Dyog Language Reference Manual

Introduction

1. Introduction:

Dyog is a simple language that allows users to design electronic storybooks. Anything from comics to children's books can be designed with Dyog. The user is able to design each page of the book by defining what backgrounds to use, what characters and props to put in the foreground, and what text to insert to tell the story. Dyog originally started as an idea of a language to design video games, but later changed direction into its current form due to the creative interests of the design team. The Dyog compiler will produce two files: DyogStory.java and DyogStory.class. The user created story can be viewed by running the latter file on a Java interpreter. Dyog automatically takes care of all of the complicated GUI issues needed for the Java interpreter so that the programmer can concentrate all of their focus on creativity.

1.1 Dyog Overview:

Dyog allows the programmer to design a storybook with his or her own:

a. Backgrounds
b. Characters
c. Background props
d. Dialogue
e. Narrative

1.1.1 Backgrounds:

Dyog allows users to create their own backgrounds for use in their story. In general a background file is a picture file whose dimensions are 700 X 500 pixels or 700 X 600 pixels if the user wishes not to use the Narrative Box. Background files are stored in the "background" datatype. Background files can be smaller than the specified sizes but there will be empty gray-space on the screen. A background file can be larger than the previously given dimensions but some of the picture will be cut off.

1.1.2 Characters:

Users can create their own unique cast of characters with the Dyog language. A any picture file that is small enough to fit within the Dyog screen can be used as a character. Characters are stored in the "item" datatype. If the file used for a character is larger than the Dyog screen, then parts of the picture will be cut out and any picture loaded to the screen prior to itself will be hidden under it. The order in which pictures are loaded will be discussed later.

1.1.3 Background Props:

Different props can be added to a background to customize and enhance it. Background props follow the same rules as characters and are also stored in "item" datatypes.

1.1.4 Dialogue:

Dialogue, in Dyog, is expressed by placing text on the screen and drawing a line from the text to the character that is speaking. In Dyog this information is stored in the "speech" datatype.

1.1.5 Narrative:

Narrative in Dyog, is achieved by simply placing text to the screen. It is like dialogue but narrative is not spoken by any specific character so there is no line to be drawn. The Dyog screen also comes with a Narrative Box, which is located at the bottom of the screen at coordinates: (0,500) and has a length of 700 pixels and a height of 100 pixels. This box is simply extra space for the user to display the storyline.

1.2 Overall:

Dyog gives the user an easy avenue to design their own creative electronic book without having the hassle of worrying about the Java GUI details or the logic that goes behind displaying and executing the book. Dyog has very simple and straightforward syntax that allows any creative writer the ability to do this.

Language Tutorial

2. Language Tutorial

2.1 Compilation:

Dyog is a simple language and easy to use. All Dyog code should be placed into a .txt file. Dyog code is compiled in the following manner:

java Main < code.txt

Where code.txt is the file that contains the Dyog code. Then a DyogStory.java file is created that can be compiled using the java compiler like so:

javac DyogStory.java

Then a DyogStory.class file is created that can be run using the java interpreter like so:

java DyogStory

On a windows operating system the user can use the dyogc or the dyogcr command to either compile or compile and run their program respectively. These commands are invoked by placing the name of the file that contains the Dyog code after the command e.g.:

dyogc code.txt
Or
dyogcr code.txt

2.2 Statements:

In Dyog, statements cannot be nested within each other. Due to the nature of the language they are just listed one after another and are separated by semicolons. A semicolon tells the Dyog compiler that the current statement is ending and any code listed after belongs to the next statement.

2.2.1 Declarations:

The first part of a Dyog program is the variable declarations. All Backgrounds used and Items, speech bubbles and captions placed on the screen must be declared in the beginning of the program. If variable declarations are placed in other parts of the program then the program may not function properly. Backgrounds are datatypes that hold picture files that are intended to be used as a background for one or more pages of the story. An item datatype holds picture files that can be used as characters in the story or background props. The speech datatype holds text that is intended to be character dialogue. The caption datatype holds text that is used to explain the storyline. Here are some examples of how to declare these variables:

The following code declares a background datatype called "street" that holds the file "city_street.jpg":

background street [city_street.jpg];
The following code declares an item datatype called "Sam" that holds a file named "boy.jpg":

item Sam [boy.jpg];

The following code declares a speech datatype called "greeting" that holds the text "Good morning":

speech greeting "Good morning";

The following code declares a caption datatype called "story" that holds the text "It was a beautiful morning on Summer Street":

caption story "It was a beautiful morning on Summer Street";

2.2.2 Dyog Pages:

After the declarations are made the pages of the book can start being defined. This is done by using the "set", "position", "spchloc", "caploc", and "show" commands. The "set" command is used to tell the compiler what background to display. The "position" command is used to tell the compiler what items to display and where to display them. The "spchloc" command is used to define what speech datatypes to display to the screen and where to display them. The "caploc" command defines what captions to display to the screen and where to display them. Finally the "show" command tells the compiler that everything above it, up to the last "show" command or the declarations section, is supposed to be together on one page and anything after it, up to the next "show" command, is supposed to go on the next page. Here are some examples of how these commands work.

If you wanted to display the background “street” that was declared earlier it would be done in the following fashion:

set street;

Placing the item “lion” in the middle of the screen is achieved like this:

position lion 350 250;

Making the lion say the text stored in the variable “greeting” is done like this:

spchloc greeting 300 200 350 250 BLACK;

Note: The first set of coordinates in the “spchloc” command define where the text will be placed. The second set of coordinates tells the compiler where the character that will be saying the text is located. The last part of the statement tells the compiler what color to make the text. There are two possible colors, black and white, and they have to be written in all capital letters or else the compiler will not recognize it. If the compiler does not recognize the color specified then it will use the default color black.

