|  |  |  |  |
| --- | --- | --- | --- |
| 1 | classdef Counterphase\_Grating < handle | 1 | classdef Counterphase\_Grating < handle |
| 2 | % Focus\_Squares: Presents simple quad-pattern for aid in stimulus focusing | 2 | % Focus\_Squares: Presents simple quad-pattern for aid in stimulus focusing |
| 3 | % | 3 | % |
| 4 | % $Id: NAME VER\_ID DATA-TIME vinje $ | 4 | % $Id: NAME VER\_ID DATA-TIME vinje $ |
| 5 | % usage: NAME(Args) | 5 | % usage: NAME(Args) |
| 6 | % by: william vinje | 6 | % by: william vinje |
| 7 | % date: Date | 7 | % date: Date |
| 8 | % copyright: (c) Date William Vinje, Eduardo Jose Chichilnisky (GPL see RSM/COPYING) | 8 | % copyright: (c) Date William Vinje, Eduardo Jose Chichilnisky (GPL see RSM/COPYING) |
| 9 | % | 9 | % |
| 10 | % usage: Examples | 10 | % usage: Examples |
| 11 | % inputs: Constructor accepts stimuli definition structure. | 11 | % inputs: Constructor accepts stimuli definition structure. |
| 12 | % outputs: Constructor returns Flashing\_Color stimulus object. | 12 | % outputs: Constructor returns Flashing\_Color stimulus object. |
| 13 | % calls: | 13 | % calls: |
| 14 | % mglGetSecs | 14 | % mglGetSecs |
| 15 | % Pulse\_DigOut\_Channel | 15 | % Pulse\_DigOut\_Channel |
| 16 | % mglClearScreen | 16 | % mglClearScreen |
| 17 | % mglMakeGrating | 17 | % mglMakeGrating |
| 18 | % mglCreateTexture | 18 | % mglCreateTexture |
| 19 | % mglBltTexture | 19 | % mglBltTexture |
| 20 | % mglFlush | 20 | % mglFlush |
| 21 | % mglDeleteTexture | 21 | % mglDeleteTexture |
| 22 | % | 22 | % |
| 23 |  | 23 |  |
| 24 |  | 24 |  |
| 25 | properties | 25 | properties |
| 26 |  | 26 |  |
| 27 | % Required props for all stim classes | 27 | % Required props for all stim classes |
| 28 | stim\_name | 28 | stim\_name |
| 29 |  | 29 |  |
| 30 | run\_date\_time | 30 | run\_date\_time |
| 31 | run\_time\_total | 31 | run\_time\_total |
| 32 |  | 32 |  |
| 33 | main\_trigger | 33 | main\_trigger |
| 34 | tmain0 | 34 | tmain0 |
| 35 |  | 35 |  |
| 36 | rep\_trigger | 36 | rep\_trigger |
| 37 | trep0 | 37 | trep0 |
| 38 |  | 38 |  |
| 39 | run\_duration | 39 | run\_duration |
| 40 | stim\_update\_freq | 40 | stim\_update\_freq |
| 41 |  | 41 |  |
| 42 | run\_script | 42 | run\_script |
| 43 |  | 43 |  |
| 44 | wait\_key | 44 | wait\_key |
| 45 | wait\_trigger | 45 | wait\_trigger |
| 46 |  | 46 |  |
| 47 | n\_repeats | 47 | n\_repeats |
| 48 | repeat\_num | 48 | repeat\_num |
| 49 |  | 49 |  |
| 50 | backgrndcolor | 50 | backgrndcolor |
| 51 |  | 51 |  |
| 52 | num\_reps | 52 | num\_reps |
| 53 | n\_frames | 53 | n\_frames |
| 54 | frames\_shown | 54 | frames\_shown |
| 55 | reps\_run | 55 | reps\_run |
| 56 |  | 56 |  |
| 57 | physical\_width | 57 | physical\_width |
| 58 | physical\_height | 58 | physical\_height |
| 59 |  | 59 |  |
| 60 | color | 60 | color |
| 61 |  | 61 |  |
| 62 | phase\_t0 | 62 | phase\_t0 |
| 63 | phase\_velocity | 63 | phase\_velocity |
| 64 |  | 64 |  |
| 65 | grating\_sf\_pix % pixels per cycle | 65 | grating\_sf\_pix % pixels per cycle |
| 66 | grating\_sf\_dva % cyc per DVA (one DVA is 1 cm on screen) | 66 | grating\_sf\_dva % cyc per DVA (one DVA is 1 cm on screen) |
| 67 |  | 67 |  |
| 68 | grating\_angle | 68 | grating\_angle |
| 69 |  | 69 |  |
|  |  | 70 | interval |
| 70 |  | 71 |  |
