g.maxloops = 8;
  g.nextloop = 0;
  canvas.addEventListener("webglcontextlost", handleContextLost, false);
  canvas.addEventListener("webglcontextrestored", handleContextRestored, false);
  canvas.addEventListener("mousedown", handleMouseDown, true);
  canvas.addEventListener("mouseup", handleMouseUp, true);
```

```
    canvas.addEventListener("mousemove", handleMouseMove, true);
    g.framerate = new Framerate("framerate");
    g.f = function() {
      render(gl);
      g.requestId = window.requestAnimFrame(g.f, gl);
    };
    g.f();
}

//]]></script></head><body onload="start()">
<div>fractaloids © 2012 Claude Heiland-Allen</div>
<canvas id="fractaloids"></canvas><div id="framerate"></div>
</body></html>
```

23 webgl/index.html

```
<html><head><title>warning</title></head><body><h1>warning</h1>
<p>all liability is disclaimed for fried brains and/or graphics cards</p>
<p>continue to <a href="fractaloids.html">fractaloids/webgl</a> at your own risk
    ↳ </p>
<h2>instructions</h2>
<p>draw shapes (slowly) with the mouse button pressed (you won't see anything
    ↳ until
the second squiggle)</p>
</body></html>
```

24 webgl/J3DI.js

```
/*
 * Copyright (C) 2009 Apple Inc. All Rights Reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY APPLE INC. ``AS IS'' AND ANY
 * EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL APPLE INC. OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
 * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
 * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */

//
// initWebGL
//
```

```
// Initialize the Canvas element with the passed name as a WebGL object and return the
// WebGLRenderingContext.
function initWebGL(canvasName, vshader, fshader, attribs, clearColor, clearDepth)
{
    var canvas = document.getElementById(canvasName);
    return gl = WebGLUtils.setupWebGL(canvas);
}

function log(msg) {
    if (window.console && window.console.log) {
        window.console.log(msg);
    }
}

// Load shaders with the passed names and create a program with them. Return this program
// in the 'program' property of the returned context.
//
// For each string in the passed attribs array, bind an attrib with that name at that index.
// Once the attribs are bound, link the program and then use it.
//
// Set the clear color to the passed array (4 values) and set the clear depth to the passed value.
// Enable depth testing and blending with a blend func of (SRC_ALPHA, ONE_MINUS_SRC_ALPHA)
//
// A console function is added to the context: console(string). This can be replaced
// by the caller. By default, it maps to the window.console() function on WebKit and to
// an empty function on other browsers.
//
function simpleSetup(gl, vshader, fshader, attribs, clearColor, clearDepth)
{
    // create our shaders
    var vertexShader = loadShader(gl, vshader);
    var fragmentShader = loadShader(gl, fshader);

    // Create the program object
    var program = gl.createProgram();

    // Attach our two shaders to the program
    gl.attachShader (program, vertexShader);
    gl.attachShader (program, fragmentShader);

    // Bind attributes
    for (var i = 0; i < attribs.length; ++i)
        gl.bindAttribLocation (program, i, attribs[i]);

    // Link the program
    gl.linkProgram(program);

    // Check the link status
    var linked = gl.getProgramParameter(program, gl.LINK_STATUS);
```

```
    if (!linked && !gl.isContextLost()) {
        // something went wrong with the link
        var error = gl.getProgramInfoLog (program);
        log("Error in program linking:"+error);

        gl.deleteProgram(program);
        gl.deleteProgram(fragmentShader);
        gl.deleteProgram(vertexShader);

        return null;
    }

    gl.useProgram(program);

    gl.clearColor(clearColor[0], clearColor[1], clearColor[2], clearColor[3]);
    gl.clearDepth(clearDepth);

    gl.enable(gl.DEPTH_TEST);
    gl.enable(gl.BLEND);
    gl.blendFunc(gl.SRC_ALPHA, gl.ONE_MINUS_SRC_ALPHA);

    return program;
}

//
// loadShader
//
// 'shaderId' is the id of a <script> element containing the shader source ↙
    ↳ string.
// Load this shader and return the WebGLShader object corresponding to it.
//
function loadShader(ctx, shaderId)
{
    var shaderScript = document.getElementById(shaderId);
    if (!shaderScript) {
        log("*** Error: shader script '"+shaderId+"' not found");
        return null;
    }

    if (shaderScript.type == "x-shader/x-vertex")
        var shaderType = ctx.VERTEX_SHADER;
    else if (shaderScript.type == "x-shader/x-fragment")
        var shaderType = ctx.FRAGMENT_SHADER;
    else {
        log("*** Error: shader script '"+shaderId+"' of undefined type '"+↙
            ↳ shaderScript.type+"'");
        return null;
    }

    // Create the shader object
    var shader = ctx.createShader(shaderType);

    // Load the shader source
    ctx.shaderSource(shader, shaderScript.text);

    // Compile the shader
    ctx.compileShader(shader);
```

```
    // Check the compile status
    var compiled = ctx.getShaderParameter(shader, ctx.COMPILE_STATUS);
    if (!compiled && !ctx.isContextLost()) {
        // Something went wrong during compilation; get the error
        var error = ctx.getShaderInfoLog(shader);
        log("*** Error compiling shader '"+shaderId+"':"+error);
        ctx.deleteShader(shader);
        return null;
    }

    return shader;
}

//
// makeBox
//
// Create a box with vertices, normals and texCoords. Create VBOs for each as well as the index array.
// Return an object with the following properties:
//
//  normalObject        WebGLBuffer object for normals
//  texCoordObject      WebGLBuffer object for texCoords
//  vertexObject        WebGLBuffer object for vertices
//  indexObject         WebGLBuffer object for indices
//  numIndices          The number of indices in the indexObject
//
function makeBox(ctx)
{
    // box
    //    v6----- v5
    //   /|      /|
    //  v1------v0|
    //  | |     | |
    //  | |v7---|-|v4
    //  |/      |/
    //  v2------v3
    //
    // vertex coords array
    var vertices = new Float32Array(
        [  1, 1, 1,  -1, 1, 1,  -1,-1, 1,   1,-1, 1,    // v0-v1-v2-v3 front
           1, 1, 1,   1,-1, 1,   1,-1,-1,   1, 1,-1,    // v0-v3-v4-v5 right
           1, 1, 1,   1, 1,-1,  -1, 1,-1,  -1, 1, 1,    // v0-v5-v6-v1 top
          -1, 1, 1,  -1, 1,-1,  -1,-1,-1,  -1,-1, 1,    // v1-v6-v7-v2 left
          -1,-1,-1,   1,-1,-1,   1,-1, 1,  -1,-1, 1,    // v7-v4-v3-v2 bottom
           1,-1,-1,  -1,-1,-1,  -1, 1,-1,   1, 1,-1 ]   // v4-v7-v6-v5 back
    );

    // normal array
    var normals = new Float32Array(
        [  0, 0, 1,   0, 0, 1,   0, 0, 1,   0, 0, 1,     // v0-v1-v2-v3 front
           1, 0, 0,   1, 0, 0,   1, 0, 0,   1, 0, 0,     // v0-v3-v4-v5 right
           0, 1, 0,   0, 1, 0,   0, 1, 0,   0, 1, 0,     // v0-v5-v6-v1 top
          -1, 0, 0,  -1, 0, 0,  -1, 0, 0,  -1, 0, 0,     // v1-v6-v7-v2 left
           0,-1, 0,   0,-1, 0,   0,-1, 0,   0,-1, 0,     // v7-v4-v3-v2 bottom
           0, 0,-1,   0, 0,-1,   0, 0,-1,   0, 0,-1 ]    // v4-v7-v6-v5 back
       );
```

```
190
        // texCoord array
        var texCoords = new Float32Array(
            [  1, 1,   0, 1,   0, 0,   1, 0,     // v0-v1-v2-v3 front
               0, 1,   0, 0,   1, 0,   1, 1,     // v0-v3-v4-v5 right
195            1, 0,   1, 1,   0, 1,   0, 0,     // v0-v5-v6-v1 top
               1, 1,   0, 1,   0, 0,   1, 0,     // v1-v6-v7-v2 left
               0, 0,   1, 0,   1, 1,   0, 1,     // v7-v4-v3-v2 bottom
               0, 0,   1, 0,   1, 1,   0, 1 ]   // v4-v7-v6-v5 back
           );
200
        // index array
        var indices = new Uint8Array(
            [  0, 1, 2,   0, 2, 3,    // front
               4, 5, 6,   4, 6, 7,    // right
205            8, 9,10,   8,10,11,    // top
              12,13,14,  12,14,15,    // left
              16,17,18,  16,18,19,    // bottom
              20,21,22,  20,22,23 ]   // back
          );
210
        var retval = { };

        retval.normalObject = ctx.createBuffer();
        ctx.bindBuffer(ctx.ARRAY_BUFFER, retval.normalObject);
215     ctx.bufferData(ctx.ARRAY_BUFFER, normals, ctx.STATIC_DRAW);

        retval.texCoordObject = ctx.createBuffer();
        ctx.bindBuffer(ctx.ARRAY_BUFFER, retval.texCoordObject);
        ctx.bufferData(ctx.ARRAY_BUFFER, texCoords, ctx.STATIC_DRAW);
220
        retval.vertexObject = ctx.createBuffer();
        ctx.bindBuffer(ctx.ARRAY_BUFFER, retval.vertexObject);
        ctx.bufferData(ctx.ARRAY_BUFFER, vertices, ctx.STATIC_DRAW);

225     ctx.bindBuffer(ctx.ARRAY_BUFFER, null);

        retval.indexObject = ctx.createBuffer();
        ctx.bindBuffer(ctx.ELEMENT_ARRAY_BUFFER, retval.indexObject);
        ctx.bufferData(ctx.ELEMENT_ARRAY_BUFFER, indices, ctx.STATIC_DRAW);
230     ctx.bindBuffer(ctx.ELEMENT_ARRAY_BUFFER, null);

        retval.numIndices = indices.length;

        return retval;
235 }

    //
    // makeSphere
    //
240 // Create a sphere with the passed number of latitude and longitude bands and ↙
        ↳ the passed radius.
    // Sphere has vertices, normals and texCoords. Create VBOs for each as well as ↙
        ↳ the index array.
    // Return an object with the following properties:
    //
```

```
    //  normalObject        WebGLBuffer object for normals
245 //  texCoordObject      WebGLBuffer object for texCoords
    //  vertexObject        WebGLBuffer object for vertices
    //  indexObject         WebGLBuffer object for indices
    //  numIndices          The number of indices in the indexObject
    //
250 function makeSphere(ctx, radius, lats, longs)
    {
        var geometryData = [ ];
        var normalData = [ ];
        var texCoordData = [ ];
255     var indexData = [ ];

        for (var latNumber = 0; latNumber <= lats; ++latNumber) {
            for (var longNumber = 0; longNumber <= longs; ++longNumber) {
                var theta = latNumber * Math.PI / lats;
260             var phi = longNumber * 2 * Math.PI / longs;
                var sinTheta = Math.sin(theta);
                var sinPhi = Math.sin(phi);
                var cosTheta = Math.cos(theta);
                var cosPhi = Math.cos(phi);
265
                var x = cosPhi * sinTheta;
                var y = cosTheta;
                var z = sinPhi * sinTheta;
                var u = 1-(longNumber/longs);
270             var v = latNumber/lats;

                normalData.push(x);
                normalData.push(y);
                normalData.push(z);
275             texCoordData.push(u);
                texCoordData.push(v);
                geometryData.push(radius * x);
                geometryData.push(radius * y);
                geometryData.push(radius * z);
280         }
        }

        for (var latNumber = 0; latNumber < lats; ++latNumber) {
            for (var longNumber = 0; longNumber < longs; ++longNumber) {
285             var first = (latNumber * (longs+1)) + longNumber;
                var second = first + longs + 1;
                indexData.push(first);
                indexData.push(second);
                indexData.push(first+1);
290
                indexData.push(second);
                indexData.push(second+1);
                indexData.push(first+1);
            }
295     }

        var retval = { };

        retval.normalObject = ctx.createBuffer();
300     ctx.bindBuffer(ctx.ARRAY_BUFFER, retval.normalObject);
```

