Lab 3 - Red Light, Green Light

Today, we will be using the PsychoPy Builder along with this worksheet. Please follow along on your computer as we build an experiment! If I move too fast at any point, please stop me and I can address any questions or technical issues as we go along.

If following along, **please download the resources folder** from the BruinLearn Week 3 Lab folder.

The Premise

One of the children games featured in the popular TV show, "Squid Games" is called Red Light, Green Light. In essence, it is a signal detection game in which the response to a "Green Light" is forward movement and the response to a "Red Light" is movement suppression.

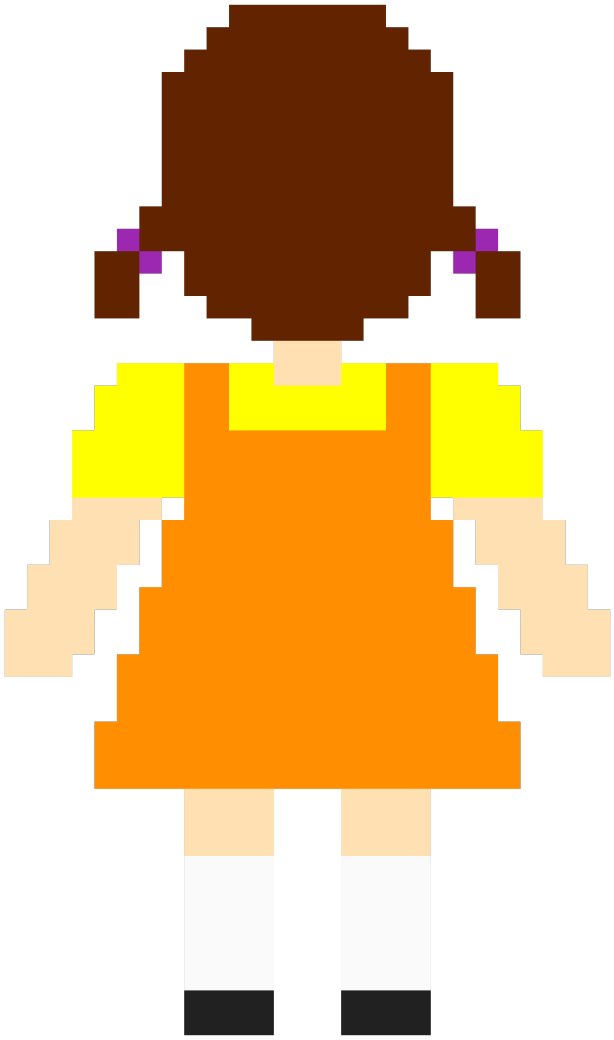
The Go/No-Go Task

We will create a "Squid Games" version of a classic cognitive psychology experiment, the **Go/No-Go Task**. The Go/No-Go paradigm is designed to test the inhibitory control of the participant. In general, these tasks involve a rapid serial presentation of stimuli. Participants have two possible response choices: push a button ("Go") or withholding a response ("No-Go"). Most of the stimuli presented are associated with a "Go" response, while "No-Go" stimuli are more infrequent. The ability to withhold a response to a "No-Go" stimulus is called **response inhibition**.

The Red Light, Green Light Game

Our version of the Go/No-Go Task will be called **Red Light, Green Light**. In this task, two different visual stimuli will be rapidly presented: an image of the Squid Games doll facing forwards and an image of the doll facing backwards. Participants should respond by pushing the spacebar whenever the doll is facing backwards ("Green Light"). Whenever the doll "turns around" and faces forward, participants should withhold their response ("Red Light").

The Red Light, Green Light stimuli:



Instructions Routine

We will need two different types of components for our instructions routine: the **Text and Keyboard components**.

Text

- Name: the variable name that you want to give to Text object
- Start: the onset time that the text will be presented
- Stop: determines the offset time (can be specified in durations or times)
- Text: contains the text that you want to display
- Letter height: the "font size" of the text
- Refer to this for more details: https://www.psychopy.org/builder/components/text.html

In general, getting the aesthetics of the text component to display nicely takes some trial and error. I usually tinker around with the Layout and Format panels to get it right.

