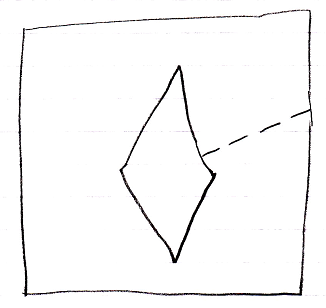
|  |
| --- |
| Circle Language Spec: Commands |

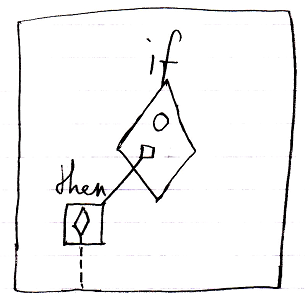
## Recursion in a Diagram

The concept of recursion was already explained in the article *Recursion*. The current article further explains the concept with the help of diagrams.

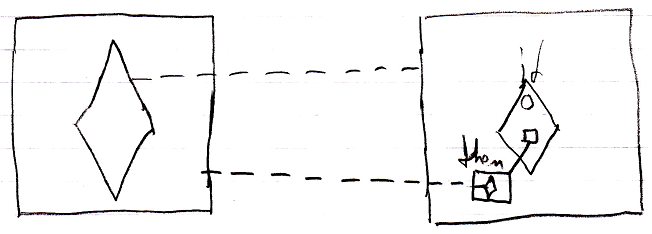
Recursion in commands happens when a command contains a call to the same command definition again.



The recursion is usually conditional, for instance by putting a break on it using an If statement:

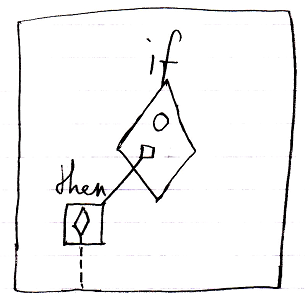


Recursion can also occur when one command calls another command and the other command potentially calls the first command again.

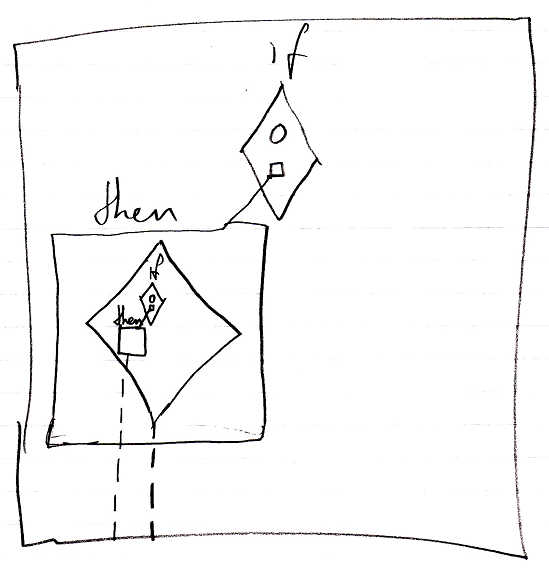


The problem with endless recursion is solved, by the fact, that private contents of an executable object are only created just before it is actually going to execute.

When a command definition contains a call to the same command definition, then the call does not contain any private content yet, because it is not executing.

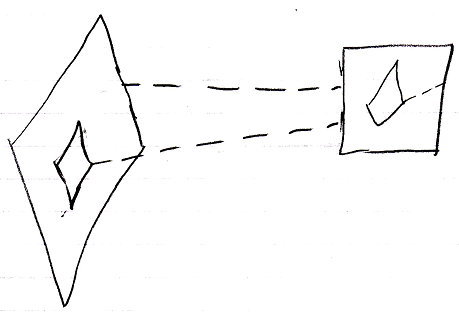


The call in the then clause does not execute, so shows no private contents. Would the then clause be executing, then the diagram would look like this:



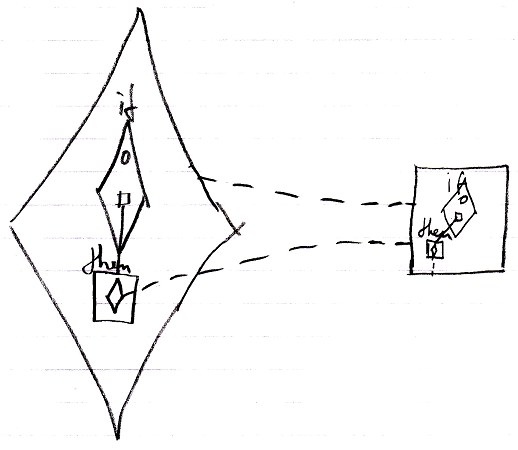
But that never happens, because an inactive command object never executes.

When a command call contains a call to the same command definition, then the call also does not contain any private content yet, before it executes.