And displaying the caption “story” in the top left corner of the Narrative Box in white text is done like this:

caploc story 0 500 WHITE;

Finally, to make this one complete page, the “show” command is invoked like so:

show;

Another page will be started if any of the previous commands are invoked, or the story will end if the “end” command is invoked like so;

end;

2.3 The following is a simple example of a Dyog program:

```
background night [bk.jpg];
background sunny [sunnyday.jpg]; //blah blah
item hero [zelda.jpg];
item monster [monster.jpg];
//more blah
speech talk1 "hello monster";
speech talk3 "";
caption talk2 "";
caption talk4 "They clash";

talk2 <- "They meet";
talk3 <- "hello hero";

set sunny;
position hero 100 300;
position monster 300 300;

spchloc talk1 100 200 100 300 WHITE;
spchloc talk3 500 200 300 300 BLACK;
caploc  talk2 300 50 WHITE;

show;

talk1 <- "Lets fight";
talk3 <- "ok";
set night;
position hero 200 300;
position monster 250 300;
spchloc talk1 150 200 200 300 BLACK;
```

```
spchloc talk3 350 200 250 300 WHITE;
caploc talk4 300 100 BLACK;

show;
end;
```

Dyog Language Reference Manual

3. Lexical Conventions:

There are seven types of tokens in the Dyog language: Identifiers, strings, files, operators, keywords, and comments and integer constants. All whitespace is ignored by the Dyog compiler and only serves the purpose of separating tokens.

3.1 Identifiers:

Identifiers are used to name variables. They must start with an alphabetic character and after that they may consist of as many alphanumeric characters as desired. The underscore is also included in that set of alphanumeric characters. The Dyog compiler is case sensitive so the identifiers: Mike, mike, MIKE, and mikE are considered different identifiers.

3.2 Strings:

A string is a cluster of text that resides between two double quotation marks. If a Double quotation mark is desired within the string it must be denoted by two double quotes back to back.

3.3 Files:

A File token is enclosed inside of a set of square braces and must end with a period and a three-letter extension. Dyog actually only supports .jpg and .gif files but the compiler will accept any extension.

3.4 Operators:

Dyog only has one operator, the push operator. The push operator is denoted by the following set of characters: "<-". This usage of this operator is described later in this manual

3.5 Keywords:

The following keywords are reserved and cannot be used as identifiers.

a.) background
b.) item
c.) speech
d.) caption
e.) set
f.) position
g.) spchloc
h.) caploc
i.) show
j.) end
k.) WHITE
l.) BLACK

3.6 Comments:

Comments in Dyog begin with a double backslash, "//", and end at a newline character. All comments are ignored by the Dyog compiler and server only to aid the programmer.

3.7 Integer Constants:

Integer constants are a sequence of digits. All integer constants in Dyog are interpreted in decimal form.

4. Syntax Notation:

All Dyog syntax used in this manual will be denoted by *Italic* font. All literal words and characters will be denoted by ***<Italic Bold font placed inside greater-than/less-than brackets>***.

5. Datatypes:

There are four datatypes in Dyog: backgrounds, items, speeches, and captions. All of these datatypes share the same namespace so, for example, an identifier cannot represent both a background and a caption datatype. Backgrounds and items are bound to the values assigned to them during declaration but speeches and captions are variables whose text can be changed during the program.

5.1 Backgrounds:

The background datatype holds information of a picture file that is used to encompass the background of a screen or page in the Dyog book. The screen's dimensions are 700 X 500 or 700 X 600 including the Narrative Box. Using a background with the latter dimension will cause the Narrative Box to be hidden. Using backgrounds that are larger or smaller than the screen dimensions are allowed but parts of the background will be cut or parts of the screen will be appear blank. Here is the formal syntax of how to declare a background datatype:

background ***<IDENTIFIER> [<FILE_NAME>];***

5.2 Items:

The item datatype holds information of a picture file. Items can be used as characters in the story or props to supplement the background. The order in which items are declared is significant. If the programmer wants to place one item in front of another item on the screen, for instance if they want to place a picture of a squirrel in front of a picture of a tree, the user has to declare the item that will be in front after the item that will be in the back. In general items in the background should be declared before items in the foreground. This is only an issue if multiple items are going to be placed in the same region on the screen. Otherwise the order of the declarations is not an issue. Here is the formal syntax of how to declare an item datatype:

item ***<IDENTIFIER> [<FILE_NAME>];***

5.3 Speeches:

The speech datatype holds information for character dialog within the story. When a speech variable is declared it has to be initialized with a string literal. The empty string is sufficient for this initialization. When a speech variable is displayed to the screen a line is drawn from the text to the character saying the text. This is explained later in the manual. The formal syntax is as follows:

speech ***<IDENTIFIER>*** *"****<STRING_LITERAL>****";*

5.4 Captions:

The caption datatype holds narrative information that is used to tell the story. Like a speech variable, captions also have to be initialized during the declaration. The formal syntax to do so is:

caption ***<ITENTIFIER>*** *"****<STRING_LITERAL>****";*

6. Operators:

Dyog only has one operator, the push operator. The push operator looks like this: "<-". It is used to change the textual data that resides in a caption or speech variable. It is used by placing the identifier of the caption or speech variable to the left of the push operator and putting the new string literal to the right of the operator. This overwrites the text data in the variable with the new string literal. The formal syntax goes like this:

<IDENTIFIER> *<- "****<STRING_LITERAL>****";*

7. Commands:

Commands are used to display the different Dyog datatypes to the screen and define the flow of the story. Each datatype has a unique command that must be used in order to display it to the screen. The "set" command is used for backgrounds, the "position" command is used for items, the "spchloc" command is used for speeches, and the "caploc" is used for captions. The other Dyog commands: "show" and "end", define the flow of the story.

7.1 Set:

The "set" command is used to display the background datatypes to the screen. The top left corner of the background file is always placed at the top left corner of the screen. The "set" command is used by placing the name of the background datatype to the right of the command like so:

set ***<IDENTIFIER>****;*

7.2 Position:

The "position" command is used to display the item datatypes to the screen. The programmer must specify at what coordinates to place the top left corner of the item. The "position" command is used by placing the name of the item datatype to the right of the command, followed by the X and Y coordinates of where to place the top left corner of the picture. E.g.:

position ***<IDENTIFIER> <X_COORDINATE> <Y_COORDINATE>****;*

7.3 Spchloc:

The "spchloc" command is used to display the speech datatypesto the screen. The coordinates, of where to place the text and where the character that will be saying the text is located, must be specified when using the "spchloc" command. The text stored in a speech variable can be displayed in black or white colored font. This command is used by placing the name of the speech variable to be displayed to the right of the command, followed by the X and Y coordinates of where to place the text, followed again by the X and Y coordinates of where the character is located, followed one more time by the color of the text. This is done like so:

spchloc ***<IDENTIFIER> <X_COOR> <Y_COOR> <X_CHAR> <Y_CHAR> <COLOR>;***

NOTE: X_COOR and Y_COOR denote the X and Y coordinates to place the text. X_CHAR and Y_CHAR denote the X and Y coordinates of the location of the character. Finally COLOR is the color of the text that can be either: "BLACK" or "WHITE". The color must be specified in all capitals and if an invalid color is entered then the Dyog compiler will use the default color of black.