```
    ctx.bufferData(ctx.ARRAY_BUFFER, new Float32Array(normalData), ctx.↙
        ↳ STATIC_DRAW);

    retval.texCoordObject = ctx.createBuffer();
    ctx.bindBuffer(ctx.ARRAY_BUFFER, retval.texCoordObject);
    ctx.bufferData(ctx.ARRAY_BUFFER, new Float32Array(texCoordData), ctx.↙
        ↳ STATIC_DRAW);

    retval.vertexObject = ctx.createBuffer();
    ctx.bindBuffer(ctx.ARRAY_BUFFER, retval.vertexObject);
    ctx.bufferData(ctx.ARRAY_BUFFER, new Float32Array(geometryData), ctx.↙
        ↳ STATIC_DRAW);

    retval.numIndices = indexData.length;
    retval.indexObject = ctx.createBuffer();
    ctx.bindBuffer(ctx.ELEMENT_ARRAY_BUFFER, retval.indexObject);
    ctx.bufferData(ctx.ELEMENT_ARRAY_BUFFER, new Uint16Array(indexData), ctx.↙
        ↳ STREAM_DRAW);

    return retval;
}

// Array of Objects curently loading
var g_loadingObjects = [];

// Clears all the Objects currently loading.
// This is used to handle context lost events.
function clearLoadingObjects() {
    for (var ii = 0; ii < g_loadingObjects.length; ++ii) {
        g_loadingObjects[ii].onreadystatechange = undefined;
    }
    g_loadingObjects = [];
}

//
// loadObj
//
// Load a .obj file from the passed URL. Return an object with a 'loaded' ↙
    ↳ property set to false.
// When the object load is complete, the 'loaded' property becomes true and the ↙
    ↳ following
// properties are set:
//
//  normalObject        WebGLBuffer object for normals
//  texCoordObject      WebGLBuffer object for texCoords
//  vertexObject        WebGLBuffer object for vertices
//  indexObject         WebGLBuffer object for indices
//  numIndices          The number of indices in the indexObject
//
function loadObj(ctx, url)
{
    var obj = { loaded : false };
    obj.ctx = ctx;
    var req = new XMLHttpRequest();
    req.obj = obj;
    g_loadingObjects.push(req);
    req.onreadystatechange = function () { processLoadObj(req) };
```

```
    req.open("GET", url, true);
    req.send(null);
    return obj;
}

function processLoadObj(req)
{
    log("req="+req)
    // only if req shows "complete"
    if (req.readyState == 4) {
        g_loadingObjects.splice(g_loadingObjects.indexOf(req), 1);
        doLoadObj(req.obj, req.responseText);
    }
}

function doLoadObj(obj, text)
{
    vertexArray = [ ];
    normalArray = [ ];
    textureArray = [ ];
    indexArray = [ ];

    var vertex = [ ];
    var normal = [ ];
    var texture = [ ];
    var facemap = { };
    var index = 0;

    // This is a map which associates a range of indices with a name
    // The name comes from the 'g' tag (of the form "g NAME"). Indices
    // are part of one group until another 'g' tag is seen. If any indices
    // come before a 'g' tag, it is given the group name "_unnamed"
    // 'group' is an object whose property names are the group name and
    // whose value is a 2 element array with [<first index>, <num indices>]
    var groups = { };
    var currentGroup = [-1, 0];
    groups["_unnamed"] = currentGroup;

    var lines = text.split("\n");
    for (var lineIndex in lines) {
        var line = lines[lineIndex].replace(/[ \t]+/g, " ").replace(/\s\s*$/, ↙
            ↳ "");

        // ignore comments
        if (line[0] == "#")
            continue;

        var array = line.split(" ");
        if (array[0] == "g") {
            // new group
            currentGroup = [indexArray.length, 0];
            groups[array[1]] = currentGroup;
        }
        else if (array[0] == "v") {
            // vertex
            vertex.push(parseFloat(array[1]));
            vertex.push(parseFloat(array[2]));
```

```
            vertex.push(parseFloat(array[3]));
        }
        else if (array[0] == "vt") {
            // normal
            texture.push(parseFloat(array[1]));
            texture.push(parseFloat(array[2]));
        }
        else if (array[0] == "vn") {
            // normal
            normal.push(parseFloat(array[1]));
            normal.push(parseFloat(array[2]));
            normal.push(parseFloat(array[3]));
        }
        else if (array[0] == "f") {
            // face
            if (array.length != 4) {
                log("*** Error: face '"+line+"' not handled");
                continue;
            }

            for (var i = 1; i < 4; ++i) {
                if (!(array[i] in facemap)) {
                    // add a new entry to the map and arrays
                    var f = array[i].split("/");
                    var vtx, nor, tex;

                    if (f.length == 1) {
                        vtx = parseInt(f[0]) - 1;
                        nor = vtx;
                        tex = vtx;
                    }
                    else if (f.length = 3) {
                        vtx = parseInt(f[0]) - 1;
                        tex = parseInt(f[1]) - 1;
                        nor = parseInt(f[2]) - 1;
                    }
                    else {
                        obj.ctx.console.log("*** Error: did not understand face ↙
                            ↳ '"+array[i]+"'");
                        return null;
                    }

                    // do the vertices
                    var x = 0;
                    var y = 0;
                    var z = 0;
                    if (vtx * 3 + 2 < vertex.length) {
                        x = vertex[vtx*3];
                        y = vertex[vtx*3+1];
                        z = vertex[vtx*3+2];
                    }
                    vertexArray.push(x);
                    vertexArray.push(y);
                    vertexArray.push(z);

                    // do the textures
                    x = 0;
```

```
                        y = 0;
                        if (tex * 2 + 1 < texture.length) {
                            x = texture[tex*2];
                            y = texture[tex*2+1];
                        }
                        textureArray.push(x);
                        textureArray.push(y);

                        // do the normals
                        x = 0;
                        y = 0;
                        z = 1;
                        if (nor * 3 + 2 < normal.length) {
                            x = normal[nor*3];
                            y = normal[nor*3+1];
                            z = normal[nor*3+2];
                        }
                        normalArray.push(x);
                        normalArray.push(y);
                        normalArray.push(z);

                        facemap[array[i]] = index++;
                    }

                    indexArray.push(facemap[array[i]]);
                    currentGroup[1]++;
                }
            }
        }

        // set the VBOs
        obj.normalObject = obj.ctx.createBuffer();
        obj.ctx.bindBuffer(obj.ctx.ARRAY_BUFFER, obj.normalObject);
        obj.ctx.bufferData(obj.ctx.ARRAY_BUFFER, new Float32Array(normalArray), obj.ctx.STATIC_DRAW);

        obj.texCoordObject = obj.ctx.createBuffer();
        obj.ctx.bindBuffer(obj.ctx.ARRAY_BUFFER, obj.texCoordObject);
        obj.ctx.bufferData(obj.ctx.ARRAY_BUFFER, new Float32Array(textureArray), obj.ctx.STATIC_DRAW);

        obj.vertexObject = obj.ctx.createBuffer();
        obj.ctx.bindBuffer(obj.ctx.ARRAY_BUFFER, obj.vertexObject);
        obj.ctx.bufferData(obj.ctx.ARRAY_BUFFER, new Float32Array(vertexArray), obj.ctx.STATIC_DRAW);

        obj.numIndices = indexArray.length;
        obj.indexObject = obj.ctx.createBuffer();
        obj.ctx.bindBuffer(obj.ctx.ELEMENT_ARRAY_BUFFER, obj.indexObject);
        obj.ctx.bufferData(obj.ctx.ELEMENT_ARRAY_BUFFER, new Uint16Array(indexArray), obj.ctx.STREAM_DRAW);

        obj.groups = groups;

        obj.loaded = true;
    }
```

```
// Array of images curently loading
var g_loadingImages = [];

// Clears all the images currently loading.
// This is used to handle context lost events.
function clearLoadingImages() {
    for (var ii = 0; ii < g_loadingImages.length; ++ii) {
        g_loadingImages[ii].onload = undefined;
    }
    g_loadingImages = [];
}

//
// loadImageTexture
//
// Load the image at the passed url, place it in a new WebGLTexture object and return the WebGLTexture.
//
function loadImageTexture(ctx, url)
{
    var texture = ctx.createTexture();
    var image = new Image();
    g_loadingImages.push(image);
    image.onload = function() { doLoadImageTexture(ctx, image, texture) }
    image.src = url;
    return texture;
}

function doLoadImageTexture(ctx, image, texture)
{
    g_loadingImages.splice(g_loadingImages.indexOf(image), 1);
    ctx.bindTexture(ctx.TEXTURE_2D, texture);
    ctx.texImage2D(
        ctx.TEXTURE_2D, 0, ctx.RGBA, ctx.RGBA, ctx.UNSIGNED_BYTE, image);
    ctx.texParameteri(ctx.TEXTURE_2D, ctx.TEXTURE_MAG_FILTER, ctx.LINEAR);
    ctx.texParameteri(ctx.TEXTURE_2D, ctx.TEXTURE_MIN_FILTER, ctx.LINEAR);
    ctx.texParameteri(ctx.TEXTURE_2D, ctx.TEXTURE_WRAP_S, ctx.CLAMP_TO_EDGE);
    ctx.texParameteri(ctx.TEXTURE_2D, ctx.TEXTURE_WRAP_T, ctx.CLAMP_TO_EDGE);
    //ctx.generateMipmap(ctx.TEXTURE_2D)
    ctx.bindTexture(ctx.TEXTURE_2D, null);
}

//
// Framerate object
//
// This object keeps track of framerate and displays it as the innerHTML text of the
// HTML element with the passed id. Once created you call snapshot at the end
// of every rendering cycle. Every 500ms the framerate is updated in the HTML element.
//
Framerate = function(id)
{
    this.numFramerates = 10;
    this.framerateUpdateInterval = 500;
    this.id = id;

```

```
    this.renderTime = -1;
    this.framerates = [ ];
    self = this;
    var fr = function() { self.updateFramerate() }
    setInterval(fr, this.framerateUpdateInterval);
}

Framerate.prototype.updateFramerate = function()
{
    var tot = 0;
    for (var i = 0; i < this.framerates.length; ++i)
        tot += this.framerates[i];

    var framerate = tot / this.framerates.length;
    framerate = Math.round(framerate);
    document.getElementById(this.id).innerHTML = "Framerate:"+framerate+"fps";
}

Framerate.prototype.snapshot = function()
{
    if (this.renderTime < 0)
        this.renderTime = new Date().getTime();
    else {
        var newTime = new Date().getTime();
        var t = newTime - this.renderTime;
        if (t == 0)
            return;
        var framerate = 1000/t;
        this.framerates.push(framerate);
        while (this.framerates.length > this.numFramerates)
            this.framerates.shift();
        this.renderTime = newTime;
    }
}
```

25 webgl/J3DIMath.js

```
/*
 * Copyright (C) 2009 Apple Inc. All Rights Reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY APPLE INC. ``AS IS'' AND ANY
 * EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL APPLE INC. OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
 * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
```

```
 * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */

// J3DI (Jedi) - A support library for WebGL.

/*
    J3DI Math Classes. Currently includes:

        J3DIMatrix4 - A 4x4 Matrix
*/

/*
    J3DIMatrix4 class

    This class implements a 4x4 matrix. It has functions which duplicate the
    functionality of the OpenGL matrix stack and glut functions. On browsers
    that support it, CSSMatrix is used to accelerate operations.

