|  |  |  |  |
| --- | --- | --- | --- |
| 1 | classdef Moving\_Grating < handle | 1 | classdef Moving\_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 | properties | 10 | properties |
| 11 |  | 11 |  |
| 12 | % Required props for all stim classes | 12 | % Required props for all stim classes |
| 13 | stim\_name | 13 | stim\_name |
|  |  | 14 | subtype |
| 14 |  | 15 |  |
| 15 | run\_date\_time | 16 | run\_date\_time |
| 16 | run\_time\_total | 17 | run\_time\_total |
| 17 |  | 18 |  |
| 18 | main\_trigger | 19 | main\_trigger |
| 19 | tmain0 | 20 | tmain0 |
| 20 |  | 21 |  |
| 21 | rep\_trigger | 22 | rep\_trigger |
| 22 | trep0 | 23 | trep0 |
| 23 |  | 24 |  |
| 24 | run\_duration | 25 | run\_duration |
| 25 | stim\_update\_freq | 26 | stim\_update\_freq |
| 26 |  | 27 |  |
| 27 | run\_script | 28 | run\_script |
| 28 |  | 29 |  |
| 29 | wait\_key | 30 | wait\_key |
| 30 | wait\_trigger | 31 | wait\_trigger |
| 31 |  | 32 |  |
| 32 | n\_repeats | 33 | n\_repeats |
| 33 | repeat\_num | 34 | repeat\_num |
| 34 |  | 35 |  |
| 35 | backgrndcolor | 36 | backgrndcolor |
| 36 |  | 37 |  |
| 37 | num\_reps % this controls how many reps we want to run | 38 | num\_reps % this controls how many reps we want to run |
| 38 | n\_frames | 39 | n\_frames |
| 39 | frames\_shown | 40 | frames\_shown |
| 40 | reps\_run % this records how many bar passes have already occured | 41 | reps\_run % this records how many bar passes have already occured |
| 41 |  | 42 |  |
| 42 | physical\_width | 43 | physical\_width |
| 43 | physical\_height | 44 | physical\_height |
| 44 |  | 45 |  |
| 45 | color | 46 | color |
| 46 |  | 47 |  |
| 47 | phase\_t0 | 48 | phase\_t0 |
| 48 | phase\_velocity | 49 | phase\_velocity |
|  |  | 50 | moving\_params |
| 49 |  | 51 |  |
| 50 | grating\_sf\_pix % pixels per cycle | 52 | grating\_sf\_pix % pixels per cycle |
| 51 | grating\_sf\_dva % cyc per DVA (one DVA is 1 cm on screen) | 53 | grating\_sf\_dva % cyc per DVA (one DVA is 1 cm on screen) |
| 52 |  | 54 |  |
| 53 | grating\_angle | 55 | grating\_angle |
|  |  | 56 | interval |
| 54 |  | 57 |  |
| 55 |  | 58 |  |
| 56 | end % properties block | 59 | end % properties block |
| 57 |  | 60 |  |
| 58 |  | 61 |  |
| 59 |  | 62 |  |
| 60 | methods | 63 | methods |
| 61 |  | 64 |  |
| 62 | % Constructor method | 65 | % Constructor method |
| 63 | function[obj] = Moving\_Grating( stimuli, exp\_obj ) | 66 | function[obj] = Moving\_Grating( stimuli, exp\_obj ) |
| 64 |  | 67 |  |
| 65 | if ( ~isfield( stimuli, 'parsed\_S') ) | 68 | if ( ~isfield( stimuli, 'parsed\_S') ) |
| 66 | %--------------------------------------------------------------------------------------------------------------- ---- | 69 | %--------------------------------------------------------------------------------------------------------------- ---- |
| 67 | % Then we use the stimuli structure constructor mode | 70 | % Then we use the stimuli structure constructor mode |
| 68 |  | 71 |  |
| 69 | if (isfield(stimuli,'rgb')) | 72 | if (isfield(stimuli,'rgb')) |
| 70 |  | 73 |  |
| 71 | obj.color = [stimuli.rgb(1); stimuli.rgb(2); stimuli.rgb(3)]; % Note: color is rgb vector in [0-1] format | 74 | obj.color = [stimuli.rgb(1); stimuli.rgb(2); stimuli.rgb(3)]; % Note: color is rgb vector in [0-1] format |
| 72 | obj.color = Color\_Test( obj.color ); | 75 | obj.color = Color\_Test( obj.color ); |
| 73 | else | 76 | else |
| 74 | fprintf('\t RSM ERROR: red gun level ("rg") not recognized. Please define rg value and try again. \n'); | 77 | fprintf('\t RSM ERROR: red gun level ("rg") not recognized. Please define rg value and try again. \n'); |
| 75 | return | 78 | return |
| 76 | end | 79 | end |
|  |  | 80 |  |
|  |  | 81 | if (isfield(stimuli,'subtype')) |
|  |  | 82 |  |
|  |  | 83 | obj.subtype = stimuli.subtype; % |
|  |  | 84 | else |
|  |  | 85 | fprintf('\t RSM ERROR: stimulus subtype not recognized. Please specify "sine" or "square" and try again. \n'); |
|  |  | 86 | return |
|  |  | 87 | end |
| 77 |  | 88 |  |
|  |  | 89 | if (isfield(stimuli,'interval')) |
|  |  | 90 |  |
|  |  | 91 | obj.interval = stimuli.interval; % |
|  |  | 92 | else |
|  |  | 93 | fprintf('\t RSM ERROR: stimulus interval not recognized. Please define interval value and try again. \n'); |
|  |  | 94 | return |
|  |  | 95 | end |
| 78 |  | 96 |  |
| 79 |  | 97 |  |
| 80 | if (isfield(stimuli,'phase0')) | 98 | if (isfield(stimuli,'phase0')) |
| 81 | if (isfield(stimuli,'temporal\_period')) | 99 | if (isfield(stimuli,'temporal\_period')) |
| 82 | if (isfield(stimuli,'spatial\_period')) | 100 | if (isfield(stimuli,'spatial\_period')) |
| 83 | if (isfield(stimuli,'direction')) | 101 | if (isfield(stimuli,'direction')) |
| 84 |  | 102 |  |
| 85 | if (stimuli.direction >= 360) | 103 | if (stimuli.direction >= 360) |
| 86 | stimuli.direction = stimuli.direction - 360; | 104 | stimuli.direction = stimuli.direction - 360; |
| 87 | end | 105 | end |
| 88 |  | 106 |  |
| 89 | if (stimuli.direction < 0) | 107 | if (stimuli.direction < 0) |
| 90 | stimuli.direction = stimuli.direction + 360; | 108 | stimuli.direction = stimuli.direction + 360; |
| 91 | end | 109 | end |
| 92 |  | 110 |  |
| 93 |  | 111 |  |
| 94 | % Convention 0 deg is 3 oclock | 112 | % Convention 0 deg is 3 oclock |
| 95 | if (stimuli.direction >= 0) || (stimuli.direction < 90) | 113 | if (stimuli.direction >= 0) && (stimuli.direction < 45) |
| 96 | polarity\_sign = 1; | 114 | obj.moving\_params = [1/cos(stimuli.direction/180\*pi), 0, 1, 1]; |
|  |  | 115 |  |
|  |  | 116 | elseif (stimuli.direction >= 315) && (stimuli.direction < 360) |
|  |  | 117 | obj.moving\_params = [1/cos(stimuli.direction/180\*pi), 0, 1, 1]; |
| 97 |  | 118 |  |
| 98 | elseif (stimuli.direction >= 90) || (stimuli.direction < 180) | 119 | elseif (stimuli.direction >= 45) && (stimuli.direction < 135) |
| 99 | polarity\_sign = -1; | 120 | obj.moving\_params = [0, 1/sin(stimuli.direction/180\*pi), 1, 1]; |
| 100 |  | 121 |  |
| 101 | elseif (stimuli.direction >= 180) || (stimuli.direction < 270) | 122 | elseif (stimuli.direction >= 135) && (stimuli.direction < 225) |
| 102 | polarity\_sign = 1; | 123 | obj.moving\_params = [1/cos(stimuli.direction/180\*pi), 0, 1, 1]; |
| 103 |  | 124 |  |
| 104 | elseif (stimuli.direction >= 270) || (stimuli.direction < 360) | 125 | elseif (stimuli.direction >= 225) && (stimuli.direction < 360) |
| 105 | polarity\_sign = -1; | 126 | obj.moving\_params = [0, 1/sin(stimuli.direction/180\*pi), 1, 1]; |
| 106 |  | 127 |  |
| 107 | end | 128 | end |
| 108 |  | 129 |  |
| 109 | phase\_velocity = polarity\_sign \* (360 / stimuli.temporal\_period); | 130 | obj.phase\_velocity = 360/stimuli.temporal\_period; |
| 110 |  | 131 | obj.phase\_t0 = stimuli.phase0; |
| 111 | obj.phase\_t0 = stimuli.phase0; |  |  |
| 112 | obj.phase\_velocity = phase\_velocity; |  |  |
| 113 | obj.grating\_sf\_pix = stimuli.spatial\_period; | 132 | obj.grating\_sf\_pix = stimuli.spatial\_period; |
| 114 | obj.grating\_sf\_dva = Convert\_SF2DVA( obj.grating\_sf\_pix, exp\_obj ); | 133 | obj.grating\_sf\_dva = Convert\_SF2DVA( obj.grating\_sf\_pix, exp\_obj ); |
| 115 | obj.grating\_angle = stimuli.direction -90; | 134 | obj.grating\_angle = stimuli.direction; |
| 116 |  | 135 |  |
|  |  | 136 |  |
| 117 | else | 137 | else |
| 118 | fprintf('\t RSM ERROR: height grating direction not recognized. Please define direction and try again. \n'); | 138 | fprintf('\t RSM ERROR: height grating direction not recognized. Please define direction and try again. \n'); |
| 119 | return | 139 | return |
| 120 | end | 140 | end |
| 121 | else | 141 | else |
| 122 | fprintf('\t RSM ERROR: spatial frequency not recognized. Please define spatial\_period and try again. \n'); | 142 | fprintf('\t RSM ERROR: spatial frequency not recognized. Please define spatial\_period and try again. \n'); |
| 123 | return | 143 | return |
| 124 | end | 144 | end |
| 125 | else | 145 | else |
| 126 | fprintf('\t RSM ERROR: temporal\_period not recognized. Please define temporal\_period value and try again. \n'); | 146 | fprintf('\t RSM ERROR: temporal\_period not recognized. Please define temporal\_period value and try again. \n'); |
| 127 | return | 147 | return |
| 128 | end | 148 | end |
| 129 | else | 149 | else |
| 130 | fprintf('\t RSM ERROR: initial phase not recognized. Please define phase0 value and try again. \n'); | 150 | fprintf('\t RSM ERROR: initial phase not recognized. Please define phase0 value and try again. \n'); |
| 131 | return | 151 | return |
| 132 | end | 152 | end |
| 133 |  | 153 |  |
| 134 |  | 154 |  |
| 135 | if (isfield(stimuli,'frames')) | 155 | if (isfield(stimuli,'frames')) |
| 136 | obj.n\_frames = stimuli.frames; | 156 | obj.n\_frames = stimuli.frames; |
| 137 | else | 157 | else |
| 138 | fprintf('\t RSM ERROR: frames not recognized. Please define frames value and try again. \n'); | 158 | fprintf('\t RSM ERROR: frames not recognized. Please define frames value and try again. \n'); |
| 139 | return | 159 | return |
| 140 | end | 160 | end |
| 141 |  | 161 |  |
| 142 | obj.n\_repeats = 1; | 162 | obj.n\_repeats = 1; |
| 143 |  | 163 |  |
| 144 | obj.backgrndcolor = [stimuli.back\_rgb(1); stimuli.back\_rgb(2); stimuli.back\_rgb(3)]; | 164 | obj.backgrndcolor = [stimuli.back\_rgb(1); stimuli.back\_rgb(2); stimuli.back\_rgb(3)]; |
| 145 | obj.backgrndcolor = Color\_Test( obj.backgrndcolor ); | 165 | obj.backgrndcolor = Color\_Test( obj.backgrndcolor ); |
| 146 |  | 166 |  |
| 147 | obj.num\_reps = [];%stim.num\_reps; | 167 | obj.num\_reps = [];%stim.num\_reps; |
| 148 | obj.wait\_trigger = stimuli.wait\_trigger; | 168 | obj.wait\_trigger = stimuli.wait\_trigger; |
| 149 | obj.wait\_key = stimuli.wait\_key; | 169 | obj.wait\_key = stimuli.wait\_key; |
| 150 |  | 170 |  |
| 151 | obj.stim\_name = 'Moving\_Grating'; | 171 | obj.stim\_name = 'Moving\_Grating'; |
| 152 |  | 172 |  |
| 153 | obj.run\_date\_time = []; | 173 | obj.run\_date\_time = []; |
| 154 | obj.run\_time\_total = []; | 174 | obj.run\_time\_total = []; |
| 155 |  | 175 |  |
| 156 | obj.stim\_update\_freq = []; % By setting this to empty we remove artificial delay in main execution while loop | 176 | obj.stim\_update\_freq = []; % By setting this to empty we remove artificial delay in main execution while loop |
| 157 |  | 177 |  |
| 158 | obj.main\_trigger = 0; | 178 | obj.main\_trigger = 0; |
| 159 | obj.tmain0 = []; | 179 | obj.tmain0 = []; |
| 160 |  | 180 |  |
| 161 | obj.rep\_trigger = 0; | 181 | obj.rep\_trigger = 0; |
| 162 | obj.trep0 = []; | 182 | obj.trep0 = []; |
| 163 |  | 183 |  |
| 164 | obj.run\_script = 'Run\_PhaseLoop\_Rep( exp\_obj.stimulus );'; | 184 | obj.run\_script = 'Run\_PhaseLoop\_Rep( exp\_obj.stimulus );'; |
| 165 |  | 185 |  |
| 166 | obj.reps\_run = 0; | 186 | obj.reps\_run = 0; |
| 167 |  | 187 |  |
| 168 | obj.physical\_width = exp\_obj.monitor.physical\_width; | 188 | obj.physical\_width = exp\_obj.monitor.physical\_width; |
| 169 | obj.physical\_height = exp\_obj.monitor.physical\_height; | 189 | obj.physical\_height = exp\_obj.monitor.physical\_height; |
| 170 |  | 190 |  |
| 171 | obj.repeat\_num = 0; | 191 | obj.repeat\_num = 0; |
| 172 |  | 192 |  |
| 173 | obj.frames\_shown = 0; | 193 | obj.frames\_shown = 0; |
| 174 |  | 194 |  |
| 175 | else | 195 | else |
| 176 | %--------------------------------------------------------------------------------------------------------------- ---- | 196 | %--------------------------------------------------------------------------------------------------------------- ---- |
| 177 | % Then we use the S\_file constructor mode | 197 | % Then we use the S\_file constructor mode |
| 178 | % Contents of parsed\_S for moving grating stim | 198 | % Contents of parsed\_S for moving grating stim |
| 179 | % This constructor creates a moving grating stimulus based upon | 199 | % This constructor creates a moving grating stimulus based upon |
| 180 | % the element of the sinusoid structure array specified by | 200 | % the element of the sinusoid structure array specified by |
| 181 | % stimuli.index. | 201 | % stimuli.index. |
| 182 |  | 202 |  |
| 183 | %parsed = | 203 | %parsed = |
| 184 |  | 204 |  |
| 185 | % spec: [1x1 struct] | 205 | % spec: [1x1 struct] |
| 186 | % type: 'REVERSING-SINUSOID' | 206 | % type: 'REVERSING-SINUSOID' |
| 187 | % orientation: 0 | 207 | % orientation: 0 |
| 188 | % frames: 480 | 208 | % frames: 480 |
| 189 | % x\_start: 0 | 209 | % x\_start: 0 |
| 190 | % x\_end: 800 | 210 | % x\_end: 800 |
| 191 | % y\_start: 0 | 211 | % y\_start: 0 |
| 192 | % y\_end: 600 | 212 | % y\_end: 600 |
| 193 |  | 213 |  |
| 194 | % sinusoids: [1x144 struct] | 214 | % sinusoids: [1x144 struct] |
| 195 | % rgb: {[0.4800] [0.4800] [0.4800]} | 215 | % rgb: {[0.4800] [0.4800] [0.4800]} |
| 196 | % spatial\_period: 16 | 216 | % spatial\_period: 16 |
| 197 | % temporal\_period: 30 | 217 | % temporal\_period: 30 |
| 198 | % spatial\_phase: 6 | 218 | % spatial\_phase: 6 |
| 199 |  | 219 |  |
| 200 |  | 220 |  |
| 201 | % The rest of the information is redundant with that found within | 221 | % The rest of the information is redundant with that found within |
| 202 | % sinusoids. FOr example, the n-th elements of the following | 222 | % sinusoids. FOr example, the n-th elements of the following |
| 203 | % arrays will contain the same information as the n-th element of | 223 | % arrays will contain the same information as the n-th element of |
| 204 | % the sinusoids structure array. | 224 | % the sinusoids structure array. |
| 205 | % spatialperiods: [1x144 double] | 225 | % spatialperiods: [1x144 double] |
| 206 | % temporalperiods: [1x144 double] | 226 | % temporalperiods: [1x144 double] |
| 207 | % rgbs: {1x144 cell} | 227 | % rgbs: {1x144 cell} |
| 208 | % spatialphases: [1x144 double] | 228 | % spatialphases: [1x144 double] |
| 209 |  | 229 |  |
| 210 | 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 | 230 | 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 |
| 211 | obj.color = Color\_Test( obj.color ); | 231 | obj.color = Color\_Test( obj.color ); |
| 212 |  | 232 |  |
| 213 | obj.phase\_t0 = stimuli.parsed\_S.sinusoids(stimuli.index).spatial\_phase; | 233 | obj.phase\_t0 = stimuli.parsed\_S.sinusoids(stimuli.index).spatial\_phase; |
| 214 |  | 234 |  |
| 215 | obj.grating\_sf\_pix = stimuli.parsed\_S.sinusoids(stimuli.index).spatial\_period; | 235 | obj.grating\_sf\_pix = stimuli.parsed\_S.sinusoids(stimuli.index).spatial\_period; |
| 216 |  | 236 |  |
| 217 | obj.grating\_sf\_dva = Convert\_SF2DVA( obj.grating\_sf\_pix, exp\_obj ); | 237 | obj.grating\_sf\_dva = Convert\_SF2DVA( obj.grating\_sf\_pix, exp\_obj ); |
| 218 |  | 238 |  |
| 219 | direction = stimuli.parsed\_S.spec.orientation; | 239 | direction = stimuli.parsed\_S.spec.orientation; |
| 220 |  | 240 |  |
| 221 | if (direction >= 360) | 241 | if (direction >= 360) |
| 222 | direction = direction - 360; | 242 | direction = direction - 360; |
| 223 | end | 243 | end |
| 224 |  | 244 |  |
| 225 | if (direction < 0) | 245 | if (direction < 0) |
| 226 | direction = direction + 360; | 246 | direction = direction + 360; |
| 227 | end | 247 | end |
| 228 |  | 248 |  |
| 229 | obj.grating\_angle = direction -90; | 249 | obj.grating\_angle = direction -90; |
| 230 |  | 250 |  |
| 231 | % Convention 0 deg is 3 oclock | 251 | % Convention 0 deg is 3 oclock |
| 232 | if (direction >= 0) || (direction < 90) | 252 | if (direction >= 0) || (direction < 90) |
| 233 | polarity\_sign = 1; | 253 | polarity\_sign = 1; |
| 234 |  | 254 |  |
| 235 | elseif (direction >= 90) || (direction < 180) | 255 | elseif (direction >= 90) || (direction < 180) |
| 236 | polarity\_sign = -1; | 256 | polarity\_sign = -1; |
| 237 |  | 257 |  |
| 238 | elseif (direction >= 180) || (direction < 270) | 258 | elseif (direction >= 180) || (direction < 270) |
| 239 | polarity\_sign = 1; | 259 | polarity\_sign = 1; |
| 240 |  | 260 |  |
| 241 | elseif (direction >= 270) || (direction < 360) | 261 | elseif (direction >= 270) || (direction < 360) |
| 242 | polarity\_sign = -1; | 262 | polarity\_sign = -1; |
| 243 |  | 263 |  |
| 244 | end | 264 | end |
| 245 |  | 265 |  |
| 246 | obj.phase\_velocity = polarity\_sign \* (360 / stimuli.parsed\_S.sinusoids(stimuli.index).temporal\_period); | 266 | obj.phase\_velocity = polarity\_sign \* (360 / stimuli.parsed\_S.sinusoids(stimuli.index).temporal\_period); |
| 247 |  | 267 |  |
| 248 |  | 268 |  |
| 249 | obj.n\_frames = stimuli.parsed\_S.spec.frames; | 269 | obj.n\_frames = stimuli.parsed\_S.spec.frames; |
| 250 |  | 270 |  |
| 251 | obj.n\_repeats = 1; % For construction via S file repeats is always set to 1. | 271 | obj.n\_repeats = 1; % For construction via S file repeats is always set to 1. |
| 252 |  | 272 |  |
| 253 | % The background color is set to the monitor default background | 273 | % The background color is set to the monitor default background |
| 254 | % color | 274 | % color |
| 255 | obj.backgrndcolor = [exp\_obj.monitor.backgrndcolor(1); exp\_obj.monitor.backgrndcolor(2); exp\_obj.monitor.backgrndcolor(3)]; | 275 | obj.backgrndcolor = [exp\_obj.monitor.backgrndcolor(1); exp\_obj.monitor.backgrndcolor(2); exp\_obj.monitor.backgrndcolor(3)]; |
| 256 | obj.backgrndcolor = Color\_Test( obj.backgrndcolor ); | 276 | obj.backgrndcolor = Color\_Test( obj.backgrndcolor ); |
| 257 |  | 277 |  |
| 258 | obj.num\_reps = [];%stim.num\_reps; | 278 | obj.num\_reps = [];%stim.num\_reps; |
| 259 | obj.wait\_trigger = 0; % We always turn off triggering for S file construction. | 279 | obj.wait\_trigger = 0; % We always turn off triggering for S file construction. |
| 260 | obj.wait\_key = 0; | 280 | obj.wait\_key = 0; |
| 261 |  | 281 |  |
| 262 |  | 282 |  |
| 263 | obj.stim\_name = 'Moving\_Grating'; | 283 | obj.stim\_name = 'Moving\_Grating'; |
| 264 |  | 284 |  |
| 265 | obj.run\_date\_time = []; | 285 | obj.run\_date\_time = []; |
| 266 | obj.run\_time\_total = []; | 286 | obj.run\_time\_total = []; |
| 267 |  | 287 |  |
| 268 | obj.stim\_update\_freq = []; % By setting this to empty we remove artificial delay in main execution while loop | 288 | obj.stim\_update\_freq = []; % By setting this to empty we remove artificial delay in main execution while loop |
| 269 |  | 289 |  |
| 270 | obj.main\_trigger = 0; | 290 | obj.main\_trigger = 0; |
| 271 | obj.tmain0 = []; | 291 | obj.tmain0 = []; |
| 272 |  | 292 |  |
| 273 | obj.rep\_trigger = 0; | 293 | obj.rep\_trigger = 0; |
| 274 | obj.trep0 = []; | 294 | obj.trep0 = []; |
| 275 |  | 295 |  |
| 276 | obj.run\_script = 'Run\_PhaseLoop\_Rep( exp\_obj.stimulus );'; | 296 | obj.run\_script = 'Run\_PhaseLoop\_Rep( exp\_obj.stimulus );'; |
| 277 |  | 297 |  |
| 278 | obj.reps\_run = 0; | 298 | obj.reps\_run = 0; |
| 279 |  | 299 |  |
| 280 | obj.physical\_width = exp\_obj.monitor.physical\_width; | 300 | obj.physical\_width = exp\_obj.monitor.physical\_width; |
| 281 | obj.physical\_height = exp\_obj.monitor.physical\_height; | 301 | obj.physical\_height = exp\_obj.monitor.physical\_height; |
| 282 |  | 302 |  |
| 283 | obj.repeat\_num = 0; | 303 | obj.repeat\_num = 0; |
| 284 |  | 304 |  |
| 285 | obj.frames\_shown = 0; | 305 | obj.frames\_shown = 0; |
| 286 |  | 306 |  |
| 287 | end % stimuli vs Sfile if-then | 307 | end % stimuli vs Sfile if-then |
| 288 |  | 308 |  |
| 289 | end % constructor | 309 | end % constructor |
| 290 |  | 310 |  |
| 291 |  | 311 |  |
| 292 |  | 312 |  |
| 293 | function Run\_PhaseLoop\_Rep( obj ) | 313 | function Run\_PhaseLoop\_Rep( obj ) |
| 294 |  | 314 |  |
| 295 | not\_done = 1; | 315 | not\_done = 1; |
| 296 | te = 0; | 316 | te = 0; |
| 297 | delta\_t = 0; | 317 | delta\_t = 0; |
| 298 | local\_t0 = mglGetSecs; | 318 | local\_t0 = mglGetSecs; |
| 299 | te\_last = 0; | 319 | te\_last = 0; |
| 300 | phi = obj.phase\_t0; | 320 | phi = obj.phase\_t0; |
| 301 |  | 321 |  |
|  |  | 322 |  |
|  |  | 323 | texWidth = 2 \* obj.grating\_sf\_dva + obj.physical\_width + obj.physical\_height; |
|  |  | 324 | numCycles = ceil(obj.grating\_sf\_dva\*texWidth/2)\*2; |
|  |  | 325 | texWidth = numCycles/obj.grating\_sf\_dva; |
|  |  | 326 | texHeight = texWidth; |
|  |  | 327 |  |
|  |  | 328 | % convert to pixels |
|  |  | 329 | texWidthPixels = round(mglGetParam('xDeviceToPixels')\*texWidth); |
|  |  | 330 | texHeightPixels = round(mglGetParam('yDeviceToPixels')\*texHeight); |
|  |  | 331 |  |
|  |  | 332 | direction = obj.grating\_angle; |
|  |  | 333 |  |
|  |  | 334 | switch obj.subtype |
|  |  | 335 | case 'square' |
|  |  | 336 | grating = 255\*(sign(sin((0:numCycles\*2\*pi/(texWidthPixels-1):numCycles\*2\*pi) + obj.phase\_t0))+1)/2; |
|  |  | 337 | case 'sine' |
|  |  | 338 | grating = 255\*(sin((0:numCycles\*2\*pi/(texWidthPixels-1):numCycles\*2\*pi) + obj.phase\_t0)+1)/2; |
|  |  | 339 | otherwise |
|  |  | 340 | fprintf('\t RSM ERROR: Invalid stimulus subtype. Please specify "sine" or "square" and try again. \n'); |
|  |  | 341 | return |
|  |  | 342 | end |
|  |  | 343 | 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)) )); |
|  |  | 344 | tex1dsquare = mglCreateTexture(colored\_grating); |
|  |  | 345 |  |
|  |  | 346 |  |
| 302 | while( not\_done ) | 347 | while( not\_done ) |
| 303 |  | 348 |  |
| 304 | % update phase | 349 | % update phase |
| 305 | phi = phi + (obj.phase\_velocity \* delta\_t); | 350 | phi = phi + (obj.phase\_velocity \* delta\_t); |
| 306 |  | 351 |  |
| 307 |  |  |  |
| 308 | % test for pulse | 352 | % test for pulse |
| 309 | if (obj.phase\_velocity > 0) | 353 | if ( phi >= (obj.phase\_t0 + 360) ) |
| 310 | if ( phi >= (obj.phase\_t0 + 360) ) | 354 | phi = phi - 360; |
| 311 | phi = phi - 360; | 355 | Pulse\_DigOut\_Channel; |
| 312 | Pulse\_DigOut\_Channel; |  |  |
| 313 | end |  |  |
| 314 | else |  |  |
| 315 | % Then phase\_velocity is negative |  |  |
| 316 | if ( phi <= (obj.phase\_t0 - 360) ) |  |  |
| 317 | phi = phi + 360; |  |  |
| 318 | Pulse\_DigOut\_Channel; |  |  |
| 319 | end |  |  |
| 320 |  |  |  |
| 321 | end | 356 | end |
| 322 |  | 357 |  |
| 323 | % Draw the grating | 358 | % Draw the grating |
| 324 | mglClearScreen( obj.backgrndcolor ); | 359 | mglClearScreen( obj.backgrndcolor ); |
| 325 |  | 360 |  |
| 326 | % The phase switch in the phase (phi) is because | 361 | % The phase switch in the phase (phi) is because |
| 327 | % mglMakeGrating adds the phase offset; whereas we want a | 362 | % mglMakeGrating adds the phase offset; whereas we want a |
| 328 | % subtracted phase offset. | 363 | % subtracted phase offset. |
| 329 | grating = mglMakeGrating(obj.physical\_width/4, obj.physical\_height/4, 4 \* obj.grating\_sf\_dva, obj.grating\_angle, (-1\*phi)); |  |  |
| 330 |  |  |  |
| 331 | grating = 255\*(grating+1)/2; |  |  |
| 332 |  |  |  |
| 333 |  |  |  |
| 334 | 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)) )); |  |  |
| 335 |  | 364 |  |
| 336 | tex = mglCreateTexture(colored\_grating, [], 0, {'GL\_TEXTURE\_MAG\_FILTER','GL\_LINEAR'}); | 365 | pos = phi/360/obj.grating\_sf\_dva; |
| 337 |  | 366 | mglBltTexture( tex1dsquare, [pos pos nan texHeight].\*obj.moving\_params, 0, 0, direction ); |
| 338 | mglBltTexture( tex, [0,0, obj.physical\_width, obj.physical\_height ] ); |  |  |
| 339 |  | 367 |  |
| 340 | mglFlush(); | 368 | mglFlush(); |
| 341 | obj.frames\_shown = obj.frames\_shown + 1; | 369 | obj.frames\_shown = obj.frames\_shown + 1; |
| 342 |  |  |  |
| 343 | mglDeleteTexture(tex); |  |  |
| 344 |  |  |  |
| 345 | % now update the elapsed time before looping again | 370 | % now update the elapsed time before looping again |
| 346 | te = mglGetSecs(local\_t0); | 371 | te = mglGetSecs(local\_t0); |
| 347 |  | 372 |  |
| 348 | delta\_t = te - te\_last; | 373 | delta\_t = te - te\_last; |
| 349 | te\_last = te; | 374 | te\_last = te; |
| 350 |  | 375 |  |
| 351 | % check for done | 376 | % check for done |
| 352 | if ( obj.frames\_shown > obj.n\_frames ) | 377 | if ( obj.frames\_shown > obj.n\_frames ) |
| 353 |  | 378 |  |
| 354 | not\_done = 0; | 379 | not\_done = 0; |
| 355 |  | 380 |  |
| 356 | end % test for end | 381 | end % test for end |
| 357 |  | 382 |  |
| 358 | end % tight loop | 383 | end % tight loop |
| 359 |  | 384 |  |
|  |  | 385 |  |
|  |  | 386 | if obj.interval ~= 0 |
|  |  | 387 | mglClearScreen([.5 .5 .5]) |
|  |  | 388 | mglFlush(); |
|  |  | 389 | mglWaitSecs(obj.interval) |
|  |  | 390 | end |
|  |  | 391 |  |
| 360 | end % run single repetition of bar across screen | 392 | end % run single repetition of bar across screen |
| 361 |  | 393 |  |
| 362 |  | 394 |  |
| 363 | end % methods block | 395 | end % methods block |
| 364 |  | 396 |  |
| 365 |  | 397 |  |
| 366 | end % Moving Grating Class | 398 | end % Moving Grating Class |