7.4 Caploc:

The "caploc" command is used to display the caption datatypes to the screen. When invoking this command the programmer must specify the X and Y coordinates of where to place the text. The Dyog screen comes with a Narrative Box that is located at the coordinates: (0,500). The Narrative Box is intended to be the designated spot for captions that tell the story but a caption can be put anywhere on the screen. The Narrative Box does not have to be used and, as explained earlier, can be hidden by displaying a big enough background. The "caploc" command is used by placing the identifier for the caption variable to the right of the command followed by the X and Y coordinates of where to place the text, followed again by the color of the text. E.g.:

caploc ***<IDENTIFIER> <X_COORDINATE> <Y_COORDINATE> <COLOR>****;*

7.5 Show:

The "show" command is used to tell the Dyog compiler that the current page/screen has finished being defined and that any other datatypes that are displayed to the screen after the show command will be displayed on the next page/screen. When the show command is invoked the screen configuration is reset for the next page. If the programmer wishes to use the same background or characters in two back-to-back pages then they must repeat the appropriate "set" and "position" commands on both pages to do so. The "show" command is invoked like so:

show;

7.6 End:

The "end" command is used to tell the Dyog compiler that the story is over and no more pages/screens will be defined. Once the compiler sees the end command it will stop reading input from the input file. The end command also automatically sets up the next page in the storybook to display the text: "The end". This command is used like so:

end;

8. Pages/Screens:

The notion of a page or screen has been talked about during this document but has not yet been explained. Once the programmer has made a full program in Dyog, the program can be run and the story can be viewed. The user views the story by cycling through the pages of the book. Typically each page will show a different screen configuration. The way the programmer designs a page or screen is by defining what background to use, and what item, caption, and speech variables to use and where to display them. Then the programmer uses the "show" command and Dyog will

interpret everything before that command, up to the last "show" command, to be one complete screen configuration.

9. Scope rules:

Dyog only contains one scope, the whole program. Variable scope starts at its declaration and ends at the end of the program when the "end" command is invoked. A screen's scope starts at the first command used since the last "show" command was invoked, and ends when the next "show" command is invoked. Except the first screen's scope which starts at the first command of the program.

10. Statement rule:

A statement is either a declaration or a command. All statements must end with a semicolon and statements cannot be nested within each other. Statements are simply listed one after another.

11. Conversions:

All Dyog datatypes are unique and cannot be converted, typecasted or used with other types commands. No conversions are allowed within a Dyog program.

12. Dyog GUI:

The Dyog graphical user interface consists of a box that is 800 x 600 pixels in size. It is separated into three components: The Dyog Screen, the Narrative Box, and the Control Bar. The Dyog Screen and Narrative Box have been discussed already. The Control Bar consists of two buttons: The "Next" button and the "Previous" button. These buttons are used to cycle between the pages of the storybook. When the reader of the book wishes to exit they only have to click the close icon on the top right of the Interface.

Project Plan

13. Process:

Designing even a small language is a very complicated task because there is much to be considered about the syntax and semantics of the language itself and implementing it, in effect, requires the creation of a whole system of software components.

When first designing the syntax and semantics of the Dyog language I thought to myself: "What are the bare essentials?" In other words, what is the minimum amount of information that the user must supply to create an electronic storybook? The reason I considered the bare minimum was because I wanted to language to be as easy and straightforward as possible. Obviously a book needs scenery, characters, and text to tell the story. I took these components and made each component its own datatype in the language. Then I figured out what information does the user need to supply about each datatype. Once that was figured out I knew what each datatype needed to hold and what it took to display the necessary components onto the screen. I decided to have separate commands to display each datatype because each datatype needed to supply different information to be displayed. Some needed coordinates, some needed text and coordinates, and some needed nothing at all. Once that was figured out I knew what types of statements my language was going to consist of so I was able to start designing the lexer and parser. Throughout the development lifecycle some changes were made to these two components but for the most part they were finished early. Once these two were implemented it was time to work on the tree walker. Early in the design it is hard to get a grasp of everything that is needed inside of the tree walker. Due to this I decided to make the first version of the tree parser simply parse the tree and output each leaf in the tree so that I could be confident that, when it was time to use them inside the tree walker, I knew how to access all the data I needed. Once the preliminary tree walker was designed I stepped away from the compiler all together and asked myself: "If I was designing a storybook in java, what would the code look like?" Thus I spent time designing a java program that displayed an electronic storybook and I tried to make things as general as possible so that they could be used in any book. Once I got that program working I figured out what parts of this code I made would be the same for all programs and what parts would be different depending on the storyline. I went through the code and separated into parts that would be identical from program to program and parts that would not. Then I started to make the file DyogDataOut.java which handled most of the backend compiler work. While doing this I also had to modify the java classes for the datatypes in my language so that they interfaced with DyogDataOut.java. Each datatype had simple fields to store necessary data and simple functions to change and retrieve that data. The DyogDataOut.java file, however, does 90% of the backend work. It stores all variables defined in the program, does part of the static semantic checking, and outputs all target language code. The tree walker used this file and passed it all the data needed to output the target language code, and in turn DyogDataOut.java passed the tree walker error checking information so that the walker could output the correct error messages. Once all these systems were complete a few tweaks had to be made here and there but for the most part the language was finished.

14. Programming style:

I was a one-man team so I used my usual programming style, which is pretty standard among well-organized programs.