    IDL:

    [
        Constructor(in J3DIMatrix4 matrix),                 // copy passed
            ↳ matrix into new J3DIMatrix4
        Constructor(in sequence<float> array)               // create new
            ↳ J3DIMatrix4 with 16 floats (row major)
        Constructor()                                       // create new
            ↳ J3DIMatrix4 with identity matrix
    ]
    interface J3DIMatrix4 {
        void load(in J3DIMatrix4 matrix);                   // copy the values
            ↳ from the passed matrix
        void load(in sequence<float> array);                // copy 16 floats
            ↳ into the matrix
        sequence<float> getAsArray();                       // return the matrix
            ↳  as an array of 16 floats
        Float32Array getAsFloat32Array();             // return the matrix as a
            ↳ Float32Array with 16 values
        void setUniform(in WebGLRenderingContext ctx,       // Send the matrix
            ↳ to the passed uniform location in the passed context
                    in WebGLUniformLocation loc,
                    in boolean transpose);
        void makeIdentity();                                // replace the
            ↳ matrix with identity
        void transpose();                                   // replace the
            ↳ matrix with its transpose
        void invert();                                      // replace the
            ↳ matrix with its inverse

        void translate(in float x, in float y, in float z); // multiply the
            ↳ matrix by passed translation values on the right
        void translate(in J3DVector3 v);                    // multiply the
            ↳ matrix by passed translation values on the right
        void scale(in float x, in float y, in float z);     // multiply the
            ↳ matrix by passed scale values on the right
        void scale(in J3DVector3 v);                        // multiply the
```

```
            ↳ matrix by passed scale values on the right
        void rotate(in float angle,                          // multiply the ↙
            ↳ matrix by passed rotation values on the right
                in float x, in float y, in float z);         // (angle is in ↙
                    ↳ degrees)
        void rotate(in float angle, in J3DVector3 v);        // multiply the ↙
            ↳ matrix by passed rotation values on the right
                                                             // (angle is in ↙
                                                                ↳ degrees)
        void multiply(in CanvasMatrix matrix);               // multiply the ↙
            ↳ matrix by the passed matrix on the right
        void divide(in float divisor);                       // divide the matrix↙
            ↳  by the passed divisor
        void ortho(in float left, in float right,            // multiply the ↙
            ↳ matrix by the passed ortho values on the right
                in float bottom, in float top,
                in float near, in float far);
        void frustum(in float left, in float right,          // multiply the ↙
            ↳ matrix by the passed frustum values on the right
                in float bottom, in float top,
                in float near, in float far);
        void perspective(in float fovy, in float aspect,     // multiply the ↙
            ↳ matrix by the passed perspective values on the right
                    in float zNear, in float zFar);
        void lookat(in J3DVector3 eye,                       // multiply the ↙
            ↳ matrix by the passed lookat
            in J3DVector3 center,  in J3DVector3 up);        // values on the ↙
                ↳ right
         bool decompose(in J3DVector3 translate,             // decompose the ↙
             ↳ matrix into the passed vector
                    in J3DVector3 rotate,
                    in J3DVector3 scale,
                    in J3DVector3 skew,
                    in sequence<float> perspective);
    }

    [
        Constructor(in J3DVector3 vector),                   // copy passed ↙
            ↳ vector into new J3DVector3
        Constructor(in sequence<float> array)                // create new ↙
            ↳ J3DVector3 with 3 floats from array
        Constructor(in float x, in float y, in float z)      // create new ↙
            ↳ J3DVector3 with 3 floats
        Constructor()                                        // create new ↙
            ↳ J3DVector3 with (0,0,0)
    ]
    interface J3DVector3 {
        void load(in J3DVector3 vector);                     // copy the values ↙
            ↳ from the passed vector
        void load(in sequence<float> array);                 // copy 3 floats ↙
            ↳ into the vector from array
        void load(in float x, in float y, in float z);       // copy 3 floats ↙
            ↳ into the vector
        sequence<float> getAsArray();                        // return the vector↙
            ↳  as an array of 3 floats
        Float32Array getAsFloat32Array();              // return the matrix as a ↙
            ↳ Float32Array with 16 values
```

```
        void multMatrix(in J3DIMatrix4 matrix);             // multiply the ↙
            ↳ vector by the passed matrix (on the right)
        float vectorLength();                               // return the length↙
            ↳  of the vector
        float dot();                                        // return the dot ↙
            ↳ product of the vector
        void cross(in J3DVector3 v);                        // replace the ↙
            ↳ vector with vector x v
        void divide(in float divisor);                      // divide the vector↙
            ↳  by the passed divisor
    }
*/

J3DIHasCSSMatrix = false;
J3DIHasCSSMatrixCopy = false;
/*
if ("WebKitCSSMatrix" in window && ("media" in window && window.media.↙
    ↳ matchMedium("(-webkit-transform-3d)")) ||
                                    ("styleMedia" in window && window.styleMedia.↙
                                        ↳ matchMedium("(-webkit-transform-3d)")))↙
                                        ↳  {
    J3DIHasCSSMatrix = true;
    if ("copy" in WebKitCSSMatrix.prototype)
        J3DIHasCSSMatrixCopy = true;
}
*/

//  console.log("J3DIHasCSSMatrix="+J3DIHasCSSMatrix);
//  console.log("J3DIHasCSSMatrixCopy="+J3DIHasCSSMatrixCopy);

//
// J3DIMatrix4
//
J3DIMatrix4 = function(m)
{
    if (J3DIHasCSSMatrix)
        this.$matrix = new WebKitCSSMatrix;
    else
        this.$matrix = new Object;

    if (typeof m == 'object') {
        if ("length" in m && m.length >= 16) {
            this.load(m);
            return;
        }
        else if (m instanceof J3DIMatrix4) {
            this.load(m);
            return;
        }
    }
    this.makeIdentity();
}

J3DIMatrix4.prototype.load = function()
{
    if (arguments.length == 1 && typeof arguments[0] == 'object') {
        var matrix;
```

```
        if (arguments[0] instanceof J3DIMatrix4) {
            matrix = arguments[0].$matrix;

            this.$matrix.m11 = matrix.m11;
            this.$matrix.m12 = matrix.m12;
            this.$matrix.m13 = matrix.m13;
            this.$matrix.m14 = matrix.m14;

            this.$matrix.m21 = matrix.m21;
            this.$matrix.m22 = matrix.m22;
            this.$matrix.m23 = matrix.m23;
            this.$matrix.m24 = matrix.m24;

            this.$matrix.m31 = matrix.m31;
            this.$matrix.m32 = matrix.m32;
            this.$matrix.m33 = matrix.m33;
            this.$matrix.m34 = matrix.m34;

            this.$matrix.m41 = matrix.m41;
            this.$matrix.m42 = matrix.m42;
            this.$matrix.m43 = matrix.m43;
            this.$matrix.m44 = matrix.m44;
            return;
        }
        else
            matrix = arguments[0];

        if ("length" in matrix && matrix.length >= 16) {
            this.$matrix.m11 = matrix[0];
            this.$matrix.m12 = matrix[1];
            this.$matrix.m13 = matrix[2];
            this.$matrix.m14 = matrix[3];

            this.$matrix.m21 = matrix[4];
            this.$matrix.m22 = matrix[5];
            this.$matrix.m23 = matrix[6];
            this.$matrix.m24 = matrix[7];

            this.$matrix.m31 = matrix[8];
            this.$matrix.m32 = matrix[9];
            this.$matrix.m33 = matrix[10];
            this.$matrix.m34 = matrix[11];

            this.$matrix.m41 = matrix[12];
            this.$matrix.m42 = matrix[13];
            this.$matrix.m43 = matrix[14];
            this.$matrix.m44 = matrix[15];
            return;
        }
    }

    this.makeIdentity();
}

J3DIMatrix4.prototype.getAsArray = function()
{
```

```
205     return [
            this.$matrix.m11, this.$matrix.m12, this.$matrix.m13, this.$matrix.m14,
            this.$matrix.m21, this.$matrix.m22, this.$matrix.m23, this.$matrix.m24,
            this.$matrix.m31, this.$matrix.m32, this.$matrix.m33, this.$matrix.m34,
            this.$matrix.m41, this.$matrix.m42, this.$matrix.m43, this.$matrix.m44
210     ];
}

J3DIMatrix4.prototype.getAsFloat32Array = function()
{
215     if (J3DIHasCSSMatrixCopy) {
            var array = new Float32Array(16);
            this.$matrix.copy(array);
            return array;
        }
220     return new Float32Array(this.getAsArray());
}

J3DIMatrix4.prototype.setUniform = function(ctx, loc, transpose)
{
225     if (J3DIMatrix4.setUniformArray == undefined) {
            J3DIMatrix4.setUniformWebGLArray = new Float32Array(16);
            J3DIMatrix4.setUniformArray = new Array(16);
        }

230     if (J3DIHasCSSMatrixCopy)
            this.$matrix.copy(J3DIMatrix4.setUniformWebGLArray);
        else {
            J3DIMatrix4.setUniformArray[0] = this.$matrix.m11;
            J3DIMatrix4.setUniformArray[1] = this.$matrix.m12;
235         J3DIMatrix4.setUniformArray[2] = this.$matrix.m13;
            J3DIMatrix4.setUniformArray[3] = this.$matrix.m14;
            J3DIMatrix4.setUniformArray[4] = this.$matrix.m21;
            J3DIMatrix4.setUniformArray[5] = this.$matrix.m22;
            J3DIMatrix4.setUniformArray[6] = this.$matrix.m23;
240         J3DIMatrix4.setUniformArray[7] = this.$matrix.m24;
            J3DIMatrix4.setUniformArray[8] = this.$matrix.m31;
            J3DIMatrix4.setUniformArray[9] = this.$matrix.m32;
            J3DIMatrix4.setUniformArray[10] = this.$matrix.m33;
            J3DIMatrix4.setUniformArray[11] = this.$matrix.m34;
245         J3DIMatrix4.setUniformArray[12] = this.$matrix.m41;
            J3DIMatrix4.setUniformArray[13] = this.$matrix.m42;
            J3DIMatrix4.setUniformArray[14] = this.$matrix.m43;
            J3DIMatrix4.setUniformArray[15] = this.$matrix.m44;

250         J3DIMatrix4.setUniformWebGLArray.set(J3DIMatrix4.setUniformArray);
        }

        ctx.uniformMatrix4fv(loc, transpose, J3DIMatrix4.setUniformWebGLArray);
}
255
J3DIMatrix4.prototype.makeIdentity = function()
{
        this.$matrix.m11 = 1;
        this.$matrix.m12 = 0;
260     this.$matrix.m13 = 0;
        this.$matrix.m14 = 0;
```

```
        this.$matrix.m21 = 0;
        this.$matrix.m22 = 1;
        this.$matrix.m23 = 0;
        this.$matrix.m24 = 0;

        this.$matrix.m31 = 0;
        this.$matrix.m32 = 0;
        this.$matrix.m33 = 1;
        this.$matrix.m34 = 0;

        this.$matrix.m41 = 0;
        this.$matrix.m42 = 0;
        this.$matrix.m43 = 0;
        this.$matrix.m44 = 1;
    }

    J3DIMatrix4.prototype.transpose = function()
    {
        var tmp = this.$matrix.m12;
        this.$matrix.m12 = this.$matrix.m21;
        this.$matrix.m21 = tmp;

        tmp = this.$matrix.m13;
        this.$matrix.m13 = this.$matrix.m31;
        this.$matrix.m31 = tmp;

        tmp = this.$matrix.m14;
        this.$matrix.m14 = this.$matrix.m41;
        this.$matrix.m41 = tmp;

        tmp = this.$matrix.m23;
        this.$matrix.m23 = this.$matrix.m32;
        this.$matrix.m32 = tmp;

        tmp = this.$matrix.m24;
        this.$matrix.m24 = this.$matrix.m42;
        this.$matrix.m42 = tmp;

        tmp = this.$matrix.m34;
        this.$matrix.m34 = this.$matrix.m43;
        this.$matrix.m43 = tmp;
    }

    J3DIMatrix4.prototype.invert = function()
    {
        if (J3DIHasCSSMatrix) {
            this.$matrix = this.$matrix.inverse();
            return;
        }

        // Calculate the 4x4 determinant
        // If the determinant is zero,
        // then the inverse matrix is not unique.
        var det = this._determinant4x4();

        if (Math.abs(det) < 1e-8)
```

```
            return null;

        this._makeAdjoint();

        // Scale the adjoint matrix to get the inverse
        this.$matrix.m11 /= det;
        this.$matrix.m12 /= det;
        this.$matrix.m13 /= det;
        this.$matrix.m14 /= det;

        this.$matrix.m21 /= det;
        this.$matrix.m22 /= det;
        this.$matrix.m23 /= det;
        this.$matrix.m24 /= det;

        this.$matrix.m31 /= det;
        this.$matrix.m32 /= det;
        this.$matrix.m33 /= det;
        this.$matrix.m34 /= det;

        this.$matrix.m41 /= det;
        this.$matrix.m42 /= det;
        this.$matrix.m43 /= det;
        this.$matrix.m44 /= det;
    }

