Super Mario Matlab

During the two lectures of the introduction to Psychophysics Toolbox (PTB) we are going to develop a simple game which will show how to structure the code for creating a task using the most common functions available in PTB.

A picture containing indoor, decorated

Description automatically generatedA close-up of a toy

Description automatically generated with medium confidenceA picture containing light

Description automatically generatedA close up of a toy

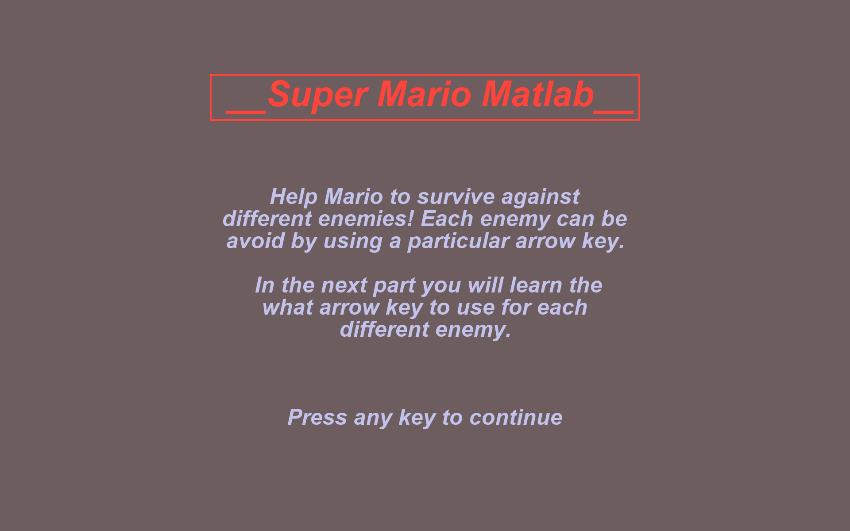
Description automatically generated with medium confidence The purpose of the game will be to avoid several different enemies, as in the Mario franchise games. The enemies we are going to encounter are: “Lakitu”, “Goomba”, “Bill” and “Super Hammer”.

The purpose of the game is to present each of these enemies, one at a time, to the user and the let them react by pressing a key on the keyboard. Each of the enemy can be “avoided” by pressing a specific key. This key can be pressed any time after the enemy is being presented, but the user will have only a specific amount of time to press the correct key. The keys to avoid the enemies are: the right arrow key for “Lakitu”, the up-arrow key for “Goomba”, the down arrow key for “Bill” and the left arrow key for “Super Hammer”.

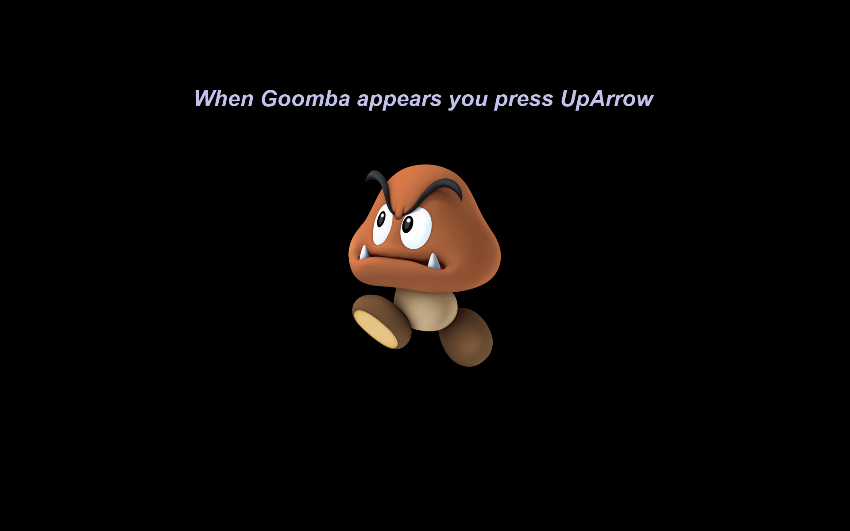
The game is divided in two parts, a tutorial and a testing phase. During the tutorial, the participant will be shown a welcome screen for the game and will learn the associations between different enemies and arrow keys. During the testing phase, the participant will be shown a random selection of enemies, one at a time, and can press a key to avoid each enemy. A proposed solution for the tutorial phase will be available after the end of the first lecture, while a proposed solution for the testing phase will be available after the end of the second lecture.

# Tutorial Phase

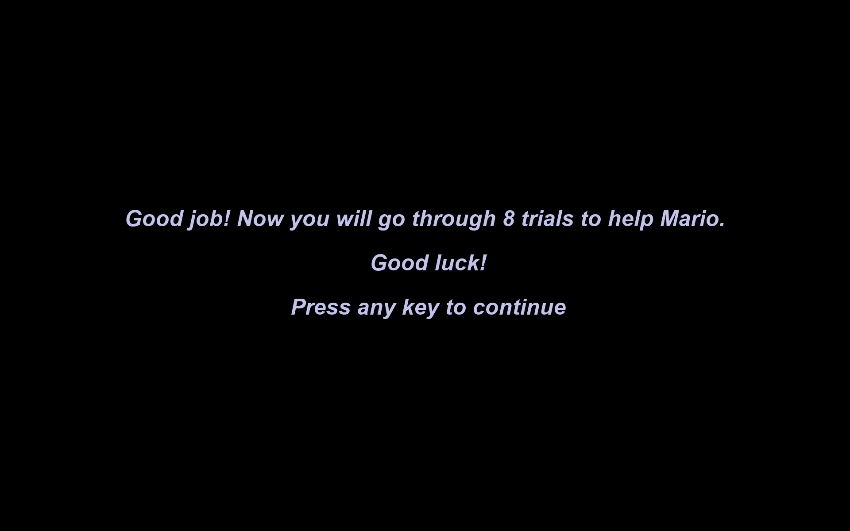
In this section a description of the tutorial with some example screenshots will be provided. The tutorial will start after initializing the PTB window.

1. First, the participant will be presented with a welcome screen showing a brief description of the game. This screen will stay on the PTB window until any key is pressed. An example of the screen is provided below.

[BONUS] Notice how the title is enclosed in a rectangular frame. You can try using the sub-function Screen(‘FrameRect’) to draw it. Type in *Screen FrameRect?* in the command window to get help information and examples for this sub-function.

1. After the welcome screen, the participant will be presented with one of the four enemy images. The image will be placed at the center of the screen, and text will be displayed on the upper part of the screen stating which key is expected from the participant in this case, as shown in the image below. This screen will remain until the participant makes a key press. The process will repeat until all the enemy images will be shown once. Enemy images can be found in the “Assets” folder contained in the zip folder that accompanies the first lecture.

[BONUS] A user should only be presented with the next image when the correct key has been pressed. Can you think of a way to check for the correct keycode?

1. After the last of four enemy images has been displayed, a final tutorial screen will remind the participant that the testing phase is about to start. The screen will remain in place until the participant presses any key. See the image below as a proposed example.
2. After step 3, you can start the Testing phase

# Testing phase

During testing phase, the participants will have a chance to react to each enemy images by providing the expected key upon enemy presentation. Each trial should proceed as follow

1. A fixation cross at the center of the screen should indicate the start of the trial
2. The fixation cross disappears after a fixed amount of time and a random image from the set should be displayed on the center of the screen. This marks the beginning of keyboard recording.
3. The image will stay on place for a fixed duration giving the participant a chance to react to it by pressing a key
4. The image should disappear, and auditory feedback is played depending on whether the participant pressed the correct key or not
5. Repeat from 1 until all the enemy images have been displayed a desired number of times

Here are also some additional requirements that should be implemented:

* Each trial has always the same amount of duration regardless of the fact the participant has provided an input or not
* The game should only capture the first key press detected after the start of the trial. All subsequent key presses should be discarded.
* At the end of the testing phase the screen should be safely closed. The data collected should be saved permanently on your project under ‘Result’ folder.
* Before starting each trial, the participant should have released all the keys from the keyboard.
* In case of error, we should be able to save the partial data obtained so far

Auditory feedback audios can be found in the ‘Assets’ folder. ‘pos\_feedback’ should be used for a positive feedback while ‘neg\_feedback’ should be used for a negative feedback.

When structuring your code, it would be nice to be able to quickly change some of the values, without hard coding them. Those values can be

1. the duration of the fixation cross
2. the number of times an enemy will be presented
3. the duration of each image to be displayed
4. the scale of each image
5. the orientation of each image

For each trial we should collect an id for the image being presented and the key associated to it, the key the user has pressed, the reaction time when the key has pressed and a logical number indicating whether the response was correct or not. Below you can find a proposed structure for the collected data. In this example each image was presented twice

A screenshot of a computer

Description automatically generated with medium confidence