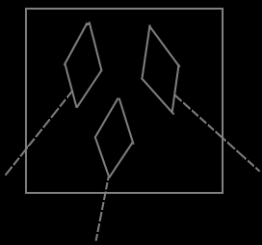
|  |
| --- |
| Circle Language Spec: Execution Control |

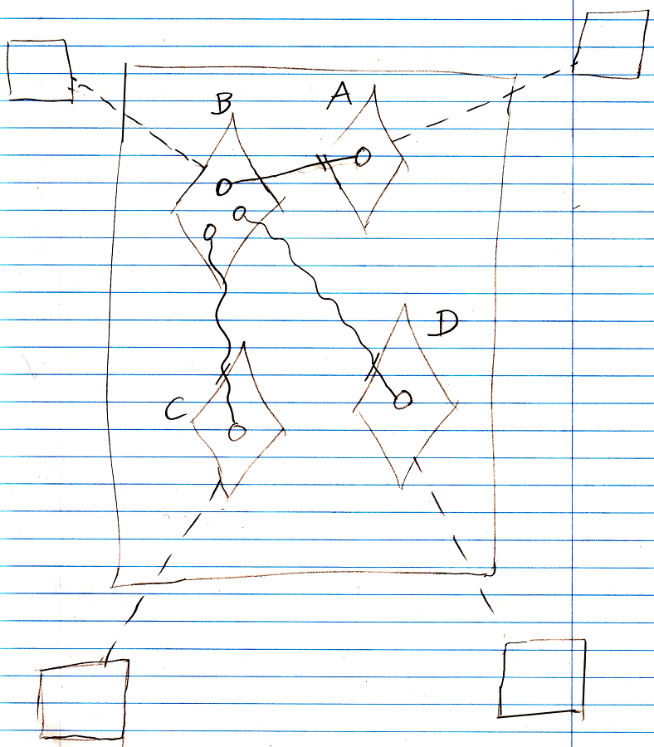
## Normal Execution Order in a Diagram

The concept of normal execution order is already covered by the article *Normal Execution Order*. This article only explains its expression in a diagram.

If you do not indicate an order in which to execute the commands, then they may execute in an arbitrary order. Sometimes this is acceptable from a functional point of view. The diamonds representing the calls are simply placed inside the command definition as follows:



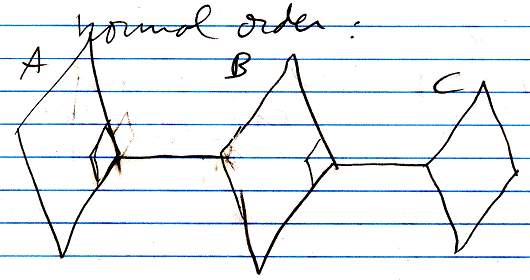
The order of execution of commands can also be determined by the input / output dependency of their parameters.



The output object of A goes into B. The two output values of B go into C and D. Therefore A executes first, then B, then C and D are executed in an arbitrary order.

This notation of parameter passings is described in the *Parameters* articles (may not be written yet)

When the order of the commands should not be arbitrary or determined by input / output direction, then the order is explicitly drawn out in the diagram. This is what we are usually talking about when we refer to *normal execution order*. One command actually *calls* the next, but the call never returns to the caller. This is displayed as follows:

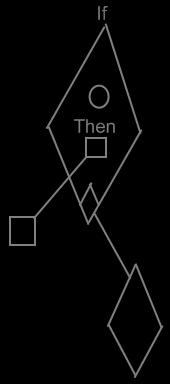


The first command to run is a command, that is nobody’s next statement, not even in the input, output dependency. In the diagram above, that would be A. The commands are executed in the order A, B, C. The fact, that B is followed up by A is displayed as an active reference from A to B. But the active reference never returns to A again. This is displayed as the active reference from A to B’s being stuck to the end of A, so that if you follow the line to B back to A, it does not enter back into A again: it does not return back into A.

The diagram notation also looks like a call to the *end* of a command.

The diamond stuck to the border of a call is actually called the Command End, and is an *actual* reference to the next command, but then called in a special way.

Normal execution order is also indicated for calls to execution control statements, like If calls and For calls.



Normal execution order only puts command *calls* next to eachother, not command *definitions*. The normal-execution-order ‘call’ (stuck to the side of the command) is always the *last* call of a sub-command. It moves execution on to the next step.