| 71 | end % properties block | 72 | end % properties block |
| 72 |  | 73 |  |
| 73 |  | 74 |  |
| 74 |  | 75 |  |
| 75 | methods | 76 | methods |
| 76 |  | 77 |  |
| 77 | function[obj] = Counterphase\_Grating( stimuli, exp\_obj ) | 78 | function[obj] = Counterphase\_Grating( stimuli, exp\_obj ) |
| 78 |  | 79 |  |
| 79 | if ( ~isfield( stimuli, 'parsed\_S') ) | 80 | if ( ~isfield( stimuli, 'parsed\_S') ) |
| 80 | %--------------------------------------------------------------------------------------------------------------- ---- | 81 | %--------------------------------------------------------------------------------------------------------------- ---- |
| 81 | % Then we use the stimuli structure constructor mode | 82 | % Then we use the stimuli structure constructor mode |
| 82 |  | 83 |  |
| 83 | if (isfield(stimuli,'rgb')) | 84 | if (isfield(stimuli,'rgb')) |
| 84 |  | 85 |  |
| 85 | obj.color = [stimuli.rgb(1); stimuli.rgb(2); stimuli.rgb(3)]; % Note: color is rgb vector in [0-1] format | 86 | obj.color = [stimuli.rgb(1); stimuli.rgb(2); stimuli.rgb(3)]; % Note: color is rgb vector in [0-1] format |
| 86 | obj.color = Color\_Test( obj.color ); | 87 | obj.color = Color\_Test( obj.color ); |
| 87 | else | 88 | else |
| 88 | fprintf('\t RSM ERROR: red gun level ("rg") not recognized. Please define rg value and try again. \n'); | 89 | fprintf('\t RSM ERROR: red gun level ("rg") not recognized. Please define rg value and try again. \n'); |
| 89 | return | 90 | return |
| 90 | end | 91 | end |
| 91 |  | 92 |  |
| 92 |  | 93 | if (isfield(stimuli,'interval')) |
|  |  | 94 |  |
|  |  | 95 | obj.interval = stimuli.interval; % |
|  |  | 96 | else |
|  |  | 97 | fprintf('\t RSM ERROR: stimulus interval not recognized. Please define interval value and try again. \n'); |
|  |  | 98 | return |
|  |  | 99 | end |
| 93 |  | 100 |  |
| 94 | if (isfield(stimuli,'phase0')) | 101 | if (isfield(stimuli,'phase0')) |
| 95 | if (isfield(stimuli,'temporal\_period')) | 102 | if (isfield(stimuli,'temporal\_period')) |
| 96 | if (isfield(stimuli,'spatial\_period')) | 103 | if (isfield(stimuli,'spatial\_period')) |
| 97 | if (isfield(stimuli,'direction')) | 104 | if (isfield(stimuli,'direction')) |
| 98 |  | 105 |  |
| 99 | if (stimuli.direction >= 360) | 106 | if (stimuli.direction >= 360) |
| 100 | stimuli.direction = stimuli.direction - 360; | 107 | stimuli.direction = stimuli.direction - 360; |
| 101 | end | 108 | end |
| 102 |  | 109 |  |
| 103 | if (stimuli.direction < 0) | 110 | if (stimuli.direction < 0) |
| 104 | stimuli.direction = stimuli.direction + 360; | 111 | stimuli.direction = stimuli.direction + 360; |
| 105 | end | 112 | end |
| 106 |  | 113 |  |
| 107 |  | 114 |  |
| 108 | % Convention 0 deg is 3 oclock | 115 | % Convention 0 deg is 3 oclock |
| 109 | if (stimuli.direction >= 0) || (stimuli.direction < 90) | 116 | if (stimuli.direction >= 0) || (stimuli.direction < 90) |
| 110 | polarity\_sign = 1; | 117 | polarity\_sign = 1; |
| 111 |  | 118 |  |
| 112 | elseif (stimuli.direction >= 90) || (stimuli.direction < 180) | 119 | elseif (stimuli.direction >= 90) || (stimuli.direction < 180) |
| 113 | polarity\_sign = -1; | 120 | polarity\_sign = -1; |
| 114 |  | 121 |  |
| 115 | elseif (stimuli.direction >= 180) || (stimuli.direction < 270) | 122 | elseif (stimuli.direction >= 180) || (stimuli.direction < 270) |