15. Project Timeline:

Date	Goal
September 18, 2004	Determine idea for language
October 17, 2004	Finish Lexer and Parser
November 21, 2004	Finish Tree Walker
December 12, 2004	Finish all Backend Classes
December 17, 2004	Finish Test Suite
December 19, 2004	Finish Report

16. Software Development Environment:

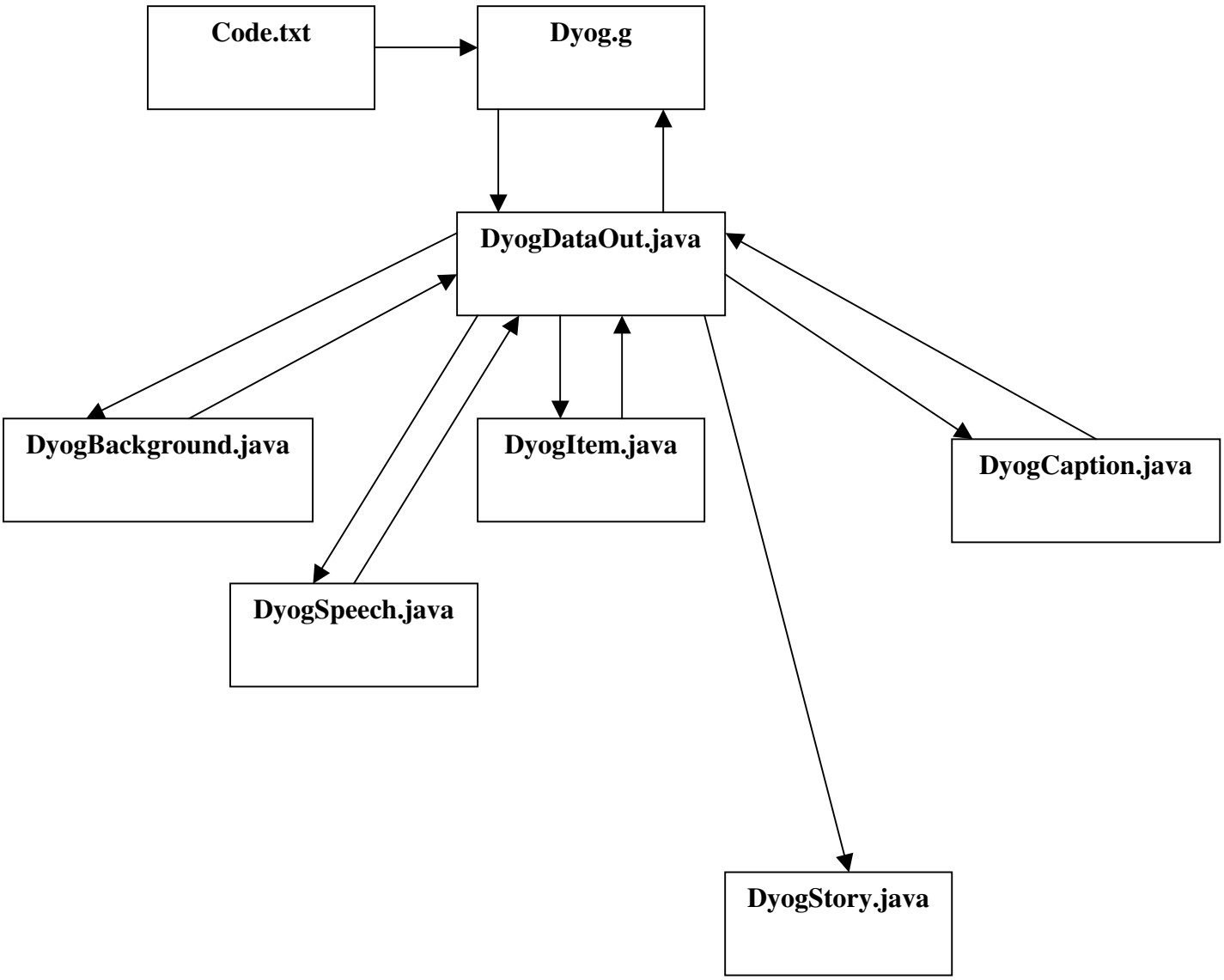
- OS: Windows 98
- Language: Java SDK 1.4.2
- Software Tool: Antlr 2.7.4
- Text Editor: Textpad 4.7.3: 32-bit edition

17. Project Log:

File Name	Date Completed
Lexer	October 21, 2004
Parser	October 21, 2004
Tree Walker	December 10, 2004
Dyog.g	December 10, 2004
DyogBackground.java	November 24, 2004
DyogItem.java	November 24, 2004
DyogSpeech.java	November 24, 2004
DyogCaption.java	November 24, 2004
DyogDataOut.java	December 10, 2004
Test Suite	December 17, 2004

Architectural Design

18. Block Diagram:



19. Interfaces between components:

19.1 Dyog.g:

Dyog.g creates an object of type DyogDataOut. This object stores all variables and gives DyogDataOut.java all of the information needed to generate the code in the target language.

19.2 DyogDataOut.java:

DyogDataOut.java creates vectors of each datatype in order to hold all necessary variables. Also generates code in the target language.

Interfacing Methods:

- addBackground(DyogBackground b) – Adds the given background datatype to the array of backgrounds.
- addItem(DyogItem i) – Adds the given item datatype to the array of items.
- addSpeech(DyogSpeech s) – Adds the given speech datatype to the array of speeches.
- addCaption(DyogCaption c) – Adds the given caption datatype to the array of captions
- push(String name, String text) – finds the caption or speech datatype name and sets its text component equal to text.
- addControl(String id, int coor1, int coor2, int coor3, int coor4, int color, int type) – Adds code to control the page-flow of the book.
- isDuplicate(String name) – Checks if the declared identifier name has already been used.
- isBackground(String name) – Checks if the identifier name is that of a valid declared background datatype.
- isItem(String name) – Checks if the identifier name is that of a valid declared item datatype.
- isSpeech(String name) – Checks if the identifier name is that of a valid declared speech datatype.
- isCaption(String name) – Checks if the identifier name is that of a valid declared caption datatype.

19.3 DyogBackground.java:

DyogBackground.java holds the necessary information for a background datatype and the necessary functions to access that information.

