The Dyog Reference Manual

Craig D. Vargas

1. Introduction

Dyog is a computer language designed to implement video games. The Dyog syntax has many resemblances to conventions of C, C++, and Java, particularly the declaration of variables and decision-making. The syntax rules of Dyog that do not resemble C and Java conventions are still very straight-forward and intuitive, and thus are very easy to use.

2. Lexical Conventions

In Dyog there are six kinds of tokens: identifiers, strings, keywords, files, operators and other separators. As in C, C++, and Java all whitespace is treated the same and serves only the purpose of separating tokens.

2.1. Comments

A comment starts with the characters "*" and ends with the characters "*/". Placing these sets of characters within a Dyog comment is restricted.

2.2. Identifiers

Identifiers consist of any alphanumeric character including the underscore character, "_". Identifiers must, however, begin with a character from the alphabet. The Dyog parser is case "sensitive" thus upper and lower case letters are considered different.

2.3. Keywords

The following is a list of Dyog's keywords. All words included in this list are reserved and have specific purposes thus they may not be used in any other fashion.

a. background
b. item
c. show
d. clear
e. choose
f. print
g. commandline
h. set
i. position
j. speech
k. caption

2.4. Strings

A sting is a sequence of characters that are enclosed between a pair of double quotation marks. Strings are used for speech bubbles, caption boxes, and also to print to the command line. In order to express a double-quote character within a string, it must be preceded by another double-quote like the following example:

“the man said “”yes I will””””

2.5. Files

A file is a sequence of characters that are enclosed by square brackets, “[]”. Files are associated with item and background datatypes. This construct tells the computer that the text inside the square brackets is the name of a file that the programmer intends to use. A file must contain a period with three alphabetic characters preceding the period, also known as the file extension. Primarily files used in Dyog will have a .gif or .jpg extension. Below is an example of a file in Dyog.

[picture.gif]

3. Syntax notation

In this manual all valid Dyog syntax will be printed in italic font.

4. Datatypes

Dyog has 4 datatypes: backgrounds, items, speech bubbles, and caption boxes. In this paragraph the purpose and proper usage of each datatype will be explained.

4.1. Background

Background datatypes hold a link to a file that is meant to be the background of the graphical user interface (GUI). Background files are assumed to encompass the majority of the GUI. As a result when a background file is displayed to the screen it will always be centered in the same position. A background file type should have a .gif extension but can also handle a .jpg extension. The following is an example of a declaration of a background file:

background bkgd_1 [forest.gif];

4.2. Item

Item datatypes hold links to picture files that are used to supplement the setting of the GUI. Item datatypes are not meant to encompass the entire GUI thus, when displayed to the screen, coordinates need to be supplied to explain were to position the item. Item datatypes should also have either a .gif or .jpg extension. The following is an example of how to declare an item datatype:

item tree [tree.gif];

4.3. Speech

Speech datatypes, as implied by their name, hold information about a character’s speech. The text to be displayed can be supplied during the speech declaration or somewhere later in the code. The following are examples of usages of speech datatypes.

speech sp1 “How are you?”;

speech sp2;
sp2 <- "I am good thank you.";

4.4. Caption

The caption datatype is similar to the speech datatype except it is meant to express supplementary information rather than character speech. The following is an example of how to use the caption datatype:

caption cpt1 "The plot thickens";
caption cpt2;
cpt2 <- "What will Jaime do?";

5. GUI manipulation

The following are a list of commands needed to manipulate the GUI. The list of the commands are as follows: show, clear, print, commandline, set, and position.