| 116 | polarity\_sign = 1; | 123 | polarity\_sign = 1; |
| 117 |  | 124 |  |
| 118 | elseif (stimuli.direction >= 270) || (stimuli.direction < 360) | 125 | elseif (stimuli.direction >= 270) || (stimuli.direction < 360) |
| 119 | polarity\_sign = -1; | 126 | polarity\_sign = -1; |
| 120 |  | 127 |  |
| 121 | end | 128 | end |
| 122 |  | 129 |  |
| 123 | phase\_velocity = polarity\_sign \* (360 / stimuli.temporal\_period); | 130 | phase\_velocity = polarity\_sign \* (360 / stimuli.temporal\_period); |
| 124 |  | 131 |  |
| 125 | obj.phase\_t0 = stimuli.phase0; | 132 | obj.phase\_t0 = stimuli.phase0; |
| 126 | obj.phase\_velocity = phase\_velocity; | 133 | obj.phase\_velocity = phase\_velocity; |
| 127 | obj.grating\_sf\_pix = stimuli.spatial\_period; | 134 | obj.grating\_sf\_pix = stimuli.spatial\_period; |
| 128 | obj.grating\_sf\_dva = Convert\_SF2DVA( obj.grating\_sf\_pix, exp\_obj ); | 135 | obj.grating\_sf\_dva = Convert\_SF2DVA( obj.grating\_sf\_pix, exp\_obj ); |
| 129 | obj.grating\_angle = stimuli.direction -90; | 136 | obj.grating\_angle = stimuli.direction -90; |
| 130 |  | 137 |  |
| 131 | else | 138 | else |
| 132 | fprintf('\t RSM ERROR: height grating direction not recognized. Please define direction and try again. \n'); | 139 | fprintf('\t RSM ERROR: height grating direction not recognized. Please define direction and try again. \n'); |
| 133 | return | 140 | return |
| 134 | end | 141 | end |
| 135 | else | 142 | else |
| 136 | fprintf('\t RSM ERROR: spatial frequency not recognized. Please define spatial\_period and try again. \n'); | 143 | fprintf('\t RSM ERROR: spatial frequency not recognized. Please define spatial\_period and try again. \n'); |
| 137 | return | 144 | return |
| 138 | end | 145 | end |
| 139 | else | 146 | else |
| 140 | fprintf('\t RSM ERROR: temporal\_period not recognized. Please define temporal\_period value and try again. \n'); | 147 | fprintf('\t RSM ERROR: temporal\_period not recognized. Please define temporal\_period value and try again. \n'); |
| 141 | return | 148 | return |
| 142 | end | 149 | end |
| 143 | else | 150 | else |
| 144 | fprintf('\t RSM ERROR: initial phase not recognized. Please define phase0 value and try again. \n'); | 151 | fprintf('\t RSM ERROR: initial phase not recognized. Please define phase0 value and try again. \n'); |
| 145 | return | 152 | return |
| 146 | end | 153 | end |
| 147 |  | 154 |  |
| 148 |  | 155 |  |
| 149 | if (isfield(stimuli,'frames')) | 156 | if (isfield(stimuli,'frames')) |
| 150 | obj.n\_frames = stimuli.frames; | 157 | obj.n\_frames = stimuli.frames; |
| 151 | else | 158 | else |
| 152 | fprintf('\t RSM ERROR: frames not recognized. Please define frames value and try again. \n'); | 159 | fprintf('\t RSM ERROR: frames not recognized. Please define frames value and try again. \n'); |
| 153 | return | 160 | return |
| 154 | end | 161 | end |
| 155 |  | 162 |  |
| 156 | obj.n\_repeats = 1; | 163 | obj.n\_repeats = 1; |
| 157 |  | 164 |  |
| 158 | obj.backgrndcolor = [stimuli.back\_rgb(1); stimuli.back\_rgb(2); stimuli.back\_rgb(3)]; | 165 | obj.backgrndcolor = [stimuli.back\_rgb(1); stimuli.back\_rgb(2); stimuli.back\_rgb(3)]; |
| 159 |  | 166 |  |
| 160 |  | 167 |  |
| 161 | obj.num\_reps = [];%stim.num\_reps; | 168 | obj.num\_reps = [];%stim.num\_reps; |