Interfacing Methods:

- getname() – returns the identifier for that background.
- getfile() – returns the file associated with that background.

19.4 DyogItem.java:

DyogItem.java holds the necessary information for an item datatype and the necessary functions to access that information.

Interfacing Methods:

- getname() – returns the identifier for that item.
- getfile() – returns the file associated with that item.

19.5 DyogSpeech.java:

DyogSpeech.java holds the necessary information for a speech datatype and the necessary functions to access that information.

Interfacing Methods:

- getname() – returns the identifier for that speech.
- gettext() – returns the text associated with that speech.

19.6 DyogCaption.java:

DyogCaption.java holds the necessary information for a caption datatype and the necessary functions to access that information.

Interfacing Methods:

- getname() – returns the identifier for that caption.
- gettext() – returns the text associated with that caption.

Test Plan

20. Test Plan:

The Lexer and Parser were the first to be designed so they were the first tested. These tests took place before many of the specifics of the language were finalized. The test simply consisted of creating sample source language code and feeding that code into the Lexer and Parser to make sure the tokens were separated properly and that the abstract syntax tree was generated in the appropriate fashion.

Next the intermediate version of the Tree Walker was made. This version of the Tree Walker simply grabbed all of the values of the AST and outputted them to the command prompt. This functionality was tested with the same sample code as the Lexer and Parser.

Next classes to hold the datatypes were designed. These classes included: DyogBackground.java, DyogItem.java, DyogCaption.java, and DyogSpeech.java. They were tested by making a mock java program that instantiated these classes and called each of their public methods to ensure correct functionality.

Once it was sure that the datatypes worked the Tree Walker was finalized in parallel with the design of DyogDataOut.java. I went through each possible node in the tree and decided what I needed to check to ensure correct semantics and what data I needed to send DyogDataOut.java inorder to output the correct code. Once these two were finalized it was time to make a test suite of source language programs to ensure the compiler was fully functional. For this I created 3 programs. The first told the Jack and Jill story, the second told an original story about alien attacks the third was a program that contained semantic errors that was used to test the static semantic checking of my compiler. The two working programs tested each command and operator and used multiple objects of each datatype. The code with the errors showed examples of all possible semantic problems that can occur. This included using operators on illegal types, invoking commands with illegal types, and sending commands invalid parameters. Here are the results of each:

20.1 Jack and Jill

Source Code: File name→ test_jack_and_jill.txt

```
//Declare necessary variables
background hill [hill.jpg];
item jack [jack.jpg];
item jack_fall1 [jack_h.jpg];
item jack_fall2 [jack_u.jpg];
item jill [jill.jpg];
item jill_fall [jill_h.jpg];
speech jack_talk "";
speech jill_talk "";
caption story "";

//page 1
set hill;
position jack 600 450;
position jill 575 400;
jack_talk <- "Lets go get water";
jill_talk <- "I'm thirsty";
story <- "Jack and Jill went up the hill to fetch a pail of water";
spchloc jack_talk 600 300 600 450 BLACK;
spchloc jill_talk 550 280 575 400 BLACK;
caploc story 100 510 BLACK;
show;

//page 2
set hill;
position jack_fall1 450 300;
position jill 400 170;
jack_talk <- "Oh no!";
jill_talk <- "Its slippery up here";
story <- "Jack fell down";
spchloc jack_talk 500 250 490 300 BLACK;
spchloc jill_talk 400 150 400 200 BLACK;
caploc  story 100 510 BLACK;
show;

//page 3
set hill;
position jack_fall2 600 450;
position jill 400 170;
jack_talk <- "Ouch";
jill_talk <- "Are you ok?";
```

```
story <- "Broke his crown";
spchloc jack_talk 630 400 600 450 BLACK;
spchloc jill_talk 400 150 400 200 BLACK;
caploc story 100 510 BLACK;
show;

//page 4
set hill;
position jack_fall2 600 450;
position jill_fall 450 300;
jack_talk <- "I need an asprin";
jill_talk <- "This is going to hurt";
story <- "And Jill came tumbling after";
spchloc jack_talk 580 400 610 450 BLACK;
spchloc jill_talk 500 250 510 300 BLACK;
caploc story 100 510 BLACK;
end;
```

