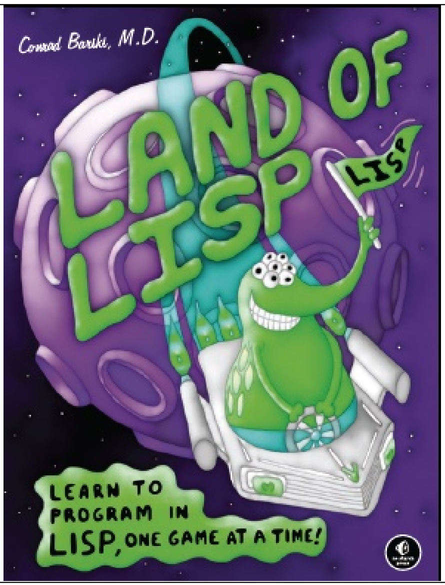


Chapter
17

Domain-Specific Languages
Ch. 17

- **Creating Custom Game Commands for our Wizard's Adventure Game**

Interesting Actions for our Game

- Creating interesting activities in the game poses a unique challenge.
- There are *clearly many similarities between different game actions.*
- I.e. *most* of them *will require us to have an object in our possession.*
- But, they all need to *have unique and idiosyncratic properties* (enabled through command-specific Lisp code) or the game becomes boring.

Creating New Game Commands by Hand

Creating a Command for Welding

- There is a welder in the attic
- Create a global variable to keep track of what has been welded

```
; The chain is initially not welded
(defparameter *chain-welded* nil)
          ;or
(setf *chain-welded* nil)
```

Welding – What Do We Mean?

Lets allow the player to

weld the chain (part1) **to**
the bucket (part 2) (two items)
if they bring those items
(chain and bucket) to the
attic (location) where the
welder (tool/action) is located.

Pre-Conditions for Welding, Precisely

These conditions must be met :

1. You must be in the attic.
2. You must be carrying the chain and bucket.
3. The chain and bucket can’t already be welded together.
4. You need the chain and bucket as the subject and object of the welding command.

Welding Function Conditions in Lisp

```
(defun weld (subject object)
      ; Check preconditions for welding
 (if (and (eq *location* 'attic)   ; #1
          (eq subject 'chain)      ; #2a
          (eq object 'bucket)      ; #2b
          (have 'chain)            ; #3a
          (have 'bucket)           ; #3b
          (not *chain-welded*))    ; #4
      (progn ; next slide
```

Welding Function Actions

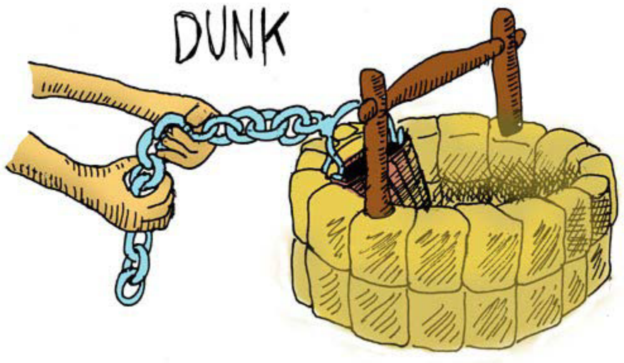
```
(progn
          ; then clause
  (setf *chain-welded* t)
  '(the chain is now securely
     welded to the bucket.))

          ; else clause
'(you cannot weld like that.)))
```

Adding the New Command to the Game-REPL

```
> (game-repl)    ; lets try weld
weld chain bucket
I do not know that command.
Quit
    ; forgot to add weld to commands!
> (pushnew 'weld *allowed-commands*)
(WELD LOOK WALK PICKUP INVENTORY)
> (game-repl)    ; try again
weld chain bucket
You cannot weld like that.
    ; what is wrong this time?
```

New Action: Dunk the Bucket in the Well



Pre-Conditions for Dunking

Dunking is possible only if:

1. You must be in the garden.
2. You must have the bucket with you.
3. The chain and bucket must be welded together.
4. You must have the bucket and the well as the subject and object of the dunking command.

