Programming FrameMaker® Shortcuts

by Todd Brasel

Contents

Overview

Defining alternate shortcuts and creating new menu commands is a handy technique for improving your productivity with FrameMaker. Because you have to take your hands from their position on the keyboard and make a mental sidestep to remember the appropriate key sequence, most of the standard FrameMaker shortcuts present a noticeable interruption to workflow. It's not a very "fluid" motion. It often seems as though you're better off just using the mouse.

For example, the sequence for importing a file is `Esc f i f`, and for inserting a cross-reference you type `Esc s c`. FrameMaker shortcuts are also case-sensitive, and I've frequently struggled to remember if the shortcut for inserting conditional text is `Esc s C` or `Esc s c`.

This article describes a technique for programming new commands, assigning those commands to menus, and creating shortcuts for them. It also describes how to add new shortcuts to default commands.

To fully understand this article, you'll need an in-depth understanding of FrameMaker and a basic understanding of a scripting language, such as JavaScript, and its syntax.

Changing the shortcuts

To change FrameMaker shortcuts, I had to create a custom menu configuration file. This is a simple text file that contains codes which tell the application how to modify its menus and shortcuts. These codes follow a simple syntax and are about as easy to implement as JavaScript.

Modifying a default command

This code block tells FrameMaker to add a new key sequence and label to the `Cross Reference` command. For exampe, to add the `Shft Ctrl R` shortcut to the `Special > Cross Reference` menu, I used the following code:

```
<Modify CrossReference
   <KeySequence +^R>
   <KeySeqLabel Shft+Ctrl+R>
>
```

Here's how it works:

- `Modify` tells FrameMaker to change the parameters of a command that is part of FrameMaker's default command set.

The trick is to find the actual name of the command you want to modify. Because I didn't have a copy of the FrameMaker Developer's Kit (FDK) ready, I had to open the file `../FrameMaker/fminit/cmds.cfg`, and search that file for a command name that seemed to match what I needed to do. In this case, I searched for "Reference" and found `CrossReference`, an obvious choice.

- `KeySequence` identifies the key sequence you will use to activate the command.

I used the Adobe convention, which uses a `+` for `Shift` and `^` for `Control`.

- `KeySeqLabel` identifies what the menu will actually display.

Note that each command must be surrounded by angle brackets <>. FrameMaker ignores text outside of a pair of angle brackets.

Creating a new command

To make a new command and add it to a menu, you need to follow three steps:

1. Define the command
2. Assign the command to a menu
3. Assign a place on that menu for the command.

For this application, I created a command called "ViewFormatting" that shows or hides borders, rulers, and text symbols.

Define the command

This code block defines the new command:

```
<Command ViewFormatting
   <Label Borders, Rulers,
   Symbols>
   <KeySequence +^A>
   <KeySeqLabel Shft+Ctrl+A>
   <Definition \x361 \x362 \x363>
   <Mode All>
>
```

Here's how it works:

- `Command` tells FrameMaker to add a new command. I gave it a meaningful name that fit with Adobe's intercap-style naming convention.
- `Label` identifies how FrameMaker will display the command on a menu.
- `KeySequence` identifies the key sequence you will use to activate the command.

I used the Adobe convention, which uses a + for `Shift` and ^ for `Control`.

- `KeySeqLabel` identifies what the menu will actually display.
- `Definition` tells FrameMaker what this command will do.

A simple way of creating a new command is to combine the functions of other commands. I searched through the `cmds.cfg` file for the other commands I wanted to use, then copied their definitions into this code block. For example, the definition for `ViewBorders` is `\x361`. You can find more information about these definitions in the FrameMaker Developer's Kit.

- `Mode` tells FrameMaker if the command is a regular command or a FrameMaker Math command. The option I used is `All`, which tells FrameMaker that this command can be used in both modes.

Assign the command to a menu

The next code block tells FrameMaker on which menu to display the new command:

```
<Add ViewFormatting
   <Menu ViewMenu>
>
```

How it works:

- `Add` tells FrameMaker to add the command you've created.
- `Menu` identifies the name of the menu.

