SimonGrataGen User Manual

SimonGrataGen was designed by Michael Lindner and Doug Saddy and programmed by Michael Lindner

m.lindner@reading.ac.uk

University of Reading, United Kingdom

School of Psychology and Clinical Language Sciences

Center for Integrative Neuroscience and Neurodynamics

https://www.reading.ac.uk/cinn/cinn-home.aspx

Content

1. Dependencies:

- Matlab 2015a or newer (www.mathworks.com)
- Psychtoolbox 3 http://psychtoolbox.org/download/
- For sending output tiggers via LPT on Windows (2000/XP/Vista/7) PCs you need to have the following io tools installed:
    - For Win (2000/XP/Vista/7) 32bit: http://apps.usd.edu/coglab/psyc770/IO32.html
        - add the files **io32.mexw32, outp32.m and config_io32.m** to a directory in your MATLAB path
        - put the file **inpoutx32.dll** to the C:\windows\system32 directory
    - For Win (2000/XP/Vista/7) 64bit: http://apps.usd.edu/coglab/psyc770/IO64.html
        - add the files **io64.mexw64, outp.m and config_io.m** to a directory in your MATLAB path
        - put the file **inpoutx64.dll** to the C:\windows\system32 directory
    - For Linux (Ubuntu) 64bit: https://github.com/widmann/ppdev-mex
        - add the files **lptwrite.m, ppdev_mex.m and ppdev_mex.mexa64** to a directory in your MATLAB path

2. Installation

- Copy the file onto a folder on your hard drive. e.g. C:\SimonGrataGen
- Add path to the files using „Set Path“ button or the „addpath“ function. e.g. addpath('C:\SimonGrataGen')

3. License:

4. SimonGrataGen

SimonGrataGen, the **Simon gra**mmar **ta**sk **gen**erator is a tool thought to easily generate Simon tasks and to run these tasks.

First steps:

- Start the SimonGrataGen.p function either by typing in *run SimonGrataGen.p* in the Matlab command window or select the SimonGrataGen.p in the Matlabs current folder window and press F9.
- From here you will have two choices (via two buttons):
  - Simon Grammar Task Generator
    - Using the Generator you can generate different types of Simon tasks. See detailed description in section 5.
  - Simon Grammar Task Experiment
    - Using the Experiment button you can start a predefined Experiment. See detailed description in section 6.

No_sync_test version:

In the SimonGrate_no_sync_test version the Sync tests in Psychtoolbox are disabled. This is ONLY a TEST VERSION and is NOT RECOMMENDED for running your experiment, because the timing of the stimulus presentation and the recording of the button presses are not reliable! For further details See Psychtoolbox homepage: http://psychtoolbox.org/ .

Enjoy playing around with this tool!

5. Simon Grammar Task Generator

With the Simon Grammar Task Generator you can set up different types of Simon tasks with a few button presses.

For example, you can generate Simon tasks using different grammars, or random orders or a mix of both. Additionally you can use the grammars (or random order) on the positions of the dots or on its colour. Furthermore, you can specify the usage of distractors. You can use input and output triggers and you can change all possible colours. And many more options are available. See a full list below.

5.1 Input parameters:

- **Name of experiment**: The parameter name of experiment is used as prefix for the filename for the output file in which all the specified parameters will be saved. It will also be used later when you run the experiment for the filename of the output file of each subject.
- **Number of blocks**: Here you can define how many blocks of equal or different grammars you want to use.
- **Number of trials per block**: Specify the number of trials of each of the blocks.
- **Number of subjects**: Specify the number of subjects that you want to run the experiment the generator will predefine the blocks and trials for all of the subjects:
- **Number of grammars**: You can specify how many different grammars you want to use in the experiment.
- **Order of grammars in blocks**: Specify the grammar that you will use for each block. The numbers from 1 to N represent the grammars from 1 to N you specify. Zeros represent random orders of the trials (instead of grammars).

  e.g. Grammar order: 0 1 1 1 1 2

  The first block will be a random order of conditions, in the blocks 2 to 5 grammar 1 will be used and in the last block grammar 2 will be used.
- **Input grammars**: As many input dialogs occur as number of grammars you specified before. Here you need to copy in the grammars (e.g. from the Lindenmayer System Explorer "LSEx") for the specified blocks.

Rules for the grammars:

1. The grammars can only include 0s and 1s

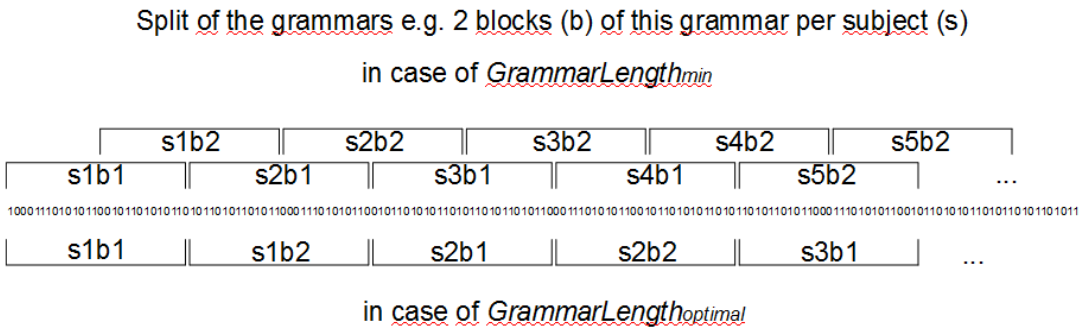
2. The grammars need to be long enough. The length of the grammar is important: The grammar need to have a minimum length of:

   $GrammarLength_{min}$ = *((Number of subjects * Number of trials per block * Number of blocks of that grammar)/2) + Number of trials per block*

   and a optimal length of:

   $GrammarLength_{optimal}$ = *(Number of subjects * Number of trials per block * Number of blocks of that grammar) + Number of trials per block.*

   The grammar will be cut into the different blocks for the different subjects (see figure 1), so that all subjects will get a different string of the grammar to prevent possible effects of specific parts of the grammar.



- **Random proportion of blocks:** For blocks with random orders of conditions (0s in the order list) you can specify the amount of 1s. e.g. to adjust the proportion to your grammar to prevent different effects depending on different proportions of conditions between the blocks and leading to e,g, different expectations or habituations of the subject.

- **Random or pseudorandom**: The random blocks can be identical for each subject (pseudorandom) or for each subject a different random order can be generated (random)

- **Use Grammar for:** Here you can specify if the grammar should be used for the positioning of the dots on the screen (Position) or for the changes of the color of the dots (Color)
- **Distractor every nth trial**: Here you can specify if a distractor should be used (values > 0). If you have chosen before that the grammar should be used for the position the distractor will be a change of the dot colour. Therefore, every nth trial the dot will have a different colour.

Depending on the user choices of the two previous parameter the following option will be shown:

- **If Grammar is used for Position:**
  - **horizontal or vertical task:** Specify the changes of the dot position related the grammar:
    - horizontal (left vs right) or
    - vertical (up vs down)
  - **Distance from center:** Specify where how far the dots should be presented from the center of the screen (in pixel)
- **If Grammar is used for Color:**
  - **If Distractors == 0**
    - **Dot Position:** Specify where the dots should be presented on the screen in relation to the center. (Only left vs right positions adjustment possible) (in pixel)
  - **If Distractors > 0**
    - **horizontal, vertical or circular distractor Position:** Specify if the distractors should be presented always on one horizontally shifted or vertically shifted position. With the circular option the distractor will be presented on random position on a circle around the main dot position.
    - **Distractor position (distance from center in pixel):** amount of horizontal or vertical shift or radius of the circle in the circular option.

- **Stimulus presentation time**: The dots can be presented until the participant presses a button (endless) or for a maximum amount of time in which the participant can press a button (limited).

  If you choose 'limited':

  - Specify the maximum presentation time in ms

- - Equal trial length? (yes or no): If button was pressed before the maximum presentation time, the trial will still wait until the maximum presentation time is over.

- **Variablity of the distractor**: If the distractor should not be fixed for each nth trial, you can specify a trial range in which the distractor differences will be chosen randomly. Example: 8 as nth trial and a variability of +/- 2 trials will lead to distractors between each 6^{th} and 10^{th} trial.

- **Background Colour**: Specify the Screen background colour (in RGB values)

- **Colour Dot1**: Specify the colour of the dot (in RGB values)

- **Colour Dot2**: Specify the colour of the second dot (for grammar used on grammar or for the distractor) (in RGB values)

- **Colour fixation point**: Specify the colour of the fixation point in the center of the screen (in RGB values)

- **Left button**: Specify the key (in keyboard) for the left button

- **Right button**: Specify the key for the left button

- **Dot size**: Specify the size of the dots (in pixel)

- **Grammar value**: Specify which value of the grammar should be used

  - for the left presentation (in case of grammar for position)

  - for colour 1 (in case of grammar for colour)

- **Present fixation point**: Use fixation point in the center of the screen. Yes or no.

- **Size of the fixation point**: Size of the fixation point if used (in pixel)

- **Length of ITI**: Specify inter trial interval time between button press and the presentation of the next dot. (in ms)
- **Variability of ITI**: If you do not want a fixed ITI time, the you can specify a range (in ms) in which a random ITI will be created for each trial separately. Example: an ITI length of 2000 ms and a variability of 200 ms will create ITIs between 1800 and 2200 ms.
- **Send output trigger**: (Yes or no) Choose yes if you want to used output triggers via LPT to other devices e.g. for EEG using the IO tools mentioned in the dependencies.

  If yes: additional dialogues will occur for specifying:

  - **Trigger type**: (smart or normal)

  In any case the marker definition will be saved as text file.

  Basic trigger:

  Start trigger: 30
  End trigger: 50
  Stop trigger: 49
  Instruction: 31
  Instruction button: 32
  Start experiment: 35
  Start Block: 40
  End Block: 41
  Start ISI: 44
  Button 1: 1
  Button 2: 2
  Button 1 correct: 10
  Button 1 incorrect: 11
  Button 2 correct: 20
  Button 2 incorrect: 21
  Stop Button: 99

  \+ if normal triggers are chosen:

  condition 0 -> 100

  condition 1 -> 200

  \+ if smart triggers are chosen:

another dialogue shows up to specify the last n trials for which the smart triggers should be defined. All different combinations of conditions in the previous n trials will get a separate trigger/marker.

Example: last n trials of a Fibonacci grammar will results in these markers:

[previous n trials] [actual trial] -> Marker50
000 0 -> 101
001 0 -> 102
010 0 -> 103
011 0 -> 104
100 0 -> 105
101 0 -> 106
110 0 -> 107
111 0 -> 108
000 1 -> 201
001 1 -> 202
010 1 -> 203
011 1 -> 204
100 1 -> 205
101 1 -> 206
110 1 -> 207
111 1 -> 208

- **Trigger port:** LPT (only LPT is supported at the moment)

Port address: If trigger via LPT was chosen you will get a dialogue for specifying the port address of the used LPT port

- **Use input trigger:** (yes or no) Input triggers can be used for starting the blocks (e.g. to synchronize with other devices (e.g. MRI))

  If yes:

  - **Input trigger type**: (only keyboard available at the moment)

  - **Input key**: Specify the keyboard key which should be used fas input trigger

- **Use instruction images:** (yes or no) Predefined images with instructions or task trials can be presented before the experiment starts.

  If yes:

  - **How many instruction images consecutively?** The number of images can be defined here.

- **Specify buttons to end each instruction image separated by semicolon.** For each instruction image the stop button can be defined (left or right).

  e.g. if you want to have left button or image 1, right button for image 2 and left button for image 3, you need to type in: left;right;left

  The number of buttons need be identical to the number of used instruction images.

  - Afterwards you need to select instruction images from the hard drive one after each other.

- **Select output directory**: Chose folder where the experiment file and the marker definition file will be saved.

  The experiment file contains a Matlab cell with all the information for all subjects, blocks and trials.

5.2 Output files

The Simon Grammar Task Generator creates one main output file. A Matlab .mat file containing a Matlab structure with all parameters needed to run the experiment. This file name is dependent on the NameOfExperiment that you defined in the first input (e.g. NameOfExperiment_parameter.mat). This file will be loaded in the Simon Grammar Task Experiment.

If you use output triggers another output file will be created: a .txt file containing the description of the Marker (e.g. NameOfExperiment_Marker.txt). This file is only an information for the user and will maybe necessary for a later analysis of the data collected with the triggered device (e.g. EEG).

6. Simon Grammar Task Experiment

With the Simon Grammar Task Experiment you can start a predefined experiment.

6.1 Task description

Each trial starts with the presentation of a dot (depending on the specified grammar and colour and with or without fixation point). Depending on the users choice, the dots will either stay until a button is pressed or until a defined time is over. The dots are followed by a blank screen (with or without fixation point) for the predefined ITI time.

A row of predefined N blocks with predefined M trials each are presented. The block will have a self paced pause, so that the participants can start the next block, when they are ready.

6.2 Input parameters

- **Select experiment parameter file**: Select the predefined parameter file of the experiment that you want to start.
- **Subject number**: Select the predefined data of subject n for the experiment you want to run.
- **Multiple Screen detected**: If you have more than one monitor detached to the PCs on which you want to run the experiment, you can specify on which monitor you want to present the experiment. Usually using value 0 the experiment will be presented on both screens (extended not duplicated!) and the value 1 to n are for the different monitor. But depending on your hardware and software setup this may differ. Therefore, you need to test it in advance on the PC on which you want to run the experiment.

6.3 Output files

The Simon Grammar Task Experiment saves a text file for each subject containing the information the following 11 information (column) of each trial (rows):

Block: The trial was present in this block number
trial: The trial was trial number of the experiment
time: Cumulative onset time of the trial in respect of the start f the experiment.
Grammar: The condition of the grammar the trial had
Distr: Was the trial a distractor? (1=yes, 0=no)
ButtonNr: Which button was pressed by the user (1=left)
ButtonPos: Position of the pressed button (Left or Right)
ACC: Was the Button press of the participant correct (1) or not (0).
RT(ms): Reaction time (time difference between dot presentation and button press)
DotCol: Colour of the dot (1 or 2)