SMALLTALK PROJECT

Andres Barragan
Computer Science
University of Arizona
Tucson, Az 85721
andresbarragan@email.arizona.edu

Jackson Troyer
Computer Science
University of Arizona
Tucson, Az 85721
jacksontroyer@email.arizona.edu

April 21, 2020

1 Introduction

2 History

Smalltalk, like many different languages, has developed and branched to the point where a number of different variants exist in one form or another. And it itself began as a variant of another language, namely Simula, the language supposedly being made on a dare as a version of that language based on message passing. It would go on to become one of the first ever object oriented programming languages in history, and many others to come would draw inspiration from it and it's design principles, including Ruby, Java, Python, and Objective C. As far as it's own use, it was originally released to a specified number of universities and firms, before seeing a general release. As of now, so far as my limited research can tell, it seems to be kept alive by a dedicated core group of users, like most languages that don't completely dominate the professional field.

3 Control Structures

Smalltalk's unique design comes from the fact that it's an OOP language built around message passing. This means most actual work inside the program is done by individual objects, or instances of objects, passing messages back and forth from one another. In some ways this works in quite recognizable manners, a method returning a value is thought of as one object invoking a method, and then getting the output back in the form of a message. There is no direct if statement. A value which evaluates to True or False is sent to ifTrue and ifFalse statements, which then have potential blocks of statements which in turn pass their messages on to whatever came above them. Loops work much the same, there are either iterations or regular while loops, which constantly test a given input and run a code block every time it evaluates to false. So far as I can tell, this regular control flow is fairly simplistic, not many more advanced forms exist, or at least advanced forms that I could easily find, such as switch cases or dowhile statements. As an OOP language, most of the rote control flow is handled by the design of the program itself, with some objects handling calling methods from other objects, passing messages between them to produce the correct output, which isn't a function of the language itself, but does imply a large amount of the expected design of programs written in this language.

4 Data Types

Following a similar format to the book Computer Programming Using GNU Smalltalk[2] we will be going over the most common classes which we will refer to as data types. In Smalltalk types are not declared and every type is originated from classes.

Boolean
These are binary represented values for a statement being true or false. The way we represent them is:

```
true
false
```

Number
A number in Smalltalk is represented by the Number class and below are some ways to declare them:

Natural numbers

```
1
10
100
```

Negative numbers

```
−1
−10
−100
```

Decimal base

```
1.0
6.9
3.14159
```

Scientific notation

```
5e1         [equals 50]
0.1e3       [equals 100]
0.12345e5   [equals 12345]
```

Fractions

```
1/2
2/3
3/2
```

Character
To represent a character in Smalltalk we need to insert a $ before the placement of the desired character. These characters are single symbols and range from digits, lowercase letters, uppercase letters and special symbols.

```
$a
$A
$1
$%
$$
```

String
In small talk strings is an object compound of multiple character objects. To represent strings we insert the desired string in between the single quote character '

```
'hello'
'12345'
'bob23 is cool'
```

Container
Similar to a String Smalltalk contains containers. Their characteristics are described in Figure 1.

Class	Characteristics
Dictionary	Like a real dictionary, it organizes information by a lookup key.
Array	Its elements are arranged in consecutive slots. Also, its size is fixed.
OrderedCollection	Like an array, but its size may increase or decrease.
Set	Its elements are not arranged in any order. Also, an object can occur in it at most once.
String	It contains 0 or more characters.
ByteArray	Like an array, but its elements are bytes.
SortedCollection	Smarter than an ordered collection, it maintains its elements in an ordering determined by some sorting criterion.
IdentityDictionary	A special, efficient dictionary, suitable mainly if the keys are **Symbol** or **SmallInteger** objects.
Stream (not a collection, but similar)	Smarter than a collection, it remembers where it was last accessed.

Figure 1: Describes the containers that exist in Smalltalk, this figure is retrieved from "Smalltalk, Objects, and Design"[3]

5 Sub-Programs

The primary way of breaking up the program that Small talk uses is it's object oriented nature. This will look fairly familiar to anyone who's ever done any OOP programming. A single large scale program can be broken up into multiple smaller classes. Some might just hold data, some just manipulate data, some drive the program, some manage interfaces between other programs, all the like.

This shows up at least partially in the project we have planned. The dungeon object will manage holding all the data for the dungeon itself, including rooms and where items are kept. The player object also interfaces with the dungeon object, asking if it can go into rooms or if it needs different stats, and getting it's stats altered as it passes from room to room. There will then be a basic larger driver object that prompts the user for commands, and then jumps to objects and tells them what the player just tried to do, which they in turn handle.

There are larger ways to do it in bigger programs. While small talk doesn't explicitly allow for interfaces, something akin to Java's observer and observable interface can be used for altering data storing objects changes to objects which interact with the data. Most other large scale object oriented programming models work just as well, such as model view controller interfaces used in many larger OOP programs to manage the sprawling code.

6 Summary

References

[1] Jigyasa Grover. https://learnxinyminutes.com/docs/smalltalk/.

[2] Canol Gökel. *Computer Programming using GNU Smalltalk*.

[3] Chamond Liu. *Smalltalk, objects and design*. ToExcel, 2000.

[1] [2] [3]