Keyboard

- Name: the variable name that you want to give to object
- Start: the onset time that the object will be presented
- Stop: determines the offset time (can be specified in durations or times)

- Allowed keys: specify which buttons on the keyboard you want to listen to.
- Store: If more than 1 response is registered, which key do you want to store? last key, first key, all keys, nothing
- Refer to this for more details: https://www.psychopy.org/builder/components/keyboard.html

For instructions, we often set the Keyboard duration to indefinite by leaving the "Stop" field empty. This gives subjects the opportunity to take as much time as they need to read the instructions. Make sure "Force end of Routine" is checked if you want a Keyboard response to advance to the next routine.

Steps:

1. Create an `instr` routine.
2. Add an `instr_text` Text object and a `instr_kb` Keyboard object to the routine
3. Copy and paste the instruction text below to the `instr_text` object.
4. Set the stop duration of `instr_text` to 20.0 s.
5. Set the stop duration of `instr_kb` to 20.0s. Make sure "Force end of Routine is checked". Set "Allowed keys" to 'space'.
6. Finally, test and tinker with the parameters

Instruction Text:

The game that you are about to play is called Red Light, Green Light. You will see a rapid serial presentation of two dolls - one facing forward and the other facing backwards. Whenever you see a doll facing backwards ("Green Light"), you will respond by pressing the space bar as quickly as you can. Whenever you see a doll facing forwards ("Red Light"), you will withhold responding.

When you're ready, please press the spacebar to begin!

Task Trial Routine

We will now get into the meat of the experiment. In this routine, we will present one of the two Red Light, Green Light stimuli. We will also record the responses.

Image

- Name: the variable name that you want to give to object
- Start: the onset time that the object will be presented
- Stop: determines the offset time (can be specified in durations or times)
- Image: relative file path to the image (relative meaning relative to .psyexp file)
- Size: controls the width and height of the image
- Refer to this for more details: https://www.psychopy.org/builder/components/image.html

If you want to assign a different image to be presented depending on the trial, the drop-down menu next to the "Image field" needs to change to "Set every repeat". We will go over this use case today.

Loop

- Name: the variable name that you want to give to object
- nReps: number of repeats for this loop
- Refer to this for more details: https://www.psychopy.org/builder/flow.html

Loops can be applied around a single routine or multiple routine to enable iteration in your experiments. Experiments are frequently divided into blocks, which are in turn comprised of trials. Trial loops can be made by

looping around a routine. Block loops can be made by looping around trial loops. We will employ just a trial-level loop today.

Please note that there are a lot more fields in the Loop component that I won't go over today. There are many ways to use loops and to iterate through trials in your experiment, but we won't cover that today.

Custom Code

- Before Experiment: Things that need to be done just once, like importing a supporting module, which do not need the experiment window to exist yet.
- Begin Experiment: Things that need to be done just once, like initialising a variable for later use, which may need to refer to the experiment window.
- Begin Routine: Certain things might need to be done at the start of a Routine e.g. at the beginning of each trial you might decide which side a stimulus will appear.
- Each Frame: Things that need to updated constantly, throughout the experiment. Note that these will be executed exactly once per video frame (on the order of every 10ms), to give dynamic displays. Static displays do not need to be updated every frame.
- End Routine: At the end of the Routine (e.g. the trial) you may need to do additional things, like checking if the participant got the right answer
- End Experiment: Use this for things like saving data to disk, presenting a graph(?), or resetting hardware to its original state.
- Copied from this page: https://www.psychopy.org/builder/components/code.html

For the most part, I write code in the "Begin Experiment" and "Begin Routine" panels.

When you create new variables in your code components, these can be passed as values in the field boxes of the components if you use the `$` character followed by the name of the variable. This is useful if you want to programmatically assign the number of repetitions in a loop or the image file of a particular trial. We will go over both these cases today.

The Code component allows you to integrate custom code to your experiment, allowing for more robust control of your components.