| 162 | obj.wait\_trigger = stimuli.wait\_trigger; | 169 | obj.wait\_trigger = stimuli.wait\_trigger; |
| 163 | obj.wait\_key = stimuli.wait\_key; | 170 | obj.wait\_key = stimuli.wait\_key; |
| 164 |  | 171 |  |
| 165 |  | 172 |  |
| 166 | obj.stim\_name = 'Counterphase\_Grating'; | 173 | obj.stim\_name = 'Counterphase\_Grating'; |
| 167 |  | 174 |  |
| 168 | obj.run\_date\_time = []; | 175 | obj.run\_date\_time = []; |
| 169 | obj.run\_time\_total = []; | 176 | obj.run\_time\_total = []; |
| 170 |  | 177 |  |
| 171 | obj.stim\_update\_freq = []; % By setting this to empty we remove artificial delay in main execution while loop | 178 | obj.stim\_update\_freq = []; % By setting this to empty we remove artificial delay in main execution while loop |
| 172 |  | 179 |  |
| 173 | obj.main\_trigger = 0; | 180 | obj.main\_trigger = 0; |
| 174 | obj.tmain0 = []; | 181 | obj.tmain0 = []; |
| 175 |  | 182 |  |
| 176 | obj.rep\_trigger = 0; | 183 | obj.rep\_trigger = 0; |
| 177 | obj.trep0 = []; | 184 | obj.trep0 = []; |
| 178 |  | 185 |  |
| 179 | obj.run\_script = 'Run\_CounterPhaseLoop\_Rep( exp\_obj.stimulus );'; | 186 | obj.run\_script = 'Run\_CounterPhaseLoop\_Rep( exp\_obj.stimulus );'; |
| 180 |  | 187 |  |
| 181 | obj.reps\_run = 0; | 188 | obj.reps\_run = 0; |
| 182 |  | 189 |  |
| 183 | obj.physical\_width = exp\_obj.monitor.physical\_width; | 190 | obj.physical\_width = exp\_obj.monitor.physical\_width; |
| 184 | obj.physical\_height = exp\_obj.monitor.physical\_height; | 191 | obj.physical\_height = exp\_obj.monitor.physical\_height; |
| 185 |  | 192 |  |
| 186 | obj.repeat\_num = 0; | 193 | obj.repeat\_num = 0; |
| 187 |  | 194 |  |
| 188 | obj.frames\_shown = 0; | 195 | obj.frames\_shown = 0; |
| 189 |  | 196 |  |
| 190 |  | 197 |  |
| 191 | else | 198 | else |
| 192 | %--------------------------------------------------------------------------------------------------------------- ---- | 199 | %--------------------------------------------------------------------------------------------------------------- ---- |
| 193 | % Then we use the S\_file constructor mode | 200 | % Then we use the S\_file constructor mode |
| 194 | % Contents of parsed\_S for moving grating stim | 201 | % Contents of parsed\_S for moving grating stim |
| 195 | % This constructor creates a moving grating stimulus based upon | 202 | % This constructor creates a moving grating stimulus based upon |
| 196 | % the element of the sinusoid structure array specified by | 203 | % the element of the sinusoid structure array specified by |
| 197 | % stimuli.index. | 204 | % stimuli.index. |
| 198 |  | 205 |  |
| 199 | %parsed = | 206 | %parsed = |
| 200 |  | 207 |  |
| 201 | % spec: [1x1 struct] | 208 | % spec: [1x1 struct] |
| 202 | % type: 'REVERSING-SINUSOID' | 209 | % type: 'REVERSING-SINUSOID' |
| 203 | % orientation: 0 | 210 | % orientation: 0 |
| 204 | % frames: 480 | 211 | % frames: 480 |
| 205 | % x\_start: 0 | 212 | % x\_start: 0 |
| 206 | % x\_end: 800 | 213 | % x\_end: 800 |
| 207 | % y\_start: 0 | 214 | % y\_start: 0 |
| 208 | % y\_end: 600 | 215 | % y\_end: 600 |
| 209 |  | 216 |  |
| 210 | % sinusoids: [1x144 struct] | 217 | % sinusoids: [1x144 struct] |
| 211 | % rgb: {[0.4800] [0.4800] [0.4800]} | 218 | % rgb: {[0.4800] [0.4800] [0.4800]} |