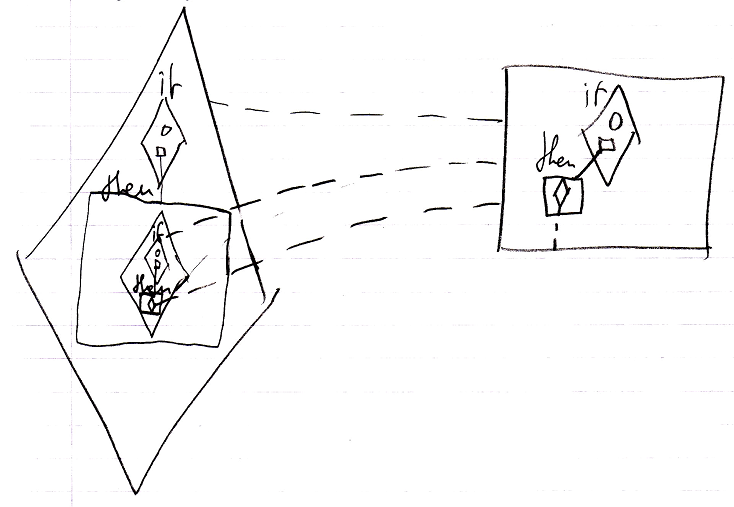


The call on the left contains a call to the same definition, but it shows no private content, because it is not running yet.

The picture below displays a call to a *conditionally* recursive command:

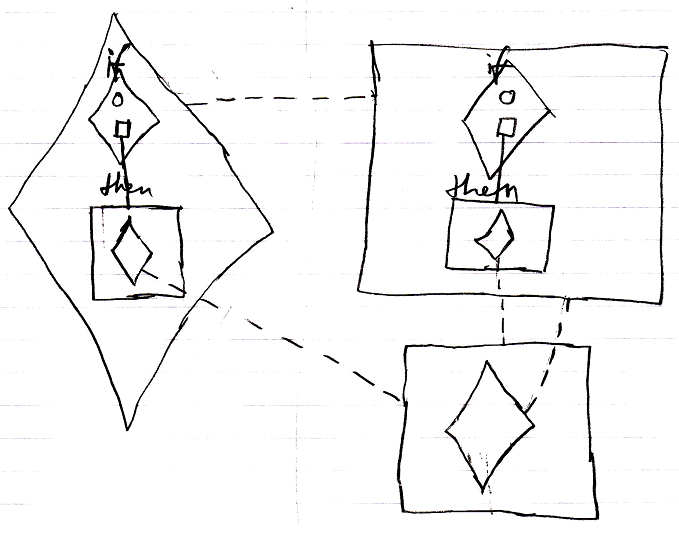


Only when the then clause in the call on the left is actually executed, the call’s private contents are created *and* visible in the diagram.

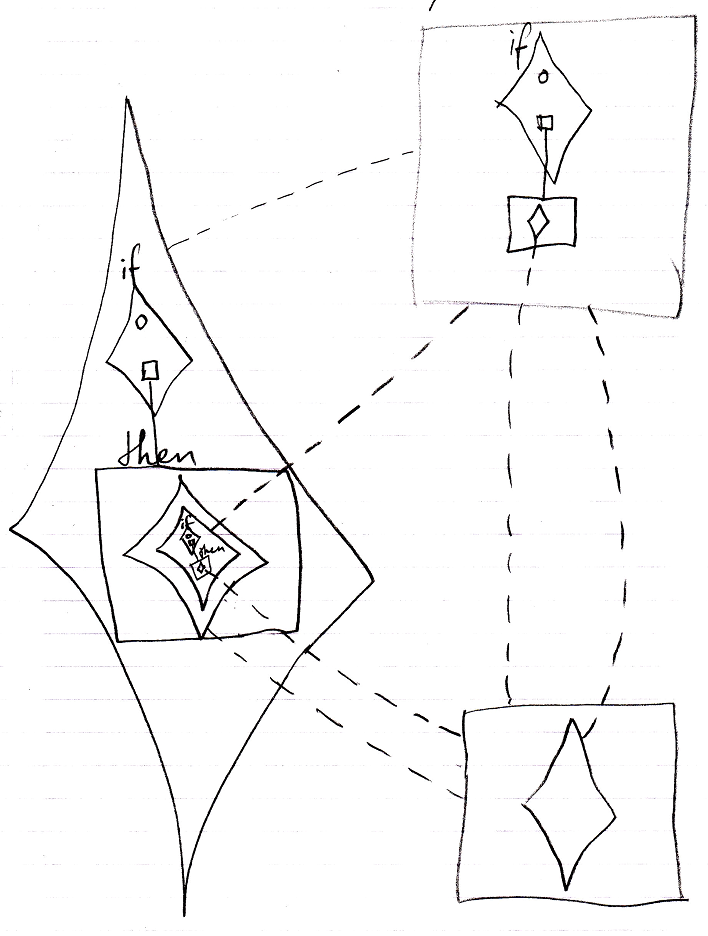


So only when recursive calls are actually busy executing, you this recursive repetition is visible in the system.

Here is a diagram of a call to a command that delegates to another command, that delegates to the first command again. The two command definitions are on the right, the calls are on the left:



The call on the right might be executing, but its call inside the then clause is not executing yet. Only when the then clause is executed, the private contents of the diamond inside the then clause are created *and* visible in the diagram.



Yes, there are a lot of class lines, but each separate call has a separate class line to the right definition. If there is a recursive command running, then each execution of the same command has a separate execution symbol, because there will be an execution contained in an execution, contained in an execution, etcetera.