5.1. Show

The "show" command displays the current GUI configuration to the screen. The configuration can be changed as many times as desired but the actual display will not be affected until the show command is invoked. Below is an example of how to invoke the "show" command:

show;

5.2. Clear

The "clear" command erases all items and backgrounds from the current GUI configuration. No change will be seen in the display of the GUI until the "show" command is invoked again. The following is an example of how to use the "clear" command.

clear;

5.3. Print

The "print" command displays text directly onto the GUI. Text displayed by the print command is different from that of captions and speech because the text displayed by the print command will automatically show up on the GUI regardless of whether or not the "show" command has been invoked. It can be thought of as an immediate display of text. The "print" command is used as follows:

print "Here is the immediate message";

5.4. Commandline

The "commandline" statement is used to display output to the command line. The "commandline" statement must be followed by the string that is to be displayed to the command line. The "commandline" statement is analogous to the "cout" statement in C++. The following is an example of how the "commandline" statement is used:

commandline "You are now reading form the command line\n" ;

5.5. Set

The "set" command will add a background file to the current GUI configuration. The "set" command must be followed by the name of the background to be added. Below is an example of how to use the "set" command:

background bkgd_1 [forest.gif];
set bkgd_1;

5.6. Position

The "position" command is used to add an item, speech, or a caption to the current GUI configuration. The position command must be followed by the name of the item to add, the x-axis coordinate and the y-axis coordinate of where the top left corner of the item should be placed. The information needs to be supplied in that order separated by whitespace. The position command should be used like this:

item tree [tree.gif];
position tree 35 100;

6. Operators

The following is a list of the operators used in the Dyog language.

6.1. Push (<-)

The "push" operator is a binary operator used to associate a caption or a speech with a set of text. The name of the speech or caption must be placed on the left side of the operator and the string to be associated with that text or caption must be placed on the right side of the operator as show below:

caption cpt1;
cpt1 <- "It is a sunny day";

7. Control flow

Dyog's main construct for establishing control in a program is the "choose" structure. The "choose" structure allows the program to make decisions. Once the computer hits a "choose" structure it will prompt the user for input. Based on the input given the computer will then follow a certain "choice" in the "choose" structure. This functionality is similar to C++'s and Java's "switch" statement. Below is an example of how a "choose" statement can work:

```
choose{
  choice 1:
    print "you have entered the value '1', you loose";
  end
```

```
  choice 2:
    print "you have entered the value '2', you win";
  end
}
```

The "end" statement after each choice tells the computer that the code for that choice ends there.

8. The Finish statement

The "finish" statement tells the computer to stop execution of your program. A finish statement can be placed anywhere in the program and can also occur multiple times in a program. Using a finish statement looks like this:

```
finish;
```

Appendix A

Sample Code

```
/*Simple Video Game 1*/

/*Define variables for the backgrounds that will be used*/
background house [house.gif];
background restaurant [restaurant.gif];
background football [fbfield.gif];

/*Define variables for the items that will be used*/
item bob [man.gif];
item mary [woman.gif];
item mary2 [woman_mad.gif];

/*Define necessary speech and captions datatypes*/
speech bsp;
speech msp;
caption cpt;
caption options;

/*Initialize caption and speech datatypes
bsp <- "I can't wait to get to the football game";
msp <- "I'm hungry lets get food first";
cpt <- "If you stop to get food first you may miss more than half of the game";
options <- "Press '1'  to go straight to the football field\n Press '2' to stop for food
first";

/*Organize the GUI*/
set house;
position bob 50 100;
position mary 150 100;
position bsp 50 150;
position msp 150 150;
position cpt 200 50;
position options 500 1000;

/*Display GUI to screen*/
show;

/*Poll for user input*/
choose{
  choice 1:
    /*The user picked choice # 1 so they loose*/
    clear;  /*clear the last GUI configuration*/
```

```
   /*Set up the next GUI configuration*/
   cpt <- "Mary is bitter because she was hungry all day.  Tonight you sleep on the
couch.  You loose.";
   set football;
   position bob 50 100;
   position mary2 150 100;
   position cpt 200 50;
   show;
  end;

  choice 2:
   /*User picked choice # 2 so they win*/
   clear;  /*clear the last GUI configuration*/

   /*Set up the next GUI configuration*/
   cpt <- "Mary is not hungry anymore and the game was delayed so you don't miss a
minute.  You win."
   set restaurant;
   position bob 50 100;
   position mary 150 100;
   position cpt 200 50;
   show;
  end;

}

finish;  /*End of program*/
```