    J3DIMatrix4.prototype.translate = function(x,y,z)
    {
        if (typeof x == 'object' && "length" in x) {
            var t = x;
            x = t[0];
            y = t[1];
            z = t[2];
        }
        else {
            if (x == undefined)
                x = 0;
            if (y == undefined)
                y = 0;
            if (z == undefined)
                z = 0;
        }

        if (J3DIHasCSSMatrix) {
            this.$matrix = this.$matrix.translate(x, y, z);
            return;
        }

        var matrix = new J3DIMatrix4();
        matrix.$matrix.m41 = x;
        matrix.$matrix.m42 = y;
        matrix.$matrix.m43 = z;

        this.multiply(matrix);
    }

    J3DIMatrix4.prototype.scale = function(x,y,z)
```

```
{
    if (typeof x == 'object' && "length" in x) {
        var t = x;
        x = t[0];
        y = t[1];
        z = t[2];
    }
    else {
        if (x == undefined)
            x = 1;
        if (z == undefined) {
            if (y == undefined) {
                y = x;
                z = x;
            }
            else
                z = 1;
        }
        else if (y == undefined)
            y = x;
    }

    if (J3DIHasCSSMatrix) {
        this.$matrix = this.$matrix.scale(x, y, z);
        return;
    }

    var matrix = new J3DIMatrix4();
    matrix.$matrix.m11 = x;
    matrix.$matrix.m22 = y;
    matrix.$matrix.m33 = z;

    this.multiply(matrix);
}

J3DIMatrix4.prototype.rotate = function(angle,x,y,z)
{
    // Forms are (angle, x,y,z), (angle,vector), (angleX, angleY, angleZ), (↙
        ↳ angle)
    if (typeof x == 'object' && "length" in x) {
        var t = x;
        x = t[0];
        y = t[1];
        z = t[2];
    }
    else {
        if (arguments.length == 1) {
            x = 0;
            y = 0;
            z = 1;
        }
        else if (arguments.length == 3) {
            this.rotate(angle, 1,0,0); // about X axis
            this.rotate(x, 0,1,0); // about Y axis
            this.rotate(y, 0,0,1); // about Z axis
            return;
        }
```

```
    }

    if (J3DIHasCSSMatrix) {
        this.$matrix = this.$matrix.rotateAxisAngle(x, y, z, angle);
        return;
    }

    // angles are in degrees. Switch to radians
    angle = angle / 180 * Math.PI;

    angle /= 2;
    var sinA = Math.sin(angle);
    var cosA = Math.cos(angle);
    var sinA2 = sinA * sinA;

    // normalize
    var len = Math.sqrt(x * x + y * y + z * z);
    if (len == 0) {
        // bad vector, just use something reasonable
        x = 0;
        y = 0;
        z = 1;
    } else if (len != 1) {
        x /= len;
        y /= len;
        z /= len;
    }

    var mat = new J3DIMatrix4();

    // optimize case where axis is along major axis
    if (x == 1 && y == 0 && z == 0) {
        mat.$matrix.m11 = 1;
        mat.$matrix.m12 = 0;
        mat.$matrix.m13 = 0;
        mat.$matrix.m21 = 0;
        mat.$matrix.m22 = 1 - 2 * sinA2;
        mat.$matrix.m23 = 2 * sinA * cosA;
        mat.$matrix.m31 = 0;
        mat.$matrix.m32 = -2 * sinA * cosA;
        mat.$matrix.m33 = 1 - 2 * sinA2;
        mat.$matrix.m14 = mat.$matrix.m24 = mat.$matrix.m34 = 0;
        mat.$matrix.m41 = mat.$matrix.m42 = mat.$matrix.m43 = 0;
        mat.$matrix.m44 = 1;
    } else if (x == 0 && y == 1 && z == 0) {
        mat.$matrix.m11 = 1 - 2 * sinA2;
        mat.$matrix.m12 = 0;
        mat.$matrix.m13 = -2 * sinA * cosA;
        mat.$matrix.m21 = 0;
        mat.$matrix.m22 = 1;
        mat.$matrix.m23 = 0;
        mat.$matrix.m31 = 2 * sinA * cosA;
        mat.$matrix.m32 = 0;
        mat.$matrix.m33 = 1 - 2 * sinA2;
        mat.$matrix.m14 = mat.$matrix.m24 = mat.$matrix.m34 = 0;
        mat.$matrix.m41 = mat.$matrix.m42 = mat.$matrix.m43 = 0;
        mat.$matrix.m44 = 1;
```

```
    } else if (x == 0 && y == 0 && z == 1) {
        mat.$matrix.m11 = 1 - 2 * sinA2;
        mat.$matrix.m12 = 2 * sinA * cosA;
        mat.$matrix.m13 = 0;
        mat.$matrix.m21 = -2 * sinA * cosA;
        mat.$matrix.m22 = 1 - 2 * sinA2;
        mat.$matrix.m23 = 0;
        mat.$matrix.m31 = 0;
        mat.$matrix.m32 = 0;
        mat.$matrix.m33 = 1;
        mat.$matrix.m14 = mat.$matrix.m24 = mat.$matrix.m34 = 0;
        mat.$matrix.m41 = mat.$matrix.m42 = mat.$matrix.m43 = 0;
        mat.$matrix.m44 = 1;
    } else {
        var x2 = x*x;
        var y2 = y*y;
        var z2 = z*z;

        mat.$matrix.m11 = 1 - 2 * (y2 + z2) * sinA2;
        mat.$matrix.m12 = 2 * (x * y * sinA2 + z * sinA * cosA);
        mat.$matrix.m13 = 2 * (x * z * sinA2 - y * sinA * cosA);
        mat.$matrix.m21 = 2 * (y * x * sinA2 - z * sinA * cosA);
        mat.$matrix.m22 = 1 - 2 * (z2 + x2) * sinA2;
        mat.$matrix.m23 = 2 * (y * z * sinA2 + x * sinA * cosA);
        mat.$matrix.m31 = 2 * (z * x * sinA2 + y * sinA * cosA);
        mat.$matrix.m32 = 2 * (z * y * sinA2 - x * sinA * cosA);
        mat.$matrix.m33 = 1 - 2 * (x2 + y2) * sinA2;
        mat.$matrix.m14 = mat.$matrix.m24 = mat.$matrix.m34 = 0;
        mat.$matrix.m41 = mat.$matrix.m42 = mat.$matrix.m43 = 0;
        mat.$matrix.m44 = 1;
    }
    this.multiply(mat);
}

J3DIMatrix4.prototype.multiply = function(mat)
{
    if (J3DIHasCSSMatrix) {
        this.$matrix = this.$matrix.multiply(mat.$matrix);
        return;
    }

    var m11 = (mat.$matrix.m11 * this.$matrix.m11 + mat.$matrix.m12 * this.$matrix.m21
            + mat.$matrix.m13 * this.$matrix.m31 + mat.$matrix.m14 * this.$matrix.m41);
    var m12 = (mat.$matrix.m11 * this.$matrix.m12 + mat.$matrix.m12 * this.$matrix.m22
            + mat.$matrix.m13 * this.$matrix.m32 + mat.$matrix.m14 * this.$matrix.m42);
    var m13 = (mat.$matrix.m11 * this.$matrix.m13 + mat.$matrix.m12 * this.$matrix.m23
            + mat.$matrix.m13 * this.$matrix.m33 + mat.$matrix.m14 * this.$matrix.m43);
    var m14 = (mat.$matrix.m11 * this.$matrix.m14 + mat.$matrix.m12 * this.$matrix.m24
            + mat.$matrix.m13 * this.$matrix.m34 + mat.$matrix.m14 * this.$matrix.m44);
```

```
        var m21 = (mat.$matrix.m21 * this.$matrix.m11 + mat.$matrix.m22 * this.$matrix.m21
            + mat.$matrix.m23 * this.$matrix.m31 + mat.$matrix.m24 * this.$matrix.m41);
        var m22 = (mat.$matrix.m21 * this.$matrix.m12 + mat.$matrix.m22 * this.$matrix.m22
            + mat.$matrix.m23 * this.$matrix.m32 + mat.$matrix.m24 * this.$matrix.m42);
        var m23 = (mat.$matrix.m21 * this.$matrix.m13 + mat.$matrix.m22 * this.$matrix.m23
            + mat.$matrix.m23 * this.$matrix.m33 + mat.$matrix.m24 * this.$matrix.m43);
        var m24 = (mat.$matrix.m21 * this.$matrix.m14 + mat.$matrix.m22 * this.$matrix.m24
            + mat.$matrix.m23 * this.$matrix.m34 + mat.$matrix.m24 * this.$matrix.m44);

        var m31 = (mat.$matrix.m31 * this.$matrix.m11 + mat.$matrix.m32 * this.$matrix.m21
            + mat.$matrix.m33 * this.$matrix.m31 + mat.$matrix.m34 * this.$matrix.m41);
        var m32 = (mat.$matrix.m31 * this.$matrix.m12 + mat.$matrix.m32 * this.$matrix.m22
            + mat.$matrix.m33 * this.$matrix.m32 + mat.$matrix.m34 * this.$matrix.m42);
        var m33 = (mat.$matrix.m31 * this.$matrix.m13 + mat.$matrix.m32 * this.$matrix.m23
            + mat.$matrix.m33 * this.$matrix.m33 + mat.$matrix.m34 * this.$matrix.m43);
        var m34 = (mat.$matrix.m31 * this.$matrix.m14 + mat.$matrix.m32 * this.$matrix.m24
            + mat.$matrix.m33 * this.$matrix.m34 + mat.$matrix.m34 * this.$matrix.m44);

        var m41 = (mat.$matrix.m41 * this.$matrix.m11 + mat.$matrix.m42 * this.$matrix.m21
            + mat.$matrix.m43 * this.$matrix.m31 + mat.$matrix.m44 * this.$matrix.m41);
        var m42 = (mat.$matrix.m41 * this.$matrix.m12 + mat.$matrix.m42 * this.$matrix.m22
            + mat.$matrix.m43 * this.$matrix.m32 + mat.$matrix.m44 * this.$matrix.m42);
        var m43 = (mat.$matrix.m41 * this.$matrix.m13 + mat.$matrix.m42 * this.$matrix.m23
            + mat.$matrix.m43 * this.$matrix.m33 + mat.$matrix.m44 * this.$matrix.m43);
        var m44 = (mat.$matrix.m41 * this.$matrix.m14 + mat.$matrix.m42 * this.$matrix.m24
            + mat.$matrix.m43 * this.$matrix.m34 + mat.$matrix.m44 * this.$matrix.m44);

        this.$matrix.m11 = m11;
        this.$matrix.m12 = m12;
        this.$matrix.m13 = m13;
        this.$matrix.m14 = m14;
```

```
        this.$matrix.m21 = m21;
        this.$matrix.m22 = m22;
        this.$matrix.m23 = m23;
        this.$matrix.m24 = m24;

        this.$matrix.m31 = m31;
        this.$matrix.m32 = m32;
        this.$matrix.m33 = m33;
        this.$matrix.m34 = m34;

        this.$matrix.m41 = m41;
        this.$matrix.m42 = m42;
        this.$matrix.m43 = m43;
        this.$matrix.m44 = m44;
}

J3DIMatrix4.prototype.divide = function(divisor)
{
        this.$matrix.m11 /= divisor;
        this.$matrix.m12 /= divisor;
        this.$matrix.m13 /= divisor;
        this.$matrix.m14 /= divisor;

        this.$matrix.m21 /= divisor;
        this.$matrix.m22 /= divisor;
        this.$matrix.m23 /= divisor;
        this.$matrix.m24 /= divisor;

        this.$matrix.m31 /= divisor;
        this.$matrix.m32 /= divisor;
        this.$matrix.m33 /= divisor;
        this.$matrix.m34 /= divisor;

        this.$matrix.m41 /= divisor;
        this.$matrix.m42 /= divisor;
        this.$matrix.m43 /= divisor;
        this.$matrix.m44 /= divisor;

}

J3DIMatrix4.prototype.ortho = function(left, right, bottom, top, near, far)
{
        var tx = (left + right) / (left - right);
        var ty = (top + bottom) / (top - bottom);
        var tz = (far + near) / (far - near);

        var matrix = new J3DIMatrix4();
        matrix.$matrix.m11 = 2 / (left - right);
        matrix.$matrix.m12 = 0;
        matrix.$matrix.m13 = 0;
        matrix.$matrix.m14 = 0;
        matrix.$matrix.m21 = 0;
        matrix.$matrix.m22 = 2 / (top - bottom);
        matrix.$matrix.m23 = 0;
        matrix.$matrix.m24 = 0;
        matrix.$matrix.m31 = 0;
        matrix.$matrix.m32 = 0;
```

```
    matrix.$matrix.m33 = -2 / (far - near);
    matrix.$matrix.m34 = 0;
    matrix.$matrix.m41 = tx;
    matrix.$matrix.m42 = ty;
    matrix.$matrix.m43 = tz;
    matrix.$matrix.m44 = 1;

    this.multiply(matrix);
}

J3DIMatrix4.prototype.frustum = function(left, right, bottom, top, near, far)
{
    var matrix = new J3DIMatrix4();
    var A = (right + left) / (right - left);
    var B = (top + bottom) / (top - bottom);
    var C = -(far + near) / (far - near);
    var D = -(2 * far * near) / (far - near);

    matrix.$matrix.m11 = (2 * near) / (right - left);
    matrix.$matrix.m12 = 0;
    matrix.$matrix.m13 = 0;
    matrix.$matrix.m14 = 0;

    matrix.$matrix.m21 = 0;
    matrix.$matrix.m22 = 2 * near / (top - bottom);
    matrix.$matrix.m23 = 0;
    matrix.$matrix.m24 = 0;

    matrix.$matrix.m31 = A;
    matrix.$matrix.m32 = B;
    matrix.$matrix.m33 = C;
    matrix.$matrix.m34 = -1;

    matrix.$matrix.m41 = 0;
    matrix.$matrix.m42 = 0;
    matrix.$matrix.m43 = D;
    matrix.$matrix.m44 = 0;

    this.multiply(matrix);
}

J3DIMatrix4.prototype.perspective = function(fovy, aspect, zNear, zFar)
{
    var top = Math.tan(fovy * Math.PI / 360) * zNear;
    var bottom = -top;
    var left = aspect * bottom;
    var right = aspect * top;
    this.frustum(left, right, bottom, top, zNear, zFar);
}

J3DIMatrix4.prototype.lookat = function(eyex, eyey, eyez, centerx, centery, ↙
    ↪ centerz, upx, upy, upz)
{
    if (typeof eyez == 'object' && "length" in eyez) {
        var t = eyez;
        upx = t[0];
        upy = t[1];
```

```
            upz = t[2];

            t = eyey;
            centerx = t[0];
            centery = t[1];
            centerz = t[2];

            t = eyex;
            eyex = t[0];
            eyey = t[1];
            eyez = t[2];
        }

        var matrix = new J3DIMatrix4();

        // Make rotation matrix

        // Z vector
        var zx = eyex - centerx;
        var zy = eyey - centery;
        var zz = eyez - centerz;
        var mag = Math.sqrt(zx * zx + zy * zy + zz * zz);
        if (mag) {
            zx /= mag;
            zy /= mag;
            zz /= mag;
        }

        // Y vector
        var yx = upx;
        var yy = upy;
        var yz = upz;

        // X vector = Y cross Z
        xx =  yy * zz - yz * zy;
        xy = -yx * zz + yz * zx;
        xz =  yx * zy - yy * zx;

        // Recompute Y = Z cross X
        yx = zy * xz - zz * xy;
        yy = -zx * xz + zz * xx;
        yx = zx * xy - zy * xx;

        // cross product gives area of parallelogram, which is < 1.0 for
        // non-perpendicular unit-length vectors; so normalize x, y here

        mag = Math.sqrt(xx * xx + xy * xy + xz * xz);
        if (mag) {
            xx /= mag;
            xy /= mag;
            xz /= mag;
        }

        mag = Math.sqrt(yx * yx + yy * yy + yz * yz);
        if (mag) {
            yx /= mag;
            yy /= mag;
```

```
        yz /= mag;
    }

    matrix.$matrix.m11 = xx;
    matrix.$matrix.m12 = xy;
    matrix.$matrix.m13 = xz;
    matrix.$matrix.m14 = 0;

    matrix.$matrix.m21 = yx;
    matrix.$matrix.m22 = yy;
    matrix.$matrix.m23 = yz;
    matrix.$matrix.m24 = 0;

    matrix.$matrix.m31 = zx;
    matrix.$matrix.m32 = zy;
    matrix.$matrix.m33 = zz;
    matrix.$matrix.m34 = 0;

    matrix.$matrix.m41 = 0;
    matrix.$matrix.m42 = 0;
    matrix.$matrix.m43 = 0;
    matrix.$matrix.m44 = 1;
    matrix.translate(-eyex, -eyey, -eyez);

    this.multiply(matrix);
}

// Returns true on success, false otherwise. All params are Array objects
J3DIMatrix4.prototype.decompose = function(_translate, _rotate, _scale, _skew, ↙
    ↪ _perspective)
{
    // Normalize the matrix.
    if (this.$matrix.m44 == 0)
        return false;

    // Gather the params
    var translate, rotate, scale, skew, perspective;

    var translate = (_translate == undefined || !("length" in _translate)) ? new↙
        ↪ J3DIVector3 : _translate;
    var rotate = (_rotate == undefined || !("length" in _rotate)) ? new ↙
        ↪ J3DIVector3 : _rotate;
    var scale = (_scale == undefined || !("length" in _scale)) ? new J3DIVector3↙
        ↪ : _scale;
    var skew = (_skew == undefined || !("length" in _skew)) ? new J3DIVector3 : ↙
        ↪ _skew;
    var perspective = (_perspective == undefined || !("length" in _perspective))↙
        ↪ ? new Array(4) : _perspective;

    var matrix = new J3DIMatrix4(this);

    matrix.divide(matrix.$matrix.m44);

    // perspectiveMatrix is used to solve for perspective, but it also provides
    // an easy way to test for singularity of the upper 3x3 component.
    var perspectiveMatrix = new J3DIMatrix4(matrix);
```

```
        perspectiveMatrix.$matrix.m14 = 0;
        perspectiveMatrix.$matrix.m24 = 0;
        perspectiveMatrix.$matrix.m34 = 0;
795     perspectiveMatrix.$matrix.m44 = 1;

        if (perspectiveMatrix._determinant4x4() == 0)
            return false;

800     // First, isolate perspective.
        if (matrix.$matrix.m14 != 0 || matrix.$matrix.m24 != 0 || matrix.$matrix.m34
            ↳ != 0) {
            // rightHandSide is the right hand side of the equation.
            var rightHandSide = [ matrix.$matrix.m14, matrix.$matrix.m24, matrix.
                ↳ $matrix.m34, matrix.$matrix.m44 ];

805         // Solve the equation by inverting perspectiveMatrix and multiplying
            // rightHandSide by the inverse.
            var inversePerspectiveMatrix = new J3DIMatrix4(perspectiveMatrix);
            inversePerspectiveMatrix.invert();
            var transposedInversePerspectiveMatrix = new J3DIMatrix4(
                ↳ inversePerspectiveMatrix);
810         transposedInversePerspectiveMatrix.transpose();
            transposedInversePerspectiveMatrix.multVecMatrix(perspective,
                ↳ rightHandSide);

            // Clear the perspective partition
            matrix.$matrix.m14 = matrix.$matrix.m24 = matrix.$matrix.m34 = 0
815         matrix.$matrix.m44 = 1;
        }
        else {
            // No perspective.
            perspective[0] = perspective[1] = perspective[2] = 0;
820         perspective[3] = 1;
        }

        // Next take care of translation
        translate[0] = matrix.$matrix.m41
825     matrix.$matrix.m41 = 0
        translate[1] = matrix.$matrix.m42
        matrix.$matrix.m42 = 0
        translate[2] = matrix.$matrix.m43
        matrix.$matrix.m43 = 0
830
        // Now get scale and shear. 'row' is a 3 element array of 3 component
            ↳ vectors
        var row0 = new J3DIVector3(matrix.$matrix.m11, matrix.$matrix.m12, matrix.
            ↳ $matrix.m13);
        var row1 = new J3DIVector3(matrix.$matrix.m21, matrix.$matrix.m22, matrix.
            ↳ $matrix.m23);
        var row2 = new J3DIVector3(matrix.$matrix.m31, matrix.$matrix.m32, matrix.
            ↳ $matrix.m33);
835
        // Compute X scale factor and normalize first row.
        scale[0] = row0.vectorLength();
        row0.divide(scale[0]);

840     // Compute XY shear factor and make 2nd row orthogonal to 1st.
```

```
        skew[0] = row0.dot(row1);
        row1.combine(row0, 1.0, -skew[0]);

        // Now, compute Y scale and normalize 2nd row.
        scale[1] = row1.vectorLength();
        row1.divide(scale[1]);
        skew[0] /= scale[1];

        // Compute XZ and YZ shears, orthogonalize 3rd row
        skew[1] = row1.dot(row2);
        row2.combine(row0, 1.0, -skew[1]);
        skew[2] = row1.dot(row2);
        row2.combine(row1, 1.0, -skew[2]);

        // Next, get Z scale and normalize 3rd row.
        scale[2] = row2.vectorLength();
        row2.divide(scale[2]);
        skew[1] /= scale[2];
        skew[2] /= scale[2];

        // At this point, the matrix (in rows) is orthonormal.
        // Check for a coordinate system flip.  If the determinant
        // is -1, then negate the matrix and the scaling factors.
        var pdum3 = new J3DIVector3(row1);
        pdum3.cross(row2);
        if (row0.dot(pdum3) < 0) {
            for (i = 0; i < 3; i++) {
                scale[i] *= -1;
                row[0][i] *= -1;
                row[1][i] *= -1;
                row[2][i] *= -1;
            }
        }

        // Now, get the rotations out
        rotate[1] = Math.asin(-row0[2]);
        if (Math.cos(rotate[1]) != 0) {
            rotate[0] = Math.atan2(row1[2], row2[2]);
            rotate[2] = Math.atan2(row0[1], row0[0]);
        }
        else {
            rotate[0] = Math.atan2(-row2[0], row1[1]);
            rotate[2] = 0;
        }

        // Convert rotations to degrees
        var rad2deg = 180 / Math.PI;
        rotate[0] *= rad2deg;
        rotate[1] *= rad2deg;
        rotate[2] *= rad2deg;

        return true;
    }

    J3DIMatrix4.prototype._determinant2x2 = function(a, b, c, d)
    {
        return a * d - b * c;
```

```
}

J3DIMatrix4.prototype._determinant3x3 = function(a1, a2, a3, b1, b2, b3, c1, c2,
    c3)
{
    return a1 * this._determinant2x2(b2, b3, c2, c3)
         - b1 * this._determinant2x2(a2, a3, c2, c3)
         + c1 * this._determinant2x2(a2, a3, b2, b3);
}

J3DIMatrix4.prototype._determinant4x4 = function()
{
    var a1 = this.$matrix.m11;
    var b1 = this.$matrix.m12;
    var c1 = this.$matrix.m13;
    var d1 = this.$matrix.m14;

    var a2 = this.$matrix.m21;
    var b2 = this.$matrix.m22;
    var c2 = this.$matrix.m23;
    var d2 = this.$matrix.m24;

    var a3 = this.$matrix.m31;
    var b3 = this.$matrix.m32;
    var c3 = this.$matrix.m33;
    var d3 = this.$matrix.m34;

    var a4 = this.$matrix.m41;
    var b4 = this.$matrix.m42;
    var c4 = this.$matrix.m43;
    var d4 = this.$matrix.m44;

    return a1 * this._determinant3x3(b2, b3, b4, c2, c3, c4, d2, d3, d4)
         - b1 * this._determinant3x3(a2, a3, a4, c2, c3, c4, d2, d3, d4)
         + c1 * this._determinant3x3(a2, a3, a4, b2, b3, b4, d2, d3, d4)
         - d1 * this._determinant3x3(a2, a3, a4, b2, b3, b4, c2, c3, c4);
}

J3DIMatrix4.prototype._makeAdjoint = function()
{
    var a1 = this.$matrix.m11;
    var b1 = this.$matrix.m12;
    var c1 = this.$matrix.m13;
    var d1 = this.$matrix.m14;

    var a2 = this.$matrix.m21;
    var b2 = this.$matrix.m22;
    var c2 = this.$matrix.m23;
    var d2 = this.$matrix.m24;

    var a3 = this.$matrix.m31;
    var b3 = this.$matrix.m32;
    var c3 = this.$matrix.m33;
    var d3 = this.$matrix.m34;

    var a4 = this.$matrix.m41;
    var b4 = this.$matrix.m42;
```

```
    var c4 = this.$matrix.m43;
    var d4 = this.$matrix.m44;