Target Language Code: File name → DyogStory.java

```
import java.awt.Container;
import java.awt.FlowLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JFrame;
import javax.swing.JButton;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;

public class DyogStory extends JFrame{
public static void main (String [] args) {
new DyogStory();
}
public DyogStory(){
setLocation (0,0);
setSize (800, 600);
setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
Container dyog_cont = getContentPane();
JPanel dyog_controls = new JPanel();
dyog_controls.setLayout (new GridLayout(5,1));
dyog_controls.setSize(150,250);
JButton button1 = new JButton ("Next");
dyog_controls.add (button1);
```

```
JButton button2 = new JButton ("Previous");
dyog_controls.add (button2);
MyScreen screen = new MyScreen(button1, button2);
dyog_cont.add (screen, BorderLayout.CENTER);
dyog_cont.add (dyog_controls, BorderLayout.EAST);
show();
}
}

class MyScreen extends JComponent implements ActionListener{
int frame;
boolean theend;
JButton b1,b2;

//list of background variables
ImageIcon hill;
boolean draw_hill;

//list of Item variables
ImageIcon jack;
boolean draw_jack;
int jackx, jacky;
ImageIcon jack_fall1;
boolean draw_jack_fall1;
int jack_fall1x, jack_fall1y;
ImageIcon jack_fall2;
boolean draw_jack_fall2;
int jack_fall2x, jack_fall2y;
ImageIcon jill;
boolean draw_jill;
int jillx, jilly;
ImageIcon jill_fall;
boolean draw_jill_fall;
int jill_fallx, jill_fally;

//list of caption variables
boolean draw_story;
int storyx, storyy;
int story_color;
String story_text;

//list of speech variables
boolean draw_jack_talk;
int jack_talkx1, jack_talky1, jack_talkx2, jack_talky2;
int jack_talk_color;
String jack_talk_text;
```

```
boolean draw_jill_talk;
int jill_talkx1, jill_talky1, jill_talkx2, jill_talky2;
int jill_talk_color;
String jill_talk_text;

//Constructor
MyScreen(JButton b1,JButton b2){
this.b1 = b1;
this.b2 = b2;
frame = 0;
theend = false;
this.b1.addActionListener(this);
this.b2.addActionListener(this);

//set background variables
hill = new ImageIcon("hill.jpg");
draw_hill = false;

//set Item variables
jack = new ImageIcon("jack.jpg");
draw_jack = false;
jack_fall1 = new ImageIcon("jack_h.jpg");
draw_jack_fall1 = false;
jack_fall2 = new ImageIcon("jack_u.jpg");
draw_jack_fall2 = false;
jill = new ImageIcon("jill.jpg");
draw_jill = false;
jill_fall = new ImageIcon("jill_h.jpg");
draw_jill_fall = false;

//set caption variables
draw_story = false;
story_color = 1;  //Defalut to BLACK
story_text = "And Jill came tumbling after";

//set speech variables
draw_jack_talk = false;
jack_talk_color = 1; //Default to BlACK
jack_talk_text = "I need an asprin";
draw_jill_talk = false;
jill_talk_color = 1; //Default to BlACK
jill_talk_text = "This is going to hurt";

this.Control();
}
```

```
public void paintComponent (Graphics g) {
Graphics2D g2 = (Graphics2D) g;
g2.setColor(Color.white);
g.fillRect(0,500,700,100);
g2.setColor(Color.black);
if(theend == true){g.drawString ("THE END",100,200);}

//draw backgrounds
if(draw_hill == true){hill.paintIcon(this,g,0,0);}
draw_hill = false;

//draw itmes
if(draw_jack == true){jack.paintIcon(this,g,jackx,jacky);}
draw_jack = false;
if(draw_jack_fall1 == true){jack_fall1.paintIcon(this,g,jack_fall1x,jack_fall1y);}
draw_jack_fall1 = false;
if(draw_jack_fall2 == true){jack_fall2.paintIcon(this,g,jack_fall2x,jack_fall2y);}
draw_jack_fall2 = false;
if(draw_jill == true){jill.paintIcon(this,g,jillx,jilly);}
draw_jill = false;
if(draw_jill_fall == true){jill_fall.paintIcon(this,g,jill_fallx,jill_fally);}
draw_jill_fall = false;

//draw captions
if(draw_story == true){
if(story_color == 2){g2.setColor(Color.white);}
g.drawString(story_text,storyx,storyy);
}
draw_story = false;
story_color = 1;
g2.setColor(Color.black);

//list of speech variables
if(draw_jack_talk == true){
if(jack_talk_color == 2){g2.setColor(Color.white);}
g.drawString(jack_talk_text,jack_talkx1,jack_talky1);
g.drawLine(jack_talkx1,jack_talky1,jack_talkx2,jack_talky2);
}
draw_jack_talk = false;
jack_talk_color = 1;
g2.setColor(Color.black);
if(draw_jill_talk == true){
if(jill_talk_color == 2){g2.setColor(Color.white);}
g.drawString(jill_talk_text,jill_talkx1,jill_talky1);
g.drawLine(jill_talkx1,jill_talky1,jill_talkx2,jill_talky2);
}
```

```
draw_jill_talk = false;
jill_talk_color = 1;
g2.setColor(Color.black);

Control();
}

//Action Preformed function
public void actionPerformed(ActionEvent e){
if(e.getSource() == b1){frame++;}
if(e.getSource() == b2){frame--;}
}

public void Control(){
if(frame == 0){
draw_hill = true;
draw_jack = true;
jackx = 600;
jacky = 450;
draw_jill = true;
jillx = 575;
jilly = 400;
draw_jack_talk = true;
jack_talkx1 = 600;
jack_talky1 = 300;
jack_talkx2 = 600;
jack_talky2 = 450;
jack_talk_color = 1;
jack_talk_text = "Lets go get water";
draw_jill_talk = true;
jill_talkx1 = 550;
jill_talky1 = 280;
jill_talkx2 = 575;
jill_talky2 = 400;
jill_talk_color = 1;
jill_talk_text = "I'm thirsty";
draw_story = true;
storyx = 100;
storyy = 510;
story_color = 1;
story_text = "Jack and Jill went up the hill to fetch a pail of water";
repaint();
theend = false;
}
if(frame == 1){
draw_hill = true;
```

```
draw_jack_fall1 = true;
jack_fall1x = 450;
jack_fall1y = 300;
draw_jill = true;
jillx = 400;
jilly = 170;
draw_jack_talk = true;
jack_talkx1 = 500;
jack_talky1 = 250;
jack_talkx2 = 490;
jack_talky2 = 300;
jack_talk_color = 1;
jack_talk_text = "Oh no!";
draw_jill_talk = true;
jill_talkx1 = 400;
jill_talky1 = 150;
jill_talkx2 = 400;
jill_talky2 = 200;
jill_talk_color = 1;
jill_talk_text = "Its slippery up here";
draw_story = true;
storyx = 100;
storyy = 510;
story_color = 1;
story_text = "Jack fell down";
repaint();
theend = false;
}
if(frame == 2){
draw_hill = true;
draw_jack_fall2 = true;
jack_fall2x = 600;
jack_fall2y = 450;
draw_jill = true;
jillx = 400;
jilly = 170;
draw_jack_talk = true;
jack_talkx1 = 630;
jack_talky1 = 400;
jack_talkx2 = 600;
jack_talky2 = 450;
jack_talk_color = 1;
jack_talk_text = "Ouch";
draw_jill_talk = true;
jill_talkx1 = 400;
jill_talky1 = 150;
```

```
jill_talkx2 = 400;
jill_talky2 = 200;
jill_talk_color = 1;
jill_talk_text = "Are you ok?";
draw_story = true;
storyx = 100;
storyy = 510;
story_color = 1;
story_text = "Broke his crown";
repaint();
theend = false;
}
if(frame == 3){
draw_hill = true;
draw_jack_fall2 = true;
jack_fall2x = 600;
jack_fall2y = 450;
draw_jill_fall = true;
jill_fallx = 450;
jill_fally = 300;
draw_jack_talk = true;
jack_talkx1 = 580;
jack_talky1 = 400;
jack_talkx2 = 610;
jack_talky2 = 450;
jack_talk_color = 1;
jack_talk_text = "I need an asprin";
draw_jill_talk = true;
jill_talkx1 = 500;
jill_talky1 = 250;
jill_talkx2 = 510;
jill_talky2 = 300;
jill_talk_color = 1;
jill_talk_text = "This is going to hurt";
draw_story = true;
storyx = 100;
storyy = 510;
story_color = 1;
story_text = "And Jill came tumbling after";
theend = true;
repaint();
}
if(frame > 3){
frame = 4;
repaint();
}
```

```
if(frame < 0){
frame = 0;
repaint();
}
}

}
```

