# Psychological Experiment Procedure for Participants

Bradley Robinson

Report for CS3430

## Abstract:

The objective of this project was to create an experimental procedure for human participants using Python, along with the Pygame library. The program would have to display text explaining the task, properly formatted, as well as image stimuli, get the users response , and record data to a spreadsheet file.

## Introduction and motivation:

As a psychology major, I have received funding and approval to implement an experimental procedure with human participants, in which they rate food items and then perform a decision-making task while electroencephalographic (EEG) measures are being made. While there is software for experimenters to design experimental procedures, many of these programs are inflexible or have other flaws (such as organizing data recordings in an unintuitive way) that can result in delayed experimentation or unrecorded data after system crashes. By using Python, I could bypass many of these problems, and have much more control over the experimental procedure.

## Background Information:

The purpose of the study utilizing this experimental procedure is to provide both behavioral and neural insights into how humans make decisions when faced with a social dilemma. A social dilemma is where an individual can choose something better for themselves or something that benefits the group, but cannot choose both.

In this procedure, participants would rate a number of food items by preference. Then, partly based on these ratings, participants would be asked to decide between purchasing a more expensive (but more preferred item) vs a cheaper item where the decision affects a joint account with another player. After gathering both the neural activity via EEG and the behavioral data (reaction times and responses) of a sufficient number of participants, the data generated from the experimental procedure would make it possible to analyze the data to see if there are patterns that give insight into the decision-making process of social dilemmas.

## Problem Description:

To design the experiment, there were three main challenges. This experimental procedure must 1) display text and images explaining the experiment, and as stimuli, 2) determine the proper stimuli to display and record based on the user's previous responses in the rating task, and 3) record important data such as timing and user responses corresponding to the proper stimuli.

While Pygame provides many useful modules for displaying text and images, determining the location was an important problem. Another problem in displaying text is proper text-wrapping, as Pygame does not provide any modules that solve this problem. The images would have to be centered properly, and the text would have to be spaced so that it is legible.

In the task, the participants needed to be able to input their decisions on the keyboard, and then their decisions in the rating task would need to be grouped in such a way as to provide groups for the following social dilemma task. Because subtle differences in the time it takes an individual to make decisions can provide an insight into the decision-making process, the procedure would need to get the user's response time accurately and record it in a spreadsheet for future data analysis. All the data from the experiment would have to be organized in a way that is conducive for future data analysis.

## Solution architecture:

1) Because each procedure had different requirements in terms of what types of stimuli (text, images) to display, how to determine which stimuli to display, how long to display them, and what to record, each type of procedure was separated into different functions. Within these functions, algorithms were implemented to determine the location of each item to be displayed, and lists were used to determine which stimuli would be used.

2) However, in order to determine the proper stimuli to display, a class representing the participant, and his/her decisions was created. This would record previous responses, and send these to be recorded, create lists based on these decisions to inform future tasks if needed. The participant class was included with the main code file, as it was an important component to all tasks.

3) Because recording data required specific procedures, I decided it would be best to separate the related functions and classes into different, easy to read, code folders. This meant that the spreadsheet code, which used a class to open and print to the file each time it was called, was separate from the timing code, which was used to time and record response time of tasks.

## Architecture Implementation:

Initially, I planned to use the libraries in PsychoPy to create the experimental procedure. However, after working with this suite for some time, I discovered that I was doing basically the same thing I would have been doing with Pygame. Since PsychoPy requires quite a bit of space to load and install, I decided it would be better to use Pygame.

1) In each method, representing parts of each procedure, the location where the stimuli would be displayed was determined, by receiving the screen size as a parameter, getting how much space each stimuli would take up, then using that information to place the stimuli centered in the x-axis, and below any other previous stimuli that needed to be displayed in the frame. Then the blit() function was used to display the frame.

2) Before beginning the experimental procedure,  a few important lists and a dictionary were created that would be used throughout the program. First, a CSV spreadsheet containing the title and filenames of all the food stimuli was loaded onto a dictionary. This allowed us to go through the dictionary, display the titles and images, according to the task. For example, in the rating task, the rating_task() algorithm loops through every dictionary item in file_foods, allowing the participant to see the food item and its description, and rate them according to their preference. Other lists (opponent_a, opponent_b, and opponent_c), that represent the AI partner that the participant can play against, which are modified by the make_opponent() method. Later, one of these lists is used (predetermined by modifying the code before the experiment begins) to determine whether the AI partner decides to cooperate (choose a less expensive item) or defect (choose a more expensive item).

Within the participant class, the data file object was held, as well as information about the participants responses. Using Tkl (in review.py), I made it possible for the experimenter to enter the participant number, and to record the date the experiment takes place automatically. These both were important to data recording. There are lists of rated foods, which, after the rating task is completed are full, and then used to derive different category lists based on the sorted responses of the user. This made it possible for the foods to be classified by least preferred foods, moderately preferred foods, and highly preferred foods.

These groups were important to decision-making task, allowing the participant to choose between items that were more preferred, but more expensive, or to choose items that were less expensive and thus better for the group. In the decision-making task, it was useful to include the last response of both the participant and the computer, so I included both of these in the participant class for accessibility among the functions (and to reduce the number of parameters needed to be passed between the functions). This way, following the users decision we could display the result of the trial after getting the computers response.

3) Data recording: As mentioned before, the “makedata.py” code was used to write to the csv file. In the participant object, a DataSheet object is made, which can be called to write to the proper file each time a trial is ended. The “timing.py” code was important to time tasks. Since Windows is the only OS that will run the data that is recorded, I opted to use the time.clock() function. It is important that Windows is used for this to properly record the time passed, as different operating systems implement this function differently. In Windows, it records the absolute time passed, and can be accurate to the millisecond. In other OS, it uses the processor time, which can be inaccurate if many processes are using the processor.

Each time the experimental procedure had data that needed to be recorded, we passed a dictionary with that trial’s data to the participant object, which processes and passes it to the DataSheet object, which then assures that each data is recorded in the proper column of the csv file. Each time it writes a new line, it opens the file and then closes it after the line is recorded, where the program can continue until more data needs to be recorded. This guarantees that, if there is a computer crash, the previously recorded data will not be lost. This is a huge advantage over using other software packages to create an experimental procedure, because the most popular ones lose all data if the computer crashes or restarts during the experiment.

## Conclusion:

Designing this experimental procedure was an opportunity for me to learn how to build a larger project. Previous assignments in all of my computer science courses have required me to create relatively small projects, where organization is not so essential. But in creating this procedure, I was required to keep track of many more different pieces, and so organization was essential. I spent quite a bit of time going back and improving the organization of working code to make it more readable, and so it could work better with other functions and classes. I was also to learn how to use Pygame more efficiently, as well as make simple GUI designs with the built-in TKL module.