    // Row column labeling reversed since we transpose rows & columns
    this.$matrix.m11 =   this._determinant3x3(b2, b3, b4, c2, c3, c4, d2, d3, d4);
    this.$matrix.m21 = - this._determinant3x3(a2, a3, a4, c2, c3, c4, d2, d3, d4);
    this.$matrix.m31 =   this._determinant3x3(a2, a3, a4, b2, b3, b4, d2, d3, d4);
    this.$matrix.m41 = - this._determinant3x3(a2, a3, a4, b2, b3, b4, c2, c3, c4);

    this.$matrix.m12 = - this._determinant3x3(b1, b3, b4, c1, c3, c4, d1, d3, d4);
    this.$matrix.m22 =   this._determinant3x3(a1, a3, a4, c1, c3, c4, d1, d3, d4);
    this.$matrix.m32 = - this._determinant3x3(a1, a3, a4, b1, b3, b4, d1, d3, d4);
    this.$matrix.m42 =   this._determinant3x3(a1, a3, a4, b1, b3, b4, c1, c3, c4);

    this.$matrix.m13 =   this._determinant3x3(b1, b2, b4, c1, c2, c4, d1, d2, d4);
    this.$matrix.m23 = - this._determinant3x3(a1, a2, a4, c1, c2, c4, d1, d2, d4);
    this.$matrix.m33 =   this._determinant3x3(a1, a2, a4, b1, b2, b4, d1, d2, d4);
    this.$matrix.m43 = - this._determinant3x3(a1, a2, a4, b1, b2, b4, c1, c2, c4);

    this.$matrix.m14 = - this._determinant3x3(b1, b2, b3, c1, c2, c3, d1, d2, d3);
    this.$matrix.m24 =   this._determinant3x3(a1, a2, a3, c1, c2, c3, d1, d2, d3);
    this.$matrix.m34 = - this._determinant3x3(a1, a2, a3, b1, b2, b3, d1, d2, d3);
    this.$matrix.m44 =   this._determinant3x3(a1, a2, a3, b1, b2, b3, c1, c2, c3);
}

//
// J3DIVector3
//
J3DIVector3 = function(x,y,z)
{
    this.load(x,y,z);
}

J3DIVector3.prototype.load = function(x,y,z)
{
    if (typeof x == 'object' && "length" in x) {
        this[0] = x[0];
        this[1] = x[1];
        this[2] = x[2];
    }
    else if (typeof x == 'number') {
```

```
995             this[0] = x;
                this[1] = y;
                this[2] = z;
            }
            else {
1000            this[0] = 0;
                this[1] = 0;
                this[2] = 0;
            }
        }
1005
        J3DIVector3.prototype.getAsArray = function()
        {
            return [ this[0], this[1], this[2] ];
        }
1010
        J3DIVector3.prototype.getAsFloat32Array = function()
        {
            return new Float32Array(this.getAsArray());
        }
1015
        J3DIVector3.prototype.vectorLength = function()
        {
            return Math.sqrt(this[0] * this[0] + this[1] * this[1] + this[2] * this[2]);
        }
1020
        J3DIVector3.prototype.divide = function(divisor)
        {
            this[0] /= divisor; this[1] /= divisor; this[2] /= divisor;
        }
1025
        J3DIVector3.prototype.cross = function(v)
        {
            this[0] =  this[1] * v[2] - this[2] * v[1];
            this[1] = -this[0] * v[2] + this[2] * v[0];
1030        this[2] =  this[0] * v[1] - this[1] * v[0];
        }

        J3DIVector3.prototype.dot = function(v)
        {
1035        return this[0] * v[0] + this[1] * v[1] + this[2] * v[2];
        }

        J3DIVector3.prototype.combine = function(v, ascl, bscl)
        {
1040        this[0] = (ascl * this[0]) + (bscl * v[0]);
            this[1] = (ascl * this[1]) + (bscl * v[1]);
            this[2] = (ascl * this[2]) + (bscl * v[2]);
        }

1045    J3DIVector3.prototype.multVecMatrix = function(matrix)
        {
            var x = this[0];
            var y = this[1];
            var z = this[2];
1050
            this[0] = matrix.$matrix.m41 + x * matrix.$matrix.m11 + y * matrix.$matrix.↙
```

```
        ↪ m21 + z * matrix.$matrix.m31;
    this[1] = matrix.$matrix.m42 + x * matrix.$matrix.m12 + y * matrix.$matrix.↙
        ↪ m22 + z * matrix.$matrix.m32;
    this[2] = matrix.$matrix.m43 + x * matrix.$matrix.m13 + y * matrix.$matrix.↙
        ↪ m23 + z * matrix.$matrix.m33;
    var w = matrix.$matrix.m44 + x * matrix.$matrix.m14 + y * matrix.$matrix.m24↙
        ↪  + z * matrix.$matrix.m34;
    if (w != 1 && w != 0) {
        this[0] /= w;
        this[1] /= w;
        this[2] /= w;
    }
}

J3DIVector3.prototype.toString = function()
{
    return "["+this[0]+","+this[1]+","+this[2]+"]";
}
```

26 webgl/webgl-debug.js

```
/*
** Copyright (c) 2012 The Khronos Group Inc.
**
** Permission is hereby granted, free of charge, to any person obtaining a
** copy of this software and/or associated documentation files (the
** "Materials"), to deal in the Materials without restriction, including
** without limitation the rights to use, copy, modify, merge, publish,
** distribute, sublicense, and/or sell copies of the Materials, and to
** permit persons to whom the Materials are furnished to do so, subject to
** the following conditions:
**
** The above copyright notice and this permission notice shall be included
** in all copies or substantial portions of the Materials.
**
** THE MATERIALS ARE PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
** EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
** MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
** IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
** CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
** TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
** MATERIALS OR THE USE OR OTHER DEALINGS IN THE MATERIALS.
*/

// Various functions for helping debug WebGL apps.

WebGLDebugUtils = function() {

/**
 * Wrapped logging function.
 * @param {string} msg Message to log.
 */
var log = function(msg) {
  if (window.console && window.console.log) {
    window.console.log(msg);
  }
};
```

```
    /**
     * Wrapped error logging function.
40   * @param {string} msg Message to log.
     */
    var error = function(msg) {
      if (window.console && window.console.error) {
        window.console.error(msg);
45    } else {
        log(msg);
      }
    };

50  /**
     * Which arguements are enums.
     * @type {!Object.<number, string>}
     */
    var glValidEnumContexts = {
55
      // Generic setters and getters

      'enable': { 0:true },
      'disable': { 0:true },
60    'getParameter': { 0:true },

      // Rendering

      'drawArrays': { 0:true },
65    'drawElements': { 0:true, 2:true },

      // Shaders

      'createShader': { 0:true },
70    'getShaderParameter': { 1:true },
      'getProgramParameter': { 1:true },

      // Vertex attributes

75    'getVertexAttrib': { 1:true },
      'vertexAttribPointer': { 2:true },

      // Textures

80    'bindTexture': { 0:true },
      'activeTexture': { 0:true },
      'getTexParameter': { 0:true, 1:true },
      'texParameterf': { 0:true, 1:true },
      'texParameteri': { 0:true, 1:true, 2:true },
85    'texImage2D': { 0:true, 2:true, 6:true, 7:true },
      'texSubImage2D': { 0:true, 6:true, 7:true },
      'copyTexImage2D': { 0:true, 2:true },
      'copyTexSubImage2D': { 0:true },
      'generateMipmap': { 0:true },
90
      // Buffer objects

      'bindBuffer': { 0:true },
```

```
    'bufferData': { 0:true, 2:true },
    'bufferSubData': { 0:true },
    'getBufferParameter': { 0:true, 1:true },

    // Renderbuffers and framebuffers

    'pixelStorei': { 0:true, 1:true },
    'readPixels': { 4:true, 5:true },
    'bindRenderbuffer': { 0:true },
    'bindFramebuffer': { 0:true },
    'checkFramebufferStatus': { 0:true },
    'framebufferRenderbuffer': { 0:true, 1:true, 2:true },
    'framebufferTexture2D': { 0:true, 1:true, 2:true },
    'getFramebufferAttachmentParameter': { 0:true, 1:true, 2:true },
    'getRenderbufferParameter': { 0:true, 1:true },
    'renderbufferStorage': { 0:true, 1:true },

    // Frame buffer operations (clear, blend, depth test, stencil)

    'clear': { 0:true },
    'depthFunc': { 0:true },
    'blendFunc': { 0:true, 1:true },
    'blendFuncSeparate': { 0:true, 1:true, 2:true, 3:true },
    'blendEquation': { 0:true },
    'blendEquationSeparate': { 0:true, 1:true },
    'stencilFunc': { 0:true },
    'stencilFuncSeparate': { 0:true, 1:true },
    'stencilMaskSeparate': { 0:true },
    'stencilOp': { 0:true, 1:true, 2:true },
    'stencilOpSeparate': { 0:true, 1:true, 2:true, 3:true },

    // Culling

    'cullFace': { 0:true },
    'frontFace': { 0:true },
};

/**
 * Map of numbers to names.
 * @type {Object}
 */
var glEnums = null;

/**
 * Initializes this module. Safe to call more than once.
 * @param {!WebGLRenderingContext} ctx A WebGL context. If
 *    you have more than one context it doesn't matter which one
 *    you pass in, it is only used to pull out constants.
 */
function init(ctx) {
  if (glEnums == null) {
    glEnums = { };
    for (var propertyName in ctx) {
      if (typeof ctx[propertyName] == 'number') {
        glEnums[ctx[propertyName]] = propertyName;
      }
    }
```

```
    }
  }

  /**
   * Checks the utils have been initialized.
   */
  function checkInit() {
    if (glEnums == null) {
      throw 'WebGLDebugUtils.init(ctx) not called';
    }
  }

  /**
   * Returns true or false if value matches any WebGL enum
   * @param {*} value Value to check if it might be an enum.
   * @return {boolean} True if value matches one of the WebGL defined enums
   */
  function mightBeEnum(value) {
    checkInit();
    return (glEnums[value] !== undefined);
  }

  /**
   * Gets an string version of an WebGL enum.
   *
   * Example:
   *   var str = WebGLDebugUtil.glEnumToString(ctx.getError());
   *
   * @param {number} value Value to return an enum for
   * @return {string} The string version of the enum.
   */
  function glEnumToString(value) {
    checkInit();
    var name = glEnums[value];
    return (name !== undefined) ? name :
        ("*UNKNOWN WebGL ENUM (0x" + value.toString(16) + ")");
  }

  /**
   * Returns the string version of a WebGL argument.
   * Attempts to convert enum arguments to strings.
   * @param {string} functionName the name of the WebGL function.
   * @param {number} argumentIndx the index of the argument.
   * @param {*} value The value of the argument.
   * @return {string} The value as a string.
   */
  function glFunctionArgToString(functionName, argumentIndex, value) {
    var funcInfo = glValidEnumContexts[functionName];
    if (funcInfo !== undefined) {
      if (funcInfo[argumentIndex]) {
        return glEnumToString(value);
      }
    }
    if (value === null) {
      return "null";
    } else if (value === undefined) {
      return "undefined";
```

```
    } else {
      return value.toString();
    }
  }

  /**
   * Converts the arguments of a WebGL function to a string.
   * Attempts to convert enum arguments to strings.
   *
   * @param {string} functionName the name of the WebGL function.
   * @param {number} args The arguments.
   * @return {string} The arguments as a string.
   */
  function glFunctionArgsToString(functionName, args) {
    // apparently we can't do args.join(",");
    var argStr = "";
    for (var ii = 0; ii < args.length; ++ii) {
      argStr += ((ii == 0) ? '' : ', ') +
          glFunctionArgToString(functionName, ii, args[ii]);
    }
    return argStr;
  };


  function makePropertyWrapper(wrapper, original, propertyName) {
    //log("wrap prop: " + propertyName);
    wrapper.__defineGetter__(propertyName, function() {
      return original[propertyName];
    });
    // TODO(gmane): this needs to handle properties that take more than
    // one value?
    wrapper.__defineSetter__(propertyName, function(value) {
      //log("set: " + propertyName);
      original[propertyName] = value;
    });
  }

  // Makes a function that calls a function on another object.
  function makeFunctionWrapper(original, functionName) {
    //log("wrap fn: " + functionName);
    var f = original[functionName];
    return function() {
      //log("call: " + functionName);
      var result = f.apply(original, arguments);
      return result;
    };
  }

  /**
   * Given a WebGL context returns a wrapped context that calls
   * gl.getError after every command and calls a function if the
   * result is not gl.NO_ERROR.
   *
   * @param {!WebGLRenderingContext} ctx The webgl context to
   *        wrap.
   * @param {!function(err, funcName, args): void} opt_onErrorFunc
   *        The function to call when gl.getError returns an
```