20.2 Alien Story

Souce Code: File name → test_aliens.txt

```
background day [cityday.jpg];
background night [citynight.jpg];
item alien1      [monster.jpg];
item alien2      [alien1.jpg];
item alien3      [alien3.jpg];
item hero        [hero2.jpg];
item boom        [boom.jpg];
speech hero_talk "";
caption story "";
caption laugh1 "HA HA HA!!";
caption laugh2 "hahahahaha";
speech scream1 "SCREAMS!!!";
speech scream2 "More SCREAMS!!";

//page 1
set day;
story <- "There once lived a peaceful city";
caploc story 100 510 BLACK;
show;

//page 2
set day;
position alien1 300 400;
position alien2 400 450;
position alien3 500 450;
story <- "One day three aliens walked into town with plans to destroy it";
caploc story 100 510 BLACK;
show;

//page 3
set day;
position alien1 340 210;
position alien2 475 260;
position alien3 600 210;
story <- "The aliens swarmed the city and scared all of the residents";
caploc story 100 510 BLACK;
spchloc scream1 450 100 425 250 BLACK;
spchloc scream2 575 50        525 200 BLACK;
show;

//page 4
set day;
```

```
position alien1 340 210;
position alien2 475 260;
position alien3 600 210;
position hero 50 430;
story <- "All hope was lost...Until a hero walked into town";
hero_talk <- "Get down from those buildings!";
caploc story 100 510 BLACK;
spchloc hero_talk 100 420 60 430 BLACK;
show;

//page 5
set day;
position alien1 300 400;
position alien2 400 450;
position alien3 500 450;
position hero 50 430;
story <- "The citizens watched from their apartments as the hero challenged the aliens";
hero_talk <- "I will teach you guys a lesson";
caploc story 100 510 BLACK;
spchloc hero_talk 100 420 60 430 BLACK;
show;

//page 6
set day;
position boom 310 410;
position alien2 400 450;
position alien3 500 450;
position hero 280 400;
story<- "The hero killed the first alien";
caploc story 100 510 BLACK;
show;

//page 7
set day;
position boom 400 450;
position alien3 500 450;
position hero 380 430;
story<- "Then the second alien";
caploc story 100 510 BLACK;
show;

//page 8
set day;
position boom 500 450;
position hero 480 430;
story<- "Then the hero finished off the third alien";
```

```
caploc story 100 510 BLACK;
show;

//page 9
set day;
position hero 600 440;
hero_talk <- "Someone else is in trouble.  I must go!";
story <- "The day was saved and the hero left to fight more crime elsewere.";
caploc story 100 510 BLACK;
spchloc hero_talk 480 420 610 440 BLACK;
show;

//page 10
set night;
story <- "The people of the city were able to sleep safe once again";
caploc story 100 510 BLACK;
show;

end;
```

