Introduction

(Overview • Structure of the manual • Acknowledgements)

Overview

Dialog is a domain-specific language for creating interactive fiction. It is heavily inspired by Inform 7 (Graham Nelson et al. 2006) and Prolog (Alain Colmerauer et al. 1972), and substantially different from both.

An optimizing compiler translates high-level Dialog code into Z-code (versions 8 and 5), a platform-independent runtime format supported by a wide range of interpreters and tools. A custom virtual machine called the Å-machine is also supported.

Dialog has been designed with three guiding principles in mind:

Lucidity: Story authors must be able to read, understand, and modify the standard library. The same programming language constructs should be used to express everything from world modelling and narrative structure to parsing and low-level utility functions.

Minimalism: The language should be elegant and small, and rest on a few powerful concepts that work well together. Being a high-level language, it should provide an abstraction of the underlying platform, and the details of the Z-machine should not shine through except where strictly necessary for performance reasons.

Performance: The interactive fiction community settled on the Z-machine as a common platform partly for nostalgic reasons. Yet the majority of recent Z-code games are unplayable on vintage hardware. The new language must be designed from the ground up with efficient object code in mind, both in terms of execution speed and memory footprint.

At its core, Dialog is a general-purpose, rule-based language in the *logic programming* family of languages. Most of the functionality pertaining to interactive storytelling, such as the world model and the parser, are implemented on top of this general-purpose language, in the form of a standard library. In other words, the library is implemented in terms of the same high-level constructs that are used to craft the story. Hopefully, this makes the standard library transparent to story authors, who are encouraged to extend and adapt the library to each particular story, as they see fit.

In addition to adhering to the principles listed above, Dialog features:

- A **concise, formal syntax**. Apart from a few built-in keywords and function names, Dialog source code is not confined to any particular natural language.
- A **comprehensive standard library** providing (among other features) an extensible parser that's capable of supporting arbitrarily complex actions. The standard library is currently limited to English stories, but it could be translated and adapted to other languages.
- A compiler that uses **advanced optimization** techniques to convert high-level rule definitions into compact, efficient Z-code that responds quickly to player input, even on resource-constrained platforms—including many vintage systems. Code compiled for the Å-machine can also take advantage of certain features of modern web browsers, such as CSS style declarations.
- A **fully open-source** implementation, provided under a 2-clause BSD license. Permission is granted to use, modify, and redistribute the compiler and standard library (with or without source code), in whole or in part, as long as attribution is given. Story authors are kindly asked, but not formally required, to include a traditionally-formatted banner in their works.

The name Dialog is suggestive of interactive storytelling, but it is also a nod to Prolog, which is a portmanteau of *programmation en logique* (programming in logic). That's why the name of the language appears to follow a North American spelling convention, even though the rest of this manual does not.

The official homepage of Dialog is: https://linusakesson.net/dialog/

Structure of the manual

The bulk of this manual is divided into two parts: **Part I** describes the Dialog programming language, starting with the basics and working its way through every language feature. This also serves as the official language specification. **Part II** describes the standard library, and starts with a hands-on tutorial, where a small game is built from scratch.

Readers who prefer a coherent step-by-step exposition, where every concept is explained before it is used, are encouraged to start with **Part I** and read all chapters in order.

Readers who are itching to get started should begin with **Part II**, play with the various examples, and optionally follow the hyperlinks back to **Part I** in order to dig deeper into the underlying mechanisms.

Acknowledgements

Dialog has been my on-and-off spare time project for many years now, and I'm very happy to have reached a point where it can be released to the public. Working on Dialog has been an extraordinary journey for me personally, but I wouldn't have gotten very far without the maps and lanterns strewn about by the brave adventurers who navigated

these twisty passages before me.

The *Z-machine Standards Document* has been an invaluable source of information and insight. I stand in awe of the dozens of people who reverse-engineered and documented the Z-machine back in the 1990s, and described their findings in such a well-written and concise reference work. Thank you for that **David Beazley**, **Kevin Bracey**, **Paul David Doherty**, **David Fillmore**, **Russell Hoare**, **Mark Howell**, **George Janczuk**, **Stefan Jokisch**, **Marnix Klooster**, **Mark Knibbs**, **Peter Lisle**, **Graham Nelson**, **Jason C. Penney**, **Matthias Pfaller**, **Andrew Plotkin**, **Matthew Russotto**, **Chris Tham**, **Mike Threepoint**, and **Dannii Willis**.

Writing and debugging a compiler involves hours of staring at disassembled code, and I thank **Mark Howell** and **Matthew Russotto** for creating and maintaining the ztools package, in particular infodump and txd.

I've also used and abused the highly hackable Z-code interpreter dumbfrotz, part of the frotz family, for which **Stefan Jokisch**, **Galen Hazelwood**, **David Griffith**, and **Alembic Petrofsky** deserve credit.

Last but not least, I extend my deepest gratitude to **Graham Nelson** and (where applicable) **Emily Short** for developing and releasing Inform 6 and 7, for writing the *Inform Designer's Manual* and *Writing with Inform*, and for distilling so many years of community experience into the default Inform world model and Standard Rules. These treasure troves are overflowing with so many good ideas that it would have been irresponsible not to steal a substantial part of them, and incorporate them into my own standard library. Any shortcomings in the Dialog world model are of course my own fault.

Lund, November 2018

Linus Åkesson

Onwards to “Software” • Back to the Table of Contents