I searched the `cmds.cfg` file for the name of the menu.

Assign a place for the command

The last code block tells FrameMaker where to display the command on the menu (after the View Rulers command):

```
<Order ViewMenu.ViewFormatting
   <After ViewMenu.ViewRulers>
>
```

How it works:

- `Order` tells FrameMaker to assign the `ViewMenu.ViewFormatting` object to the place you specify.
- `After` tells FrameMaker to place it after the command you specify.

You could also use `Before` to place it before a command, or `First` and `Last` to place your command first or last on the specified menu. Note that FrameMaker uses an object-oriented syntax to specify the position of commands on a menu.

Installing the modifications

To use the modifications, save the code file as `customui.cfg` in the path `../FrameMaker/fminit/configui`. Then restart FrameMaker. You can start using the modifications immediately.

To use the new shortcuts, just hold down the `Shift` and `Control` keys with your left hand and press the appropriate letter key with your right. It's a more fluid movement than using the `Escape` key sequences.

In case you forget the shortcuts, they will display next to each command on the menus. Of course, if you're a FrameMaker purist, the original key sequences still work.

List of shortcuts

The new key sequences in the attached file are described in Table 1, *New Key Sequences*.

Removing the modifications

To remove the modifications, just remove the `customui.cfg` file from the `configui` folder, or rename the file. When you next launch FrameMaker, the menus and shortcuts will be back to normal.

Making your own modifications

To make your own modifications, feel free to alter the code examples I've provided in the **Code listing** on page 5. The files `sample.cfg` and `cmds.cfg`, in the `configui` folder, provide many useful examples and an ad-hoc menu reference.

You'll also find a more detailed explanation of the process and concepts I've outlined here in the `Customizing_FrameMaker_Products.pdf` file. This file is in the path `../FrameMaker/OnlineManuals`.

Table 1: New Key Sequences

Command	Current	New
Insert cross-reference	`Esc s c`	`Shft Ctrl R`
Delete page	`Esc s p d`	`Shft Ctrl D`
Import file	`Esc f i f`	`Shft Ctrl I`
View Rulers	`Esc v r`	`Shft Ctrl Y`
View Text Symbols	`Esc v t`	`Shft Ctrl U`
View Borders, Symbols and Rulers	(no equivalent shortcut)	`Shft Ctrl A`

Code listing

This is the code I wrote to add the commands described in **Table 1** on page 4. FrameMaker ignores any text outside of a pair of angle brackets. No header information is needed for the file.

```
<Modify ImportFile
   <KeySequence +^I>
   <KeySeqLabel Shft+Ctrl+I>
>

<Modify CrossReference
   <KeySequence +^R>
   <KeySeqLabel Shft+Ctrl+R>
>

<Modify !PageDelete
   <KeySequence +^D>
   <KeySeqLabel Shft+Ctrl+D>
>

<Modify ViewRulers
   <KeySequence +^Y>
   <KeySeqLabel Shft+Ctrl+Y>
>

<Modify ViewTextSymbols
   <KeySequence +^U>
   <KeySeqLabel Shft+Ctrl+U>
>

<Command ViewFormatting
   <Label Borders, Rulers, Symbols>
   <KeySequence +^A>
   <KeySeqLabel Shft+Ctrl+A>
   <Definition  \x361 \x362 \x363>
   <Mode All>
>

<Add ViewFormatting
   <Menu ViewMenu>
>

<Order ViewMenu.ViewFormatting
   <After ViewMenu.ViewRulers>
>
```

About the author

Todd Brasel is a consultant with Documentation Strategies, Inc. in Rensselaer, NY and is an adjunct instructor of Computer Science at The Sage Colleges, Albany, NY. He has used FrameMaker to create software manuals, training materials, catalogs, and (with WebWorks Publisher) a Microsoft Word-to-XML document conversion process.

His email address is brasel@docstrats.com.