```
265    *        error. If not specified the default function calls
       *        console.log with a message.
       * @param {!function(funcName, args): void} opt_onFunc The
       *        function to call when each webgl function is called.
       *        You can use this to log all calls for example.
270    */
       function makeDebugContext(ctx, opt_onErrorFunc, opt_onFunc) {
         init(ctx);
         opt_onErrorFunc = opt_onErrorFunc || function(err, functionName, args) {
               // apparently we can't do args.join(",");
275            var argStr = "";
               for (var ii = 0; ii < args.length; ++ii) {
                 argStr += ((ii == 0) ? '' : ', ') +
                     glFunctionArgToString(functionName, ii, args[ii]);
               }
280            error("WebGL error "+ glEnumToString(err) + " in "+ functionName +
                     "(" + argStr + ")");
             };

         // Holds booleans for each GL error so after we get the error ourselves
285      // we can still return it to the client app.
         var glErrorShadow = { };

         // Makes a function that calls a WebGL function and then calls getError.
         function makeErrorWrapper(ctx, functionName) {
290        return function() {
             if (opt_onFunc) {
               opt_onFunc(functionName, arguments);
             }
             var result = ctx[functionName].apply(ctx, arguments);
295          var err = ctx.getError();
             if (err != 0) {
               glErrorShadow[err] = true;
               opt_onErrorFunc(err, functionName, arguments);
             }
300          return result;
           };
         }

         // Make a an object that has a copy of every property of the WebGL context
305      // but wraps all functions.
         var wrapper = {};
         for (var propertyName in ctx) {
           if (typeof ctx[propertyName] == 'function') {
              wrapper[propertyName] = makeErrorWrapper(ctx, propertyName);
310        } else {
              makePropertyWrapper(wrapper, ctx, propertyName);
            }
         }

315      // Override the getError function with one that returns our saved results.
         wrapper.getError = function() {
           for (var err in glErrorShadow) {
             if (glErrorShadow.hasOwnProperty(err)) {
               if (glErrorShadow[err]) {
320              glErrorShadow[err] = false;
                 return err;
```

```
        }
      }
    }
    return ctx.NO_ERROR;
  };

  return wrapper;
}

function resetToInitialState(ctx) {
  var numAttribs = ctx.getParameter(ctx.MAX_VERTEX_ATTRIBS);
  var tmp = ctx.createBuffer();
  ctx.bindBuffer(ctx.ARRAY_BUFFER, tmp);
  for (var ii = 0; ii < numAttribs; ++ii) {
    ctx.disableVertexAttribArray(ii);
    ctx.vertexAttribPointer(ii, 4, ctx.FLOAT, false, 0, 0);
    ctx.vertexAttrib1f(ii, 0);
  }
  ctx.deleteBuffer(tmp);

  var numTextureUnits = ctx.getParameter(ctx.MAX_TEXTURE_IMAGE_UNITS);
  for (var ii = 0; ii < numTextureUnits; ++ii) {
    ctx.activeTexture(ctx.TEXTURE0 + ii);
    ctx.bindTexture(ctx.TEXTURE_CUBE_MAP, null);
    ctx.bindTexture(ctx.TEXTURE_2D, null);
  }

  ctx.activeTexture(ctx.TEXTURE0);
  ctx.useProgram(null);
  ctx.bindBuffer(ctx.ARRAY_BUFFER, null);
  ctx.bindBuffer(ctx.ELEMENT_ARRAY_BUFFER, null);
  ctx.bindFramebuffer(ctx.FRAMEBUFFER, null);
  ctx.bindRenderbuffer(ctx.RENDERBUFFER, null);
  ctx.disable(ctx.BLEND);
  ctx.disable(ctx.CULL_FACE);
  ctx.disable(ctx.DEPTH_TEST);
  ctx.disable(ctx.DITHER);
  ctx.disable(ctx.SCISSOR_TEST);
  ctx.blendColor(0, 0, 0, 0);
  ctx.blendEquation(ctx.FUNC_ADD);
  ctx.blendFunc(ctx.ONE, ctx.ZERO);
  ctx.clearColor(0, 0, 0, 0);
  ctx.clearDepth(1);
  ctx.clearStencil(-1);
  ctx.colorMask(true, true, true, true);
  ctx.cullFace(ctx.BACK);
  ctx.depthFunc(ctx.LESS);
  ctx.depthMask(true);
  ctx.depthRange(0, 1);
  ctx.frontFace(ctx.CCW);
  ctx.hint(ctx.GENERATE_MIPMAP_HINT, ctx.DONT_CARE);
  ctx.lineWidth(1);
  ctx.pixelStorei(ctx.PACK_ALIGNMENT, 4);
  ctx.pixelStorei(ctx.UNPACK_ALIGNMENT, 4);
  ctx.pixelStorei(ctx.UNPACK_FLIP_Y_WEBGL, false);
  ctx.pixelStorei(ctx.UNPACK_PREMULTIPLY_ALPHA_WEBGL, false);
  // TODO: Delete this IF.
```

```
    if (ctx.UNPACK_COLORSPACE_CONVERSION_WEBGL) {
      ctx.pixelStorei(ctx.UNPACK_COLORSPACE_CONVERSION_WEBGL, ctx.BROWSER_DEFAULT_WEBGL);
    }
    ctx.polygonOffset(0, 0);
    ctx.sampleCoverage(1, false);
    ctx.scissor(0, 0, ctx.canvas.width, ctx.canvas.height);
    ctx.stencilFunc(ctx.ALWAYS, 0, 0xFFFFFFFF);
    ctx.stencilMask(0xFFFFFFFF);
    ctx.stencilOp(ctx.KEEP, ctx.KEEP, ctx.KEEP);
    ctx.viewport(0, 0, ctx.canvas.width, ctx.canvas.height);
    ctx.clear(ctx.COLOR_BUFFER_BIT | ctx.DEPTH_BUFFER_BIT | ctx.STENCIL_BUFFER_BIT);

    // TODO: This should NOT be needed but Firefox fails with 'hint'
    while(ctx.getError());
  }

  function makeLostContextSimulatingCanvas(canvas) {
    var unwrappedContext_;
    var wrappedContext_;
    var onLost_ = [];
    var onRestored_ = [];
    var wrappedContext_ = {};
    var contextId_ = 1;
    var contextLost_ = false;
    var resourceId_ = 0;
    var resourceDb_ = [];
    var numCallsToLoseContext_ = 0;
    var numCalls_ = 0;
    var canRestore_ = false;
    var restoreTimeout_ = 0;

    // Holds booleans for each GL error so can simulate errors.
    var glErrorShadow_ = { };

    canvas.getContext = function(f) {
      return function() {
        var ctx = f.apply(canvas, arguments);
        // Did we get a context and is it a WebGL context?
        if (ctx instanceof WebGLRenderingContext) {
          if (ctx != unwrappedContext_) {
            if (unwrappedContext_) {
              throw "got different context"
            }
            unwrappedContext_ = ctx;
            wrappedContext_ = makeLostContextSimulatingContext(unwrappedContext_);
          }
          return wrappedContext_;
        }
        return ctx;
      }
    }(canvas.getContext);

    function wrapEvent(listener) {
      if (typeof(listener) == "function") {
        return listener;
```

```
    } else {
      return function(info) {
        listener.handleEvent(info);
      }
    }
  }

  var addOnContextLostListener = function(listener) {
    onLost_.push(wrapEvent(listener));
  };

  var addOnContextRestoredListener = function(listener) {
    onRestored_.push(wrapEvent(listener));
  };


  function wrapAddEventListener(canvas) {
    var f = canvas.addEventListener;
    canvas.addEventListener = function(type, listener, bubble) {
      switch (type) {
        case 'webglcontextlost':
          addOnContextLostListener(listener);
          break;
        case 'webglcontextrestored':
          addOnContextRestoredListener(listener);
          break;
        default:
          f.apply(canvas, arguments);
      }
    };
  }

  wrapAddEventListener(canvas);

  canvas.loseContext = function() {
    if (!contextLost_) {
      contextLost_ = true;
      numCallsToLoseContext_ = 0;
      ++contextId_;
      while (unwrappedContext_.getError());
      clearErrors();
      glErrorShadow_[unwrappedContext_.CONTEXT_LOST_WEBGL] = true;
      var event = makeWebGLContextEvent("context lost");
      var callbacks = onLost_.slice();
      setTimeout(function() {
          //log("numCallbacks:" + callbacks.length);
          for (var ii = 0; ii < callbacks.length; ++ii) {
            //log("calling callback:" + ii);
            callbacks[ii](event);
          }
          if (restoreTimeout_ >= 0) {
            setTimeout(function() {
                canvas.restoreContext();
              }, restoreTimeout_);
          }
        }, 0);
    }
```

```
  };

  canvas.restoreContext = function() {
    if (contextLost_) {
      if (onRestored_.length) {
        setTimeout(function() {
            if (!canRestore_) {
              throw "can not restore. webglcontestlost listener did not call event.preventDefault";
            }
            freeResources();
            resetToInitialState(unwrappedContext_);
            contextLost_ = false;
            numCalls_ = 0;
            canRestore_ = false;
            var callbacks = onRestored_.slice();
            var event = makeWebGLContextEvent("context restored");
            for (var ii = 0; ii < callbacks.length; ++ii) {
              callbacks[ii](event);
            }
          }, 0);
      }
    }
  };

  canvas.loseContextInNCalls = function(numCalls) {
    if (contextLost_) {
      throw "You can not ask a lost contet to be lost";
    }
    numCallsToLoseContext_ = numCalls_ + numCalls;
  };

  canvas.getNumCalls = function() {
    return numCalls_;
  };

  canvas.setRestoreTimeout = function(timeout) {
    restoreTimeout_ = timeout;
  };

  function isWebGLObject(obj) {
    //return false;
    return (obj instanceof WebGLBuffer ||
            obj instanceof WebGLFramebuffer ||
            obj instanceof WebGLProgram ||
            obj instanceof WebGLRenderbuffer ||
            obj instanceof WebGLShader ||
            obj instanceof WebGLTexture);
  }

  function checkResources(args) {
    for (var ii = 0; ii < args.length; ++ii) {
      var arg = args[ii];
      if (isWebGLObject(arg)) {
        return arg.__webglDebugContextLostId__ == contextId_;
      }
    }
```

```
    return true;
  }

  function clearErrors() {
    var k = Object.keys(glErrorShadow_);
    for (var ii = 0; ii < k.length; ++ii) {
      delete glErrorShadow_[k];
    }
  }

  function loseContextIfTime() {
    ++numCalls_;
    if (!contextLost_) {
      if (numCallsToLoseContext_ == numCalls_) {
        canvas.loseContext();
      }
    }
  }

  // Makes a function that simulates WebGL when out of context.
  function makeLostContextFunctionWrapper(ctx, functionName) {
    var f = ctx[functionName];
    return function() {
      // log("calling:" + functionName);
      // Only call the functions if the context is not lost.
      loseContextIfTime();
      if (!contextLost_) {
        //if (!checkResources(arguments)) {
        //  glErrorShadow_[wrappedContext_.INVALID_OPERATION] = true;
        //  return;
        //}
        var result = f.apply(ctx, arguments);
        return result;
      }
    };
  }

  function freeResources() {
    for (var ii = 0; ii < resourceDb_.length; ++ii) {
      var resource = resourceDb_[ii];
      if (resource instanceof WebGLBuffer) {
        unwrappedContext_.deleteBuffer(resource);
      } else if (resource instanceof WebGLFramebuffer) {
        unwrappedContext_.deleteFramebuffer(resource);
      } else if (resource instanceof WebGLProgram) {
        unwrappedContext_.deleteProgram(resource);
      } else if (resource instanceof WebGLRenderbuffer) {
        unwrappedContext_.deleteRenderbuffer(resource);
      } else if (resource instanceof WebGLShader) {
        unwrappedContext_.deleteShader(resource);
      } else if (resource instanceof WebGLTexture) {
        unwrappedContext_.deleteTexture(resource);
      }
    }
  }

  function makeWebGLContextEvent(statusMessage) {
```

```
    return {
605   statusMessage: statusMessage,
      preventDefault: function() {
          canRestore_ = true;
        }
    };
610 }

  return canvas;

  function makeLostContextSimulatingContext(ctx) {
615   // copy all functions and properties to wrapper
    for (var propertyName in ctx) {
      if (typeof ctx[propertyName] == 'function') {
         wrappedContext_[propertyName] = makeLostContextFunctionWrapper(
             ctx, propertyName);
620    } else {
        makePropertyWrapper(wrappedContext_, ctx, propertyName);
       }
    }

625   // Wrap a few functions specially.
    wrappedContext_.getError = function() {
      loseContextIfTime();
      if (!contextLost_) {
        var err;
630     while (err = unwrappedContext_.getError()) {
          glErrorShadow_[err] = true;
        }
      }
      for (var err in glErrorShadow_) {
635     if (glErrorShadow_[err]) {
          delete glErrorShadow_[err];
          return err;
        }
      }
640   return wrappedContext_.NO_ERROR;
    };

    var creationFunctions = [
      "createBuffer",
645   "createFramebuffer",
      "createProgram",
      "createRenderbuffer",
      "createShader",
      "createTexture"
650 ];
    for (var ii = 0; ii < creationFunctions.length; ++ii) {
      var functionName = creationFunctions[ii];
      wrappedContext_[functionName] = function(f) {
        return function() {
655       loseContextIfTime();
          if (contextLost_) {
            return null;
          }
          var obj = f.apply(ctx, arguments);
660       obj.__webglDebugContextLostId__ = contextId_;
```

```
          resourceDb_.push(obj);
          return obj;
        };
      }(ctx[functionName]);
    }

    var functionsThatShouldReturnNull = [
      "getActiveAttrib",
      "getActiveUniform",
      "getBufferParameter",
      "getContextAttributes",
      "getAttachedShaders",
      "getFramebufferAttachmentParameter",
      "getParameter",
      "getProgramParameter",
      "getProgramInfoLog",
      "getRenderbufferParameter",
      "getShaderParameter",
      "getShaderInfoLog",
      "getShaderSource",
      "getTexParameter",
      "getUniform",
      "getUniformLocation",
      "getVertexAttrib"
    ];
    for (var ii = 0; ii < functionsThatShouldReturnNull.length; ++ii) {
      var functionName = functionsThatShouldReturnNull[ii];
      wrappedContext_[functionName] = function(f) {
        return function() {
          loseContextIfTime();
          if (contextLost_) {
            return null;
          }
          return f.apply(ctx, arguments);
        }
      }(wrappedContext_[functionName]);
    }

    var isFunctions = [
      "isBuffer",
      "isEnabled",
      "isFramebuffer",
      "isProgram",
      "isRenderbuffer",
      "isShader",
      "isTexture"
    ];
    for (var ii = 0; ii < isFunctions.length; ++ii) {
      var functionName = isFunctions[ii];
      wrappedContext_[functionName] = function(f) {
        return function() {
          loseContextIfTime();
          if (contextLost_) {
            return false;
          }
          return f.apply(ctx, arguments);
        }
```

```
        }(wrappedContext_[functionName]);
      }

      wrappedContext_.checkFramebufferStatus = function(f) {
        return function() {
          loseContextIfTime();
          if (contextLost_) {
            return wrappedContext_.FRAMEBUFFER_UNSUPPORTED;
          }
          return f.apply(ctx, arguments);
        };
      }(wrappedContext_.checkFramebufferStatus);

      wrappedContext_.getAttribLocation = function(f) {
        return function() {
          loseContextIfTime();
          if (contextLost_) {
            return -1;
          }
          return f.apply(ctx, arguments);
        };
      }(wrappedContext_.getAttribLocation);

      wrappedContext_.getVertexAttribOffset = function(f) {
        return function() {
          loseContextIfTime();
          if (contextLost_) {
            return 0;
          }
          return f.apply(ctx, arguments);
        };
      }(wrappedContext_.getVertexAttribOffset);

      wrappedContext_.isContextLost = function() {
        return contextLost_;
      };

      return wrappedContext_;
    }
  }

  return {
      /**
       * Initializes this module. Safe to call more than once.
       * @param {!WebGLRenderingContext} ctx A WebGL context. If
      }
     *    you have more than one context it doesn't matter which one
     *    you pass in, it is only used to pull out constants.
     */
    'init': init,

    /**
     * Returns true or false if value matches any WebGL enum
     * @param {*} value Value to check if it might be an enum.
     * @return {boolean} True if value matches one of the WebGL defined enums
     */
    'mightBeEnum': mightBeEnum,
```

```
775
        /**
         * Gets an string version of an WebGL enum.
         *
         * Example:
780      *   WebGLDebugUtil.init(ctx);
         *   var str = WebGLDebugUtil.glEnumToString(ctx.getError());
         *
         * @param {number} value Value to return an enum for
         * @return {string} The string version of the enum.
785      */
        'glEnumToString': glEnumToString,

        /**
         * Converts the argument of a WebGL function to a string.
790      * Attempts to convert enum arguments to strings.
         *
         * Example:
         *   WebGLDebugUtil.init(ctx);
         *   var str = WebGLDebugUtil.glFunctionArgToString('bindTexture', 0, gl.↙
             ↳ TEXTURE_2D);
795      *
         * would return 'TEXTURE_2D'
         *
         * @param {string} functionName the name of the WebGL function.
         * @param {number} argumentIndx the index of the argument.
800      * @param {*} value The value of the argument.
         * @return {string} The value as a string.
         */
        'glFunctionArgToString': glFunctionArgToString,

805     /**
         * Converts the arguments of a WebGL function to a string.
         * Attempts to convert enum arguments to strings.
         *
         * @param {string} functionName the name of the WebGL function.
810      * @param {number} args The arguments.
         * @return {string} The arguments as a string.
         */
        'glFunctionArgsToString': glFunctionArgsToString,

815     /**
         * Given a WebGL context returns a wrapped context that calls
         * gl.getError after every command and calls a function if the
         * result is not NO_ERROR.
         *
820      * You can supply your own function if you want. For example, if you'd like
         * an exception thrown on any GL error you could do this
         *
         *    function throwOnGLError(err, funcName, args) {
         *      throw WebGLDebugUtils.glEnumToString(err) +
825      *            " was caused by call to " + funcName;
         *    };
         *
         *    ctx = WebGLDebugUtils.makeDebugContext(
         *        canvas.getContext("webgl"), throwOnGLError);
830      *
```

```
     * @param {!WebGLRenderingContext} ctx The webgl context to wrap.
     * @param {!function(err, funcName, args): void} opt_onErrorFunc The function
     *     to call when gl.getError returns an error. If not specified the default
     *     function calls console.log with a message.
     * @param {!function(funcName, args): void} opt_onFunc The
     *     function to call when each webgl function is called. You
     *     can use this to log all calls for example.
     */
    'makeDebugContext': makeDebugContext,

    /**
     * Given a canvas element returns a wrapped canvas element that will
     * simulate lost context. The canvas returned adds the following functions.
     *
     * loseContext:
     *   simulates a lost context event.
     *
     * restoreContext:
     *   simulates the context being restored.
     *
     * lostContextInNCalls:
     *   loses the context after N gl calls.
     *
     * getNumCalls:
     *   tells you how many gl calls there have been so far.
     *
     * setRestoreTimeout:
     *   sets the number of milliseconds until the context is restored
     *   after it has been lost. Defaults to 0. Pass -1 to prevent
     *   automatic restoring.
     *
     * @param {!Canvas} canvas The canvas element to wrap.
     */
    'makeLostContextSimulatingCanvas': makeLostContextSimulatingCanvas,

    /**
     * Resets a context to the initial state.
     * @param {!WebGLRenderingContext} ctx The webgl context to
     *     reset.
     */
    'resetToInitialState': resetToInitialState
};

}();
```

27 webgl/webgl-utils.js

```
/*
 * Copyright 2010, Google Inc.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are
 * met:
 *
 *     * Redistributions of source code must retain the above copyright
 * notice, this list of conditions and the following disclaimer.
```

```
 *     * Redistributions in binary form must reproduce the above
 * copyright notice, this list of conditions and the following disclaimer
 * in the documentation and/or other materials provided with the
 * distribution.
 *     * Neither the name of Google Inc. nor the names of its
 * contributors may be used to endorse or promote products derived from
 * this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
 * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
 * DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
 * THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */


/**
 * @fileoverview This file contains functions every webgl program will need
 * a version of one way or another.
 *
 * Instead of setting up a context manually it is recommended to
 * use. This will check for success or failure. On failure it
 * will attempt to present an approriate message to the user.
 *
 *       gl = WebGLUtils.setupWebGL(canvas);
 *
 * For animated WebGL apps use of setTimeout or setInterval are
 * discouraged. It is recommended you structure your rendering
 * loop like this.
 *
 *       function render() {
 *         window.requestAnimFrame(render, canvas);
 *
 *         // do rendering
 *         ...
 *       }
 *       render();
 *
 * This will call your rendering function up to the refresh rate
 * of your display but will stop rendering if your app is not
 * visible.
 */

WebGLUtils = function() {

/**
 * Creates the HTLM for a failure message
 * @param {string} canvasContainerId id of container of th
 *        canvas.
 * @return {string} The html.
 */
```

```
var makeFailHTML = function(msg) {
  return '' +
    '<table style="background-color: #8CE; width: 100%; height: 100%;"><tr>' +
    '<td align="center">' +
    '<div style="display: table-cell; vertical-align: middle;">' +
    '<div style="">' + msg + '</div>' +
    '</div>' +
    '</td></tr></table>';
};

/**
 * Mesasge for getting a webgl browser
 * @type {string}
 */
var GET_A_WEBGL_BROWSER = '' +
  'This page requires a browser that supports WebGL.<br/>' +
  '<a href="http://get.webgl.org">Click here to upgrade your browser.</a>';

/**
 * Mesasge for need better hardware
 * @type {string}
 */
var OTHER_PROBLEM = '' +
  "It doesn't appear your computer can support WebGL.<br/>" +
  '<a href="http://get.webgl.org/troubleshooting/">Click here for more ↙
      ↳ information.</a>';

/**
 * Creates a webgl context. If creation fails it will
 * change the contents of the container of the <canvas>
 * tag to an error message with the correct links for WebGL.
 * @param {Element} canvas. The canvas element to create a
 *     context from.
 * @param {WebGLContextCreationAttirbutes} opt_attribs Any
 *     creation attributes you want to pass in.
 * @return {WebGLRenderingContext} The created context.
 */
var setupWebGL = function(canvas, opt_attribs) {
  function showLink(str) {
    var container = canvas.parentNode;
    if (container) {
      container.innerHTML = makeFailHTML(str);
    }
  };

  if (!window.WebGLRenderingContext) {
    showLink(GET_A_WEBGL_BROWSER);
    return null;
  }

  var context = create3DContext(canvas, opt_attribs);
  if (!context) {
    showLink(OTHER_PROBLEM);
  }
  return context;
};
```

```
    /**
125  * Creates a webgl context.
     * @param {!Canvas} canvas The canvas tag to get context
     *     from. If one is not passed in one will be created.
     * @return {!WebGLContext} The created context.
     */
130 var create3DContext = function(canvas, opt_attribs) {
      var names = ["webgl", "experimental-webgl", "webkit-3d", "moz-webgl"];
      var context = null;
      for (var ii = 0; ii < names.length; ++ii) {
        try {
135       context = canvas.getContext(names[ii], opt_attribs);
        } catch(e) {}
        if (context) {
          break;
        }
140   }
      return context;
    }

    return {
145   create3DContext: create3DContext,
      setupWebGL: setupWebGL
    };
    }();

150 /**
     * Provides requestAnimationFrame in a cross browser way.
     */
    window.requestAnimFrame = (function() {
      return window.requestAnimationFrame ||
155          window.webkitRequestAnimationFrame ||
             window.mozRequestAnimationFrame ||
             window.oRequestAnimationFrame ||
             window.msRequestAnimationFrame ||
             function(/* function FrameRequestCallback */ callback, /* DOMElement ↙
                 ↳ Element */ element) {
160            return window.setTimeout(callback, 1000/60);
             };
    })();

    /**
165  * Provides cancelAnimationFrame in a cross browser way.
     */
    window.cancelAnimFrame = (function() {
      return window.cancelAnimationFrame ||
             window.webkitCancelAnimationFrame ||
170          window.mozCancelAnimationFrame ||
             window.oCancelAnimationFrame ||
             window.msCancelAnimationFrame ||
             window.clearTimeout;
    })();
```