| 212 | % spatial\_period: 16 | 219 | % spatial\_period: 16 |
| 213 | % temporal\_period: 30 | 220 | % temporal\_period: 30 |
| 214 | % spatial\_phase: 6 | 221 | % spatial\_phase: 6 |
| 215 |  | 222 |  |
| 216 |  | 223 |  |
| 217 | % The rest of the information is redundant with that found within | 224 | % The rest of the information is redundant with that found within |
| 218 | % sinusoids. FOr example, the n-th elements of the following | 225 | % sinusoids. FOr example, the n-th elements of the following |
| 219 | % arrays will contain the same information as the n-th element of | 226 | % arrays will contain the same information as the n-th element of |
| 220 | % the sinusoids structure array. | 227 | % the sinusoids structure array. |
| 221 | % spatialperiods: [1x144 double] | 228 | % spatialperiods: [1x144 double] |
| 222 | % temporalperiods: [1x144 double] | 229 | % temporalperiods: [1x144 double] |
| 223 | % rgbs: {1x144 cell} | 230 | % rgbs: {1x144 cell} |
| 224 | % spatialphases: [1x144 double] | 231 | % spatialphases: [1x144 double] |
| 225 |  | 232 |  |
| 226 | obj.color = [stimuli.parsed\_S.sinusoids(stimuli.index).rgb{1}; stimuli.parsed\_S.sinusoids(stimuli.index).rgb{2}; stimuli.parsed\_S.sinusoids(stimuli.index).rgb{3}]; % Note: color is rgb vector in [0-1] format | 233 | obj.color = [stimuli.parsed\_S.sinusoids(stimuli.index).rgb{1}; stimuli.parsed\_S.sinusoids(stimuli.index).rgb{2}; stimuli.parsed\_S.sinusoids(stimuli.index).rgb{3}]; % Note: color is rgb vector in [0-1] format |
| 227 | obj.color = Color\_Test( obj.color ); | 234 | obj.color = Color\_Test( obj.color ); |
| 228 |  | 235 |  |
| 229 | obj.phase\_t0 = stimuli.parsed\_S.sinusoids(stimuli.index).spatial\_phase; | 236 | obj.phase\_t0 = stimuli.parsed\_S.sinusoids(stimuli.index).spatial\_phase; |
| 230 |  | 237 |  |
| 231 | obj.grating\_sf\_pix = stimuli.parsed\_S.sinusoids(stimuli.index).spatial\_period; | 238 | obj.grating\_sf\_pix = stimuli.parsed\_S.sinusoids(stimuli.index).spatial\_period; |
| 232 |  | 239 |  |
| 233 | obj.grating\_sf\_dva = Convert\_SF2DVA( obj.grating\_sf\_pix, exp\_obj ); | 240 | obj.grating\_sf\_dva = Convert\_SF2DVA( obj.grating\_sf\_pix, exp\_obj ); |
| 234 |  | 241 |  |
| 235 | direction = stimuli.parsed\_S.spec.orientation; | 242 | direction = stimuli.parsed\_S.spec.orientation; |
| 236 |  | 243 |  |
| 237 | if (direction >= 360) | 244 | if (direction >= 360) |
| 238 | direction = direction - 360; | 245 | direction = direction - 360; |
| 239 | end | 246 | end |
| 240 |  | 247 |  |
| 241 | if (direction < 0) | 248 | if (direction < 0) |
| 242 | direction = direction + 360; | 249 | direction = direction + 360; |
| 243 | end | 250 | end |
| 244 |  | 251 |  |
| 245 | obj.grating\_angle = direction -90; | 252 | obj.grating\_angle = direction -90; |
| 246 |  | 253 |  |
| 247 | % Convention 0 deg is 3 oclock | 254 | % Convention 0 deg is 3 oclock |
| 248 | if (direction >= 0) || (direction < 90) | 255 | if (direction >= 0) || (direction < 90) |
| 249 | polarity\_sign = 1; | 256 | polarity\_sign = 1; |
| 250 |  | 257 |  |
| 251 | elseif (direction >= 90) || (direction < 180) | 258 | elseif (direction >= 90) || (direction < 180) |
| 252 | polarity\_sign = -1; | 259 | polarity\_sign = -1; |
| 253 |  | 260 |  |
