## Game Gestures

The game of rock-papers-scissors requires at least three distinct gestures to represent all of the three items. Moreover, to signal the beginning of a round, we added a fourth gesture (called sync), giving us a total of four distinct gestures.

In order to clearly differentiate between the gestures, we define each one as the sum of two micro-gestures. For example, the sync gesture would result from a positive displacement of the board in the z axis, followed by a negative displacement in the same axis (effectively moving the board up and down, as one does when playing the normal game). The definition of each gesture as a set of two micro-gestures was originally set as follows:

1. Sync: positive z displacement followed by negative z displacement (or vice versa)
2. Paper: positive y displacement followed by negative y displacement (or vice versa)
3. Scissors: positive x displacement followed by negative x displacement (or vice versa)
4. Rock: positive roll followed by a negative roll (or vice versa)

In code, we define 2 enums to represent our micro-gestures and symbols. Then we use a struct to couple micro-gestures with their resulting symbols. This gives us the following code in :

typedef enum {

pos\_x = 1,

neg\_x,

pos\_y,

neg\_y,

pos\_z,

neg\_z,

pos\_roll,

neg\_roll

} mgest\_t; /\* micro gesture (eg. forward x, backward x, etc.) \*/

typedef enum {

paper = 1,

rock = 2,

scissors = 4,

sync = 8,

no\_move = 0

} symbol\_t;

typedef struct {

mgest\_t mgest[2];

symbol\_t symbol;

} gesture; /\* macro gestures: symbol = mgest1 + mgest2 \*/

We then define the 8 valid moves of our game in an array of gesture structs:

#define VALID\_MOVES 8

/\* valid moves for the game \*/

gesture valid\_moves[VALID\_MOVES] = {

{{pos\_x, neg\_x}, scissors}, {{pos\_y, neg\_y}, paper}, {{pos\_z, neg\_z}, sync}, {{pos\_roll, pos\_roll}, rock}, {{neg\_x, pos\_x}, scissors}, {{neg\_y, pos\_y}, paper}, {{neg\_z, pos\_z}, sync}, {{neg\_roll, neg\_roll}, rock}

};

As can be seen from the array, each symbol/item is associated with an axis, and its two possible moves are defined as moving forward-backward or backward-forward on that axis. The rock is a special case that uses an angle (roll) instead of an axis, and will be discussed in a later section.

## Gesture Identification

A gesture is recorded only if two of its defining micro-gestures are executed successively. For example, a sync would only be identified if the board is moved up then down, or down then up, as per its definition in valid\_moves. Moving the board up, then right, and finally down would not return a sync gesture as the defining micro-gestures were not executed consecutively.

This logic is implemented in the form of an intelligent stack of two slots of micro-gestures (mgest\_t). The first time the stack receives a micro-gesture, it stores it, and waits for another one. If the next recorded micro-gesture is the same as the one that was most recently stored, we simply ignore it. This way, we store a micro-gesture only once, even if it is recorded many times in the process of executing it. When the stack fills up with two micro-gestures, a function is called to process them by comparing them to one of the valid\_moves gestures. If a match is found, the corresponding symbol\_t is returned as the effective gesture executed. If the two micro-gestures don’t have a matching gesture, only the last is left on the stack and the oldest is cleared to make way for the next incoming micro-gesture.

The logic illustrated above is implemented in the file which stands for micro-gesture stack.

## Gesture Update

When new data is available from the sensors, this is signalled to two functions that eventually lead to a move being deduced. The first one is which filters the raw accelerometer data, and computes an accurate measurement of roll. This processed data is then fed to a function that is responsible of identifying the micro-gesture from the data, passing it to the intelligent stack, and receiving resulting gestures from it (if any). This function is called and is found in the file .

The manner in which determines the micro-gesture is by computing the difference between the current accelerometer data and that of the previous call, and then comparing this delta to a threshold value experimentally set for each axis. If the delta on one of the axes exceeds its threshold, then this is equated to a jerk executed on that axis, and is thus translated to an appropriate micro-gesture (pushed onto the intelligent stack).

The roll is treated differently. If its value exceeds 30° AND the z-axis accelerometer reading is negative (meaning the board is flipped), then this motion is processed as a rock. Originally, a rock was defined as a positive roll followed by a negative roll. But it was quickly realized that this motion was sometimes being confused with the other motions (as jerks were detected in some axes when the board is rotated). Thus, we added the negative z-axis requirement, and ensured that roll was checked for first in our processing. This resolved our problem, and made the gesture system more robust.