Target Language Code: File name → DyogStory.java

```
import java.awt.Container;
import java.awt.FlowLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JFrame;
import javax.swing.JButton;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;

public class DyogStory extends JFrame{
public static void main (String [] args) {
new DyogStory();
}
public DyogStory(){
setLocation (0,0);
setSize (800, 600);
setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
Container dyog_cont = getContentPane();
JPanel dyog_controls = new JPanel();
dyog_controls.setLayout (new GridLayout(5,1));
dyog_controls.setSize(150,250);
```

```
JButton button1 = new JButton ("Next");
dyog_controls.add (button1);
JButton button2 = new JButton ("Previous");
dyog_controls.add (button2);
MyScreen screen = new MyScreen(button1, button2);
dyog_cont.add (screen, BorderLayout.CENTER);
dyog_cont.add (dyog_controls, BorderLayout.EAST);
show();
}
}

class MyScreen extends JComponent implements ActionListener{
int frame;
boolean theend;
JButton b1,b2;

//list of background variables
ImageIcon day;
boolean draw_day;
ImageIcon night;
boolean draw_night;

//list of Item variables
ImageIcon alien1;
boolean draw_alien1;
int alien1x, alien1y;
ImageIcon alien2;
boolean draw_alien2;
int alien2x, alien2y;
ImageIcon alien3;
boolean draw_alien3;
int alien3x, alien3y;
ImageIcon hero;
boolean draw_hero;
int herox, heroy;
ImageIcon boom;
boolean draw_boom;
int boomx, boomy;

//list of caption variables
boolean draw_story;
int storyx, storyy;
int story_color;
String story_text;
boolean draw_laugh1;
int laugh1x, laugh1y;
```

```
int laugh1_color;
String laugh1_text;
boolean draw_laugh2;
int laugh2x, laugh2y;
int laugh2_color;
String laugh2_text;

//list of speech variables
boolean draw_hero_talk;
int hero_talkx1, hero_talky1, hero_talkx2, hero_talky2;
int hero_talk_color;
String hero_talk_text;
boolean draw_scream1;
int scream1x1, scream1y1, scream1x2, scream1y2;
int scream1_color;
String scream1_text;
boolean draw_scream2;
int scream2x1, scream2y1, scream2x2, scream2y2;
int scream2_color;
String scream2_text;

//Constructor
MyScreen(JButton b1,JButton b2){
this.b1 = b1;
this.b2 = b2;
frame = 0;
theend = false;
this.b1.addActionListener(this);
this.b2.addActionListener(this);

//set background variables
day = new ImageIcon("cityday.jpg");
draw_day = false;
night = new ImageIcon("citynight.jpg");
draw_night = false;

//set Item variables
alien1 = new ImageIcon("monster.jpg");
draw_alien1 = false;
alien2 = new ImageIcon("alien1.jpg");
draw_alien2 = false;
alien3 = new ImageIcon("alien3.jpg");
draw_alien3 = false;
hero = new ImageIcon("hero2.jpg");
draw_hero = false;
boom = new ImageIcon("boom.jpg");
```

```
draw_boom = false;

//set caption variables
draw_story = false;
story_color = 1;  //Defalut to BLACK
story_text = "The people of the city were able to sleep safe once again";
draw_laugh1 = false;
laugh1_color = 1;  //Defalut to BLACK
laugh1_text = "HA HA HA!!";
draw_laugh2 = false;
laugh2_color = 1;  //Defalut to BLACK
laugh2_text = "hahahahaha";

//set speech variables
draw_hero_talk = false;
hero_talk_color = 1; //Default to BlACK
hero_talk_text = "Someone else is in trouble.  I must go!";
draw_scream1 = false;
scream1_color = 1; //Default to BlACK
scream1_text = "SCREAMS!!!";
draw_scream2 = false;
scream2_color = 1; //Default to BlACK
scream2_text = "More SCREAMS!!";

this.Control();
}

public void paintComponent (Graphics g) {
Graphics2D g2 = (Graphics2D) g;
g2.setColor(Color.white);
g.fillRect(0,500,700,100);
g2.setColor(Color.black);
if(theend == true){g.drawString ("THE END",100,200);}

//draw backgrounds
if(draw_day == true){day.paintIcon(this,g,0,0);}
draw_day = false;
if(draw_night == true){night.paintIcon(this,g,0,0);}
draw_night = false;

//draw itmes
if(draw_alien1 == true){alien1.paintIcon(this,g,alien1x,alien1y);}
draw_alien1 = false;
if(draw_alien2 == true){alien2.paintIcon(this,g,alien2x,alien2y);}
draw_alien2 = false;
if(draw_alien3 == true){alien3.paintIcon(this,g,alien3x,alien3y);}
```

```
draw_alien3 = false;
if(draw_hero == true){hero.paintIcon(this,g,herox,heroy);}
draw_hero = false;
if(draw_boom == true){boom.paintIcon(this,g,boomx,boomy);}
draw_boom = false;

//draw captions
if(draw_story == true){
if(story_color == 2){g2.setColor(Color.white);}
g.drawString(story_text,storyx,storyy);
}
draw_story = false;
story_color = 1;
g2.setColor(Color.black);
if(draw_laugh1 == true){
if(laugh1_color == 2){g2.setColor(Color.white);}
g.drawString(laugh1_text,laugh1x,laugh1y);
}
draw_laugh1 = false;
laugh1_color = 1;
g2.setColor(Color.black);
if(draw_laugh2 == true){
if(laugh2_color == 2){g2.setColor(Color.white);}
g.drawString(laugh2_text,laugh2x,laugh2y);
}
draw_laugh2 = false;
laugh2_color = 1;
g2.setColor(Color.black);

//list of speech variables
if(draw_hero_talk == true){
if(hero_talk_color == 2){g2.setColor(Color.white);}
g.drawString(hero_talk_text,hero_talkx1,hero_talky1);
g.drawLine(hero_talkx1,hero_talky1,hero_talkx2,hero_talky2);
}
draw_hero_talk = false;
hero_talk_color = 1;
g2.setColor(Color.black);
if(draw_scream1 == true){
if(scream1_color == 2){g2.setColor(Color.white);}
g.drawString(scream1_text,scream1x1,scream1y1);
g.drawLine(scream1x1,scream1y1,scream1x2,scream1y2);
}
draw_scream1 = false;
scream1_color = 1;
g2.setColor(Color.black);
```

```
if(draw_scream2 == true){
if(scream2_color == 2){g2.setColor(Color.white);}
g.drawString(scream2_text,scream2x1,scream2y1);
g.drawLine(scream2x1,scream2y1,scream2x2,scream2y2);
}
draw_scream2 = false;
scream2_color = 1;
g2.setColor(Color.black);

Control();
}

//Action Preformed function
public void actionPerformed(ActionEvent e){
if(e.getSource() == b1){frame++;}
if(e.getSource() == b2){frame--;}
}

public void Control(){
if(frame == 0){
draw_day = true;
draw_story = true;
storyx = 100;
storyy = 510;
story_color = 1;
story_text = "There once lived a peaceful city";
repaint();
theend = false;
}
if(frame == 1){
draw_day = true;
draw_alien1 = true;
alien1x = 300;
alien1y = 400;
draw_alien2 = true;
alien2x = 400;
alien2y = 450;
draw_alien3 = true;
alien3x = 500;
alien3y = 450;
draw_story = true;
storyx = 100;
storyy = 510;
story_color = 1;
story_text = "One day three aliens walked into town with plans to destroy it";
repaint();
```

```
theend = false;
}
if(frame == 2){
draw_day = true;
draw_alien1 = true;
alien1x = 340;
alien1y = 210;
draw_alien2 = true;
alien2x = 475;
alien2y = 260;
draw_alien3 = true;
alien3x = 600;
alien3y = 210;
draw_story = true;
storyx = 100;
storyy = 510;
story_color = 1;
story_text = "The aliens swarmed the city and scared all of the residents";
draw_scream1 = true;
scream1x1 = 450;
scream1y1 = 100;
scream1x2 = 425;
scream1y2 = 250;
scream1_color = 1;
scream1_text = "SCREAMS!!!";
draw_scream2 = true;
scream2x1 = 575;
scream2y1 = 50;
scream2x2 = 525;
scream2y2 = 200;
scream2_color = 1;
scream2_text = "More SCREAMS!!";
repaint();
theend = false;
}
if(frame == 3){
draw_day = true;
draw_alien1 = true;
alien1x = 340;
alien1y = 210;
draw_alien2 = true;
alien2x = 475;
alien2y = 260;
draw_alien3 = true;
alien3x = 600;
alien3y = 210;
```

```
draw_hero = true;
herox = 50;
heroy = 430;
draw_story = true;
storyx = 100;
storyy = 510;
story_color = 1;
story_text = "All hope was lost...Until a hero walