The Cuis-Smalltalk book

Work-In-Progress

Hilaire Fernandes and Juan Vuletich

This book is for Cuis-Smalltalk (5.0#4253), a free and modern implementation of the Smalltalk language and environment.

The syntax chapter is from the *Squeak by Example* book.

Compilation : 18 August 2020
Documentation source: `https: / / github . com / Cuis-Smalltalk / TheCuisBook`

Thanks to Ken Dickey for its careful review.

Short Contents

Table of Contents

Preface

What Cuis-Smalltalk is: Cuis-Smalltalk – or in short Cuis – is a portable environment for doing, building, and sharing software.

Cuis is a journey and a process and comes with a philosophy of reducing complexity while providing a complete, live software development experience.

This book is an introduction and invitation to exploring Cuis.

We hope you will join us in this journey to make Cuis ever better.

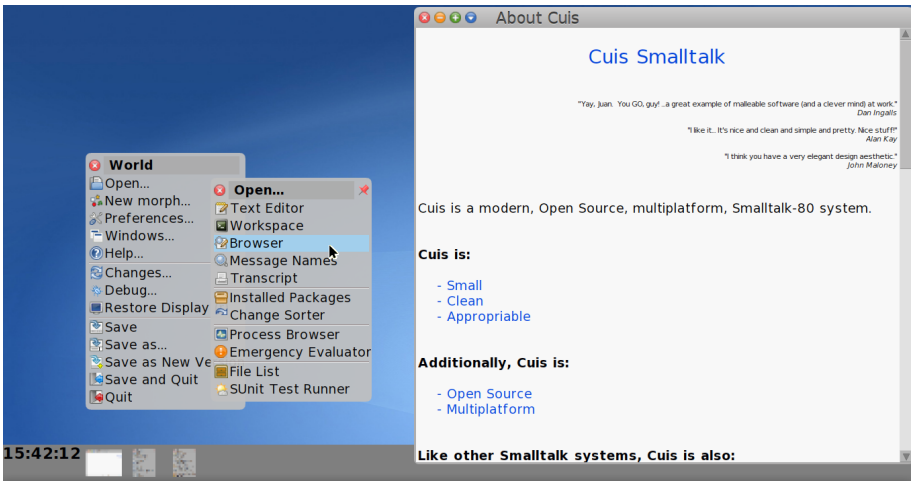

Figure 1: Cuis

To make your journey with this book more enjoyable, the *Spacewar!*[1] project is its recurring theme. It is distilled along the book in code examples, exercises and dedicated chapters. At the end of the book, you will have written a replica of this historic video game.

[1] `https://en.wikipedia.org/wiki/Spacewar!`

1 Smalltalk Philisophy

2 The Message Way Of Life

3 Classes

4 Control Flow Messaging

5 Visual With Morph

6 Model And View

7 Events

8 Process

9 Code Management

Appendix A Liste des exercices

Appendix B Solutions of the exercises

Installing and configuring Cuis-Smalltalk

⟨undefined⟩ [exePlacement], page ⟨undefined⟩

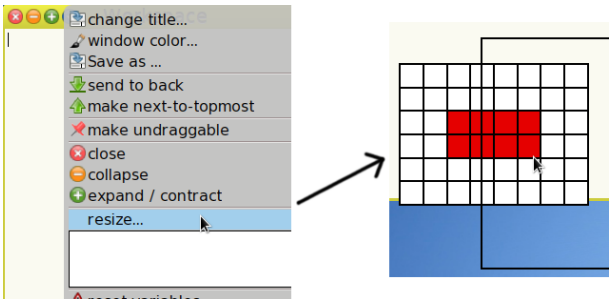

Figure B.1: Placement

Writing your first scripts

Fun with numbers

⟨undefined⟩ [inverseSum], page ⟨undefined⟩

```
1 + (1/2) + (1/3) + (1/4)
⇒ 25/12
```

⟨undefined⟩ [squaredSum], page ⟨undefined⟩

```
1 + (1/2) squared + (1/3) squared + (1/4) squared
⇒  205 / 144
```

⟨undefined⟩ [exeFloatPrecision], page ⟨undefined⟩

```
5.2 + 0.9  - 6.1
⇒ 8.881784197001252e-16

5.2 + 0.7 + 0.11
⇒ 6.010000000000001

1.2 * 3  - 3.6
⇒ -4.440892098500626e-16
```

⟨undefined⟩ [exeZeroDivide], page ⟨undefined⟩

The system returns the error ZeroDivide, division by zero.

⟨undefined⟩ [exeFractionPrecision], page ⟨undefined⟩

```
(52/10) + (9/10)  - (61/10)
⇒ 0

(52/10) + (7/10)  + (11/100)
⇒ 601/100 soit 6.01

(12/10) * 3  - (36/10)
⇒ 0
```

Fun with text

⟨undefined⟩ [capWordNumber], page ⟨undefined⟩

Several messages can be sent one after the other:

```
2020 printStringWords capitalized
```

⟨undefined⟩ [helloBelle], page ⟨undefined⟩

```
'Hello'
   at: 1 put: $B;
   at: 2 put: $e;
   at: 3 put: $l;
   at: 4 put: $l;
   at: 5 put: $e;
   yourself
```

Fun with collection

⟨undefined⟩ [exeNegativeIntegers], page ⟨undefined⟩

```
(-80 to: 50) asArray
```

⟨undefined⟩ [holeSet], page ⟨undefined⟩

```
(1 to: 100) difference: (25 to: 75)
⇒ #(1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
22 23 24 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91
92 93 94 95 96 97 98 99 100)
```

⟨undefined⟩ [oddNumbers], page ⟨undefined⟩

```
(-20 to: 45) select: [:z | z odd]
```

⟨undefined⟩ [qtyPrime200], page ⟨undefined⟩

```
((101 to: 200) select: [:n | n isPrime]) size
⇒ 21
```

⟨undefined⟩ [multiples7], page ⟨undefined⟩

```
(1 to: 100) select:[:n | n isDivisibleBy: 7]
⇒ #(7 14 21 28 35 42 49 56 63 70 77 84 91 98)
```

⟨undefined⟩ [oddNonPrime], page ⟨undefined⟩

This solution, based on set operations and multiple use of the #select: message, is mostly compatible with the knowledge acquired at this point of the book.

```
| primeNumbers nonPrimeNumbers |
primeNumbers := (1 to: 100) select: [:n | n isPrime].
nonPrimeNumbers := (1 to: 100) difference: primeNumbers.
nonPrimeNumbers select: [:n | n odd]
⇒ #(1 9 15 21 25 27 33 35 39 45 49 51 55 57 63 65 69 75
77 81 85 87 91 93 95 99)
```

A shorter solution with logical operations we have not discussed so far:

```
(1 to: 100) select:[:n | n isPrime not and: [n odd]]
```

⟨undefined⟩ [decodeCipher], page ⟨undefined⟩

```
'Zpvs!bsf!cptt' collect: [:c |
   (c asciiValue - 1) asCharacter]
⇒ 'Your are a boss'
```

⟨undefined⟩ [alphabetCipher], page ⟨undefined⟩

```
($A to: $Z) collect: [:c |
   (c asciiValue - 65 + 3 \\ 26 + 65) asCharacter]
```

⟨undefined⟩ [encodeCaesar], page ⟨undefined⟩

In the solution of ⟨undefined⟩ [alphabetCipher], page ⟨undefined⟩, we just need to replace the characters intervale with a string:

```
'SMALLTALKEXPRESSION' collect: [:c |
   (c asciiValue - 65 + 3 \\ 26 + 65) asCharacter]
⇒ 'VPDOOWDONHASUHVVLRQ'
```

⟨undefined⟩ [decodeCaesar], page ⟨undefined⟩

```
  'DOHDMDFWDHVW' collect: [:c |
    (c asciiValue - 65 - 3 \\ 26 + 65) asCharacter]
⇒ 'ALEAJACTAEST'
```

Introduction to the system class

⟨undefined⟩ [stringArith], page ⟨undefined⟩

From a System Browser, do from the left panel to the right ...`Kernel-Text` → String → arithmetic... the count of methods in the last right panel is 6: *, +, -, /, // and \\.