Adding Dunk to the Game-REPL

```
(defparameter *bucket-filled* nil)

(defun dunk (subject object)
 (if
    (and (eq *location* 'garden)
         (eq subject 'bucket)
         (eq object 'well)
         (have 'bucket)
          *chain-welded*)
    (progn                           ; then clause
      (setf *bucket-filled* 't)
     '(the bucket is now full of water))
                                     ; else clause
     '(you cannot dunk like that.)))

(pushnew 'dunk *allowed-commands*)
```

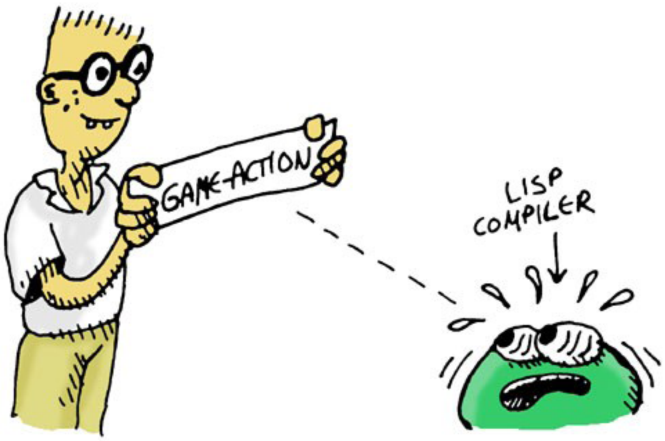
Observations on Weld and Dunk

- Can we add the action name to the allowed actions automatically?
- Can we omit some repetitive parts?
- Yes! – use macros!!!

Writing a Game Action Macro!

- The weld and dunk commands are very similar.
- However, each game command needs to contain a certain amount of logic in it, to customize the behavior of the command.
- So, lets write a game-action macro that addresses these issues.
- It will make it much easier to create new game commands.

Game Action Macro



```
(defmacro game-action (command subj
        obj place &body body)
  `(progn
    (defun ,command (subject object)
      (if (and (eq *location* ',place)
                (eq subject ',subj)
                (eq object ',obj)
                (have ',subj))
                 ,@body
         '(i cant ,command like that.)))
     (pushnew ',command
              *allowed-commands*)
    )
 )
```

Welding Revised

```
(defparameter *chain-welded* nil)

(game-action weld chain bucket attic
   (if (and (have 'bucket)
            (not *chain-welded*))
     (progn (setf *chain-welded* 't)
        '(the chain is now securely
             welded to the bucket.))
      '(you do not have a bucket.)))
```

Dunking Revised

```
(defparameter *bucket-filled* nil)

(game-action dunk bucket well garden
  (if *chain-welded*)
    (progn
       (setf *bucket-filled*'t)
       '(the bucket is now full of
                                 water))
    '(the water level is too low to
                               reach.)))
```

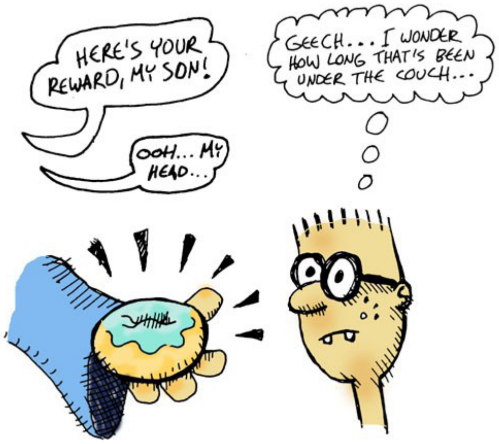
Splash Action

```
(game-action splash bucket wizard living-room
   (cond
      ((not *bucket-filled*)
       '(the bucket has nothing in it.))
      ((have 'frog)
       '(the wizard awakens and sees that
          you stole his frog.  he is
          so upset he banishes you to the
          netherworlds - you lose! the end.))
       (t '(the wizard awakens from his
            slumber and greets you warmly.
            he hands you the magic low-carb
       donut - you win! the end.))))
```

Now – The Complete Game

```
> (game-repl)
look
You are in the living-room. There is a wizard
snoring loudly on the couch.
There is a door going west from here. There is a
ladder going upstairs from
here. You see a whiskey on the floor. You see a
bucket on the floor.
pickup bucket
You are now carrying the bucket
pickup whiskey
You are now carrying the whiskey
inventory
Items- whiskey bucket
......................
```

Can You Win the Reward?


HERE'S YOUR REWARD, MY SON!
GEECH... I WONDER HOW LONG THAT'S BEEN UNDER THE COUCH...
OOH... MY HEAD...

Summary

- Now you are ready to create your own adventure game!