| 254 | elseif (direction >= 180) || (direction < 270) | 261 | elseif (direction >= 180) || (direction < 270) |
| 255 | polarity\_sign = 1; | 262 | polarity\_sign = 1; |
| 256 |  | 263 |  |
| 257 | elseif (direction >= 270) || (direction < 360) | 264 | elseif (direction >= 270) || (direction < 360) |
| 258 | polarity\_sign = -1; | 265 | polarity\_sign = -1; |
| 259 |  | 266 |  |
| 260 | end | 267 | end |
| 261 |  | 268 |  |
| 262 | obj.phase\_velocity = polarity\_sign \* (360 / stimuli.parsed\_S.sinusoids(stimuli.index).temporal\_period); | 269 | obj.phase\_velocity = polarity\_sign \* (360 / stimuli.parsed\_S.sinusoids(stimuli.index).temporal\_period); |
| 263 |  | 270 |  |
| 264 |  | 271 |  |
| 265 | obj.n\_frames = stimuli.parsed\_S.spec.frames; | 272 | obj.n\_frames = stimuli.parsed\_S.spec.frames; |
| 266 |  | 273 |  |
| 267 | obj.n\_repeats = 1; % For construction via S file repeats is always set to 1. | 274 | obj.n\_repeats = 1; % For construction via S file repeats is always set to 1. |
| 268 |  | 275 |  |
| 269 | % The background color is set to the monitor default background | 276 | % The background color is set to the monitor default background |
| 270 | % color | 277 | % color |
| 271 | obj.backgrndcolor = [exp\_obj.monitor.backgrndcolor(1); exp\_obj.monitor.backgrndcolor(2); exp\_obj.monitor.backgrndcolor(3)]; | 278 | obj.backgrndcolor = [exp\_obj.monitor.backgrndcolor(1); exp\_obj.monitor.backgrndcolor(2); exp\_obj.monitor.backgrndcolor(3)]; |
| 272 | obj.backgrndcolor = Color\_Test( obj.backgrndcolor ); | 279 | obj.backgrndcolor = Color\_Test( obj.backgrndcolor ); |
| 273 |  | 280 |  |
| 274 | obj.num\_reps = [];%stim.num\_reps; | 281 | obj.num\_reps = [];%stim.num\_reps; |
| 275 | obj.wait\_trigger = 0; % We always turn off triggering for S file construction. | 282 | obj.wait\_trigger = 0; % We always turn off triggering for S file construction. |
| 276 | obj.wait\_key = 0; | 283 | obj.wait\_key = 0; |
| 277 |  | 284 |  |
| 278 |  | 285 |  |
| 279 | obj.stim\_name = 'Counterphase\_Grating'; | 286 | obj.stim\_name = 'Counterphase\_Grating'; |
| 280 |  | 287 |  |
| 281 | obj.run\_date\_time = []; | 288 | obj.run\_date\_time = []; |
| 282 | obj.run\_time\_total = []; | 289 | obj.run\_time\_total = []; |
| 283 |  | 290 |  |
| 284 | obj.stim\_update\_freq = []; % By setting this to empty we remove artificial delay in main execution while loop | 291 | obj.stim\_update\_freq = []; % By setting this to empty we remove artificial delay in main execution while loop |
| 285 |  | 292 |  |
| 286 | obj.main\_trigger = 0; | 293 | obj.main\_trigger = 0; |
| 287 | obj.tmain0 = []; | 294 | obj.tmain0 = []; |
| 288 |  | 295 |  |
| 289 | obj.rep\_trigger = 0; | 296 | obj.rep\_trigger = 0; |
| 290 | obj.trep0 = []; | 297 | obj.trep0 = []; |
| 291 |  | 298 |  |
| 292 | obj.run\_script = 'Run\_CounterPhaseLoop\_Rep( exp\_obj.stimulus );'; | 299 | obj.run\_script = 'Run\_CounterPhaseLoop\_Rep( exp\_obj.stimulus );'; |
| 293 |  | 300 |  |
| 294 | obj.reps\_run = 0; | 301 | obj.reps\_run = 0; |
| 295 |  | 302 |  |
| 296 | obj.physical\_width = exp\_obj.monitor.physical\_width; | 303 | obj.physical\_width = exp\_obj.monitor.physical\_width; |
| 297 | obj.physical\_height = exp\_obj.monitor.physical\_height; | 304 | obj.physical\_height = exp\_obj.monitor.physical\_height; |
| 298 |  | 305 |  |