⟨undefined⟩ [floatInfo], page ⟨undefined⟩

When the Float is selected, the wide text pane prints: "class definition for Float ° 92 instance methods ° 34 class methods ° 1280 total lines of code"

⟨undefined⟩ [cosTable], page ⟨undefined⟩

```
0 to: Float twoPi by: 1/10 do: [:i |
   Transcript show: i cos; cr]
```

⟨undefined⟩ [multiplyBy1024], page ⟨undefined⟩

1024 is not a random number. It is 2^{10} then written in base 2 : 10000000000, it is also 1 << 10:

```
2^10 ⇒ 1024
1024 printStringBase: 2 ⇒ '10000000000'
1 << 10 ⇒ 1024
```

Therefore, to multiply an integer by 1024, we shift left of 10 its digits:

```
360 << 10 ⇒ 368640
360 * 1024 ⇒ 368640
```

⟨undefined⟩ [selectApples], page ⟨undefined⟩

There are different options, with slightly different results:

```
'There are 12 apples' select: [:i |i isLetter].
⇒ 'Thereareapples'
```

Not really satisfying. So another option:

```
'There are 12 apples' select: [:i |i isDigit not].
⇒ 'There are  apples'
```

Or even a shorter option with the #reject: message:

```
'There are 12 apples' reject: [:i |i isDigit].
⇒ 'There are  apples'
```

⟨undefined⟩ [formatString], page ⟨undefined⟩

In String, search for the method category `format`, there you find the format: method:

```
'Joe bought {1} apples and {2} oranges' format: #(5 4)
⇒  'Joe bought 5 apples and 4 oranges'
```

⟨undefined⟩ [cutString], page ⟨undefined⟩

Open the protocol browser on the class String, search for the method allButFirst: implemented in SequenceableCollection. Read its comment in its source code.

```
'Hello My Friend' allButFirst: 6
⇒ 'My Friend'
```

⟨undefined⟩ [collFirst], page ⟨undefined⟩

The appropriate message is #first:, defined in the parent class SequenceableCollection. You need to use the protocol or hierarchy browser on Array to discover it:

```
array1 first: 2
⇒ #(2 'Apple')
```

⟨undefined⟩ [fillArray], page ⟨undefined⟩

You could simply do a thumb:

```
array4 at: 1 put: 'kiwi'.
array4 at: 2 put: 'kiwi'.
array4 at: 3 put: 'kiwi'.
array4 at: 4 put: 'kiwi'.
```

Or even a bit less thumb:

```
1 to: 4 do: [:index |
   array4 at: index put: 'kiwi']
```

But if you search for carefully the Array protocol, you can just do:

```
array4 atAllPut: 'kiwi'.
```

⟨undefined⟩ [addAfter], page ⟨undefined⟩

In the OrderedCollection protocol search for the method add:after:.

```
coll1 := {2 . 'Apple' . 2@1 . 1/3 } asOrderedCollection .
coll1 add: 'Orange' after: 'Apple'; yourself.
⇒  an OrderedCollection(2 'Apple' 'Orange' 2@1 1/3)
```

⟨undefined⟩ [setLetters], page ⟨undefined⟩

```
Set  new
   addAll: 'buenos días';
   addAll: 'bonjour';
   yourself.
⇒  a Set($e $j $o $a $u $b $  $í $r $d $n $s)
```

⟨undefined⟩ [nameColor], page ⟨undefined⟩

```
colors keysDo: [:key |
   colors at: key put: key asString capitalized].
```

```
colors
⇒ a Dictionary(#blue->'Blue' #green->'Green' #red->'Red'
#yellow->'Yellow' )
```

⟨undefined⟩ [blockDivisor], page ⟨undefined⟩

```
| divisors |
divisors := [:x | (1 to: x) select: [:d | x \\ d = 0] ].
divisors value: 60.
"⇒ #(1 2 3 4 5 6 10 12 15 20 30 60)"
divisors value: 45
"⇒ #(1 3 5 9 15 45)"
```

⟨undefined⟩ [implementingAndOr], page ⟨undefined⟩

Check the implementations in Boolean, True and False.

Appendix C List of the examples

Appendix D List of the figures

Appendix E Conceptual index

(Index is nonexistent)