Steps

1. Add an Image component called `doll_im` and a Keyboard component called `task_kb` to a new routine called `task`.
2. Create a code component. In the Begin Experiment panel, we will set up all the stimuli to be used in each iteration of the `task` routine. The goal is to build a list `im_sequence` of file names for the doll stimuli. We will also get the length of the sequence `seq_len`, which is the number of repetitions that we will have in our loop. In the Begin Routine tab, we will assign one of the strings in `im_sequence` to a different variable `doll_im_path`.
3. Wrap the `task` routine in a loop. Pass the variable `seq_len` in the nReps field.
4. Set `$doll_im_path` as the value in the Image field of `doll_im` component. Set duration to 1.0s.
5. Uncheck "Force stop routine" in `task_kb` and set duration to 1.2s.

This is a personal preference, but I like to create a separate code component for everything that I would do at the beginning of the experiment, and put add it to the first routine of the experiment. This makes it easier to keep track of where the code is located.

Note that there are many ways to create a sequence, each with their own pro's and con's. We will see a couple alternatives to generating a sequence of the image to display for each trial. This will be a list of strings, where

each element is the relative file path to the image that you want to display for each trial.

In [ ]:
```
# Approach 1 (simple hard-coding):
im_sequence = ['resources\squid_game_doll_backward', 'resources\squid_game_doll_backward',


# Approach 2 (hard-coding a list of indices that reference a filename, and then looping):
sequence = [0, 0, 0, 1, 0, 1, 0, 0, 0]
image_names = ['resources\squid_game_doll_backward',"resources\squid_game_doll_forward"]


# Approach 3 (using the numpy.tile approach):
import numpy as np
num_go_trials = 2
num_nogo_trials = 8
im_sequence = []

# Shuffle the sequence (and seed using np.random.seed; np.random.shuffle)!

im_sequence
```

Feedback Routine

We will learn to access the recorded responses and calculate the **commission error rate**, which occurs when the participant hits the space bar for a "no-go" trial (i.e., a false alarm). We will then display the commission error rate for the participants to see.

Steps

1. Create a "feedback" routine. Add a Code component, a Keyboard component called `fb_kb`, and a Text component called `fb_text`.
2. In the Begin Routine panel of the Code component, we will calculate the commission error rate of the subject. First, access the stored responses of the `task_kb` object with this line of code: `task_kb_keys = trials.data['task_kb.keys']`. Then loop through each of the recorded responses and derive the `nogo_count` (number of no-go trials) and `commission_error_count` (number of commission errors) using a nested if statement. THen calculate `commission_error_rate` from those two variables. Then build a `feedback_str` to be displayed by the text component.
3. In the text component, change duration to 20s. Add `$feedback_str` to the Text field.
4. Set up the `fb_kb` Keyboard component. Duration to 20s, allowable keys to 'space', make sure "Force end routine" is checked.

Exercises

For the remainder of class, try to work through as many of these exercises as you can! Some of these may appear in your Lab 3 homework assignment. Flag me down if you have any questions doing any of these exercises.

Exercise 1:

Try running the experiment on yourself! What is your commission error rate?

Exercise 2:

In the "resources" folder on BruinLearn, there's an image file called "squid_game_doll_both.png" that contains both the forward and backwards dolls. Please create an additional instruction routine the includes this image as well as a Text component displaying: "Below are examples of the dolls that will be presented. When you see the doll facing you, do not respond, when you see the doll facing away from you, press the spacebar as fast as you can.

Exercise 3:

Change the duration that the image is presented on the screen to 0.8s, and the duration of the inter-stimulus interval (the gap after the image offset) to 0.4s.

Exercise 4:

Let's randomize the image sequence list. To do so, we need to import the `random` library, then run `random.shuffle(im_sequence)`. (This library is automatically imported by psychopy).

Exercise 5:

I want to be able to specify the number of Go and No-Go trials. Create two new variables with the following assignments: `go_trials = 30` and `nogo_trials = 10`. Create a new `sequence` list based on these parameters, and derive the randomized list sequence.

Exercise 6:

Let's say we want the experiment to consist of two blocks, with the feedback be displayed at the end of each block. Implement this by using a block-level loop.

Exercise 7:

We now have two blocks, but are the sequences in each block different? Make the sequence for block 2 different from block 1.

Exercise 8:

We calculated commission errors already, so now let's calculate **omission errors** and add that to the feedback routine.