| 299 | obj.repeat\_num = 0; | 306 | obj.repeat\_num = 0; |
| 300 |  | 307 |  |
| 301 | obj.frames\_shown = 0; | 308 | obj.frames\_shown = 0; |
| 302 |  | 309 |  |
| 303 | end % stimuli vs Sfile if-then | 310 | end % stimuli vs Sfile if-then |
| 304 |  | 311 |  |
| 305 | end % constructor | 312 | end % constructor |
| 306 |  | 313 |  |
| 307 |  | 314 |  |
| 308 |  | 315 |  |
| 309 |  | 316 |  |
| 310 | function Run\_CounterPhaseLoop\_Rep( obj ) | 317 | function Run\_CounterPhaseLoop\_Rep( obj ) |
| 311 |  | 318 |  |
| 312 | not\_done = 1; | 319 | not\_done = 1; |
| 313 | te = 0; | 320 | te = 0; |
| 314 | delta\_t = 0; | 321 | delta\_t = 0; |
| 315 | local\_t0 = mglGetSecs; | 322 | local\_t0 = mglGetSecs; |
| 316 | te\_last = 0; | 323 | te\_last = 0; |
| 317 | phi = obj.phase\_t0; | 324 | phi = obj.phase\_t0; |
| 318 | phi2 = obj.phase\_t0; | 325 | phi2 = obj.phase\_t0; |
| 319 |  | 326 |  |
| 320 | while( not\_done ) | 327 | while( not\_done ) |
| 321 |  | 328 |  |
| 322 | % update phase | 329 | % update phase |
| 323 | phi = phi + (obj.phase\_velocity \* delta\_t); | 330 | phi = phi + (obj.phase\_velocity \* delta\_t); |
| 324 |  | 331 |  |
| 325 | phi2 = phi2 - (obj.phase\_velocity \* delta\_t); | 332 | phi2 = phi2 - (obj.phase\_velocity \* delta\_t); |
| 326 |  | 333 |  |
| 327 | % test for pulse | 334 | % test for pulse |
| 328 | if (obj.phase\_velocity > 0) | 335 | if (obj.phase\_velocity > 0) |
| 329 | if ( phi >= (obj.phase\_t0 + 360) ) | 336 | if ( phi >= (obj.phase\_t0 + 360) ) |
| 330 | phi = phi - 360; | 337 | phi = phi - 360; |
| 331 | Pulse\_DigOut\_Channel; | 338 | Pulse\_DigOut\_Channel; |
| 332 | end | 339 | end |
| 333 | else | 340 | else |
| 334 | % Then phase\_velocity is negative | 341 | % Then phase\_velocity is negative |
| 335 | if ( phi <= (obj.phase\_t0 - 360) ) | 342 | if ( phi <= (obj.phase\_t0 - 360) ) |
| 336 | phi = phi + 360; | 343 | phi = phi + 360; |
| 337 | Pulse\_DigOut\_Channel; | 344 | Pulse\_DigOut\_Channel; |
| 338 | end | 345 | end |
| 339 |  | 346 |  |
| 340 | end | 347 | end |
| 341 |  | 348 |  |
| 342 | phi1 = phi; | 349 | phi1 = phi; |
| 343 |  | 350 |  |
| 344 |  | 351 |  |
| 345 | % test for pulse | 352 | % test for pulse |
| 346 | if (obj.phase\_velocity < 0) | 353 | if (obj.phase\_velocity < 0) |
| 347 | if ( phi2 >= (obj.phase\_t0 + 360) ) | 354 | if ( phi2 >= (obj.phase\_t0 + 360) ) |
| 348 | phi2 = phi2 - 360; | 355 | phi2 = phi2 - 360; |
| 349 | end | 356 | end |
| 350 | else | 357 | else |
| 351 | % Then phase\_velocity is negative | 358 | % Then phase\_velocity is negative |
| 352 | if ( phi2 <= (obj.phase\_t0 - 360) ) | 359 | if ( phi2 <= (obj.phase\_t0 - 360) ) |
| 353 | phi2 = phi2 + 360; | 360 | phi2 = phi2 + 360; |
| 354 | end | 361 | end |
| 355 |  | 362 |  |
| 356 | end | 363 | end |
| 357 |  | 364 |  |
| 358 |  | 365 |  |
| 359 |  | 366 |  |
| 360 | % Draw the grating | 367 | % Draw the grating |
| 361 | mglClearScreen( obj.backgrndcolor ); | 368 | mglClearScreen( obj.backgrndcolor ); |
| 362 |  | 369 |  |
| 363 | % The phase switch in the phase (phi) is because | 370 | % The phase switch in the phase (phi) is because |
| 364 | % mglMakeGrating adds the phase offset; whereas EJ wants a | 371 | % mglMakeGrating adds the phase offset; whereas EJ wants a |
| 365 | % subtracted phase offset. | 372 | % subtracted phase offset. |
| 366 |  | 373 |  |
| 367 | grating1 = mglMakeGrating(obj.physical\_width/4, obj.physical\_height/4, 4 \* obj.grating\_sf\_dva, obj.grating\_angle, (-1\*phi1)); | 374 | grating1 = mglMakeGrating(obj.physical\_width/4, obj.physical\_height/4, 4 \* obj.grating\_sf\_dva, obj.grating\_angle, (-1\*phi1)); |
| 368 | grating2 = mglMakeGrating(obj.physical\_width/4, obj.physical\_height/4, 4 \* obj.grating\_sf\_dva, obj.grating\_angle, (-1\*phi2)); | 375 | grating2 = mglMakeGrating(obj.physical\_width/4, obj.physical\_height/4, 4 \* obj.grating\_sf\_dva, obj.grating\_angle, (-1\*phi2)); |
| 369 |  | 376 |  |
| 370 | grating1 = 255\*(grating1+1)/2; | 377 | grating1 = 255\*(grating1+1)/2; |
| 371 | grating2 = 255\*(grating2+1)/2; | 378 | grating2 = 255\*(grating2+1)/2; |
| 372 |  | 379 |  |
| 373 | grating = (grating1 + grating2) / 2; | 380 | grating = (grating1 + grating2) / 2; |
| 374 |  | 381 |  |
| 375 | colored\_grating = cat(3, ( (grating .\* obj.color(1)) + round(255 .\* obj.backgrndcolor(1)) ), ( (grating .\* obj.color(2)) + round(255 .\* obj.backgrndcolor(2)) ), ( (grating .\* obj.color(3)) + round(255 .\* obj.backgrndcolor(3)) )); | 382 | colored\_grating = cat(3, ( (grating .\* obj.color(1)) + round(255 .\* obj.backgrndcolor(1)) ), ( (grating .\* obj.color(2)) + round(255 .\* obj.backgrndcolor(2)) ), ( (grating .\* obj.color(3)) + round(255 .\* obj.backgrndcolor(3)) )); |
| 376 |  | 383 |  |
| 377 | tex = mglCreateTexture(colored\_grating, [], 0, {'GL\_TEXTURE\_MAG\_FILTER','GL\_LINEAR'}); | 384 | tex = mglCreateTexture(colored\_grating, [], 0, {'GL\_TEXTURE\_MAG\_FILTER','GL\_LINEAR'}); |
| 378 |  | 385 |  |
| 379 | mglBltTexture( tex, [0,0, obj.physical\_width, obj.physical\_height ] ); | 386 | mglBltTexture( tex, [0,0, obj.physical\_width, obj.physical\_height ] ); |
| 380 |  | 387 |  |
| 381 | mglFlush(); | 388 | mglFlush(); |
| 382 | obj.frames\_shown = obj.frames\_shown + 1; | 389 | obj.frames\_shown = obj.frames\_shown + 1; |
| 383 |  | 390 |  |
| 384 | mglDeleteTexture(tex); | 391 | mglDeleteTexture(tex); |
| 385 |  | 392 |  |
| 386 | % now update the elapsed time before looping again | 393 | % now update the elapsed time before looping again |
| 387 | te = mglGetSecs(local\_t0); | 394 | te = mglGetSecs(local\_t0); |
| 388 |  | 395 |  |
| 389 | delta\_t = te - te\_last; | 396 | delta\_t = te - te\_last; |
| 390 | te\_last = te; | 397 | te\_last = te; |
| 391 |  | 398 |  |
| 392 | % check for done | 399 | % check for done |
| 393 | if ( obj.frames\_shown > obj.n\_frames ) | 400 | if ( obj.frames\_shown > obj.n\_frames ) |
| 394 |  | 401 |  |
| 395 | not\_done = 0; | 402 | not\_done = 0; |
| 396 |  | 403 |  |
| 397 | end % test for end | 404 | end % test for end |
| 398 | end % tight loop | 405 | end % tight loop |
|  |  | 406 |  |
|  |  | 407 | if obj.interval ~= 0 |
|  |  | 408 | mglClearScreen([.5 .5 .5]) |
|  |  | 409 | mglFlush |
|  |  | 410 | mglWaitSecs(obj.interval) |
|  |  | 411 | end |
| 399 |  | 412 |  |
| 400 | end % run single repetition of bar across screen | 413 | end % run single repetition of bar across screen |
| 401 |  | 414 |  |
| 402 |  | 415 |  |
| 403 | end % methods block | 416 | end % methods block |
| 404 |  | 417 |  |
| 405 |  | 418 |  |
| 406 | end % Moving Grating Class | 419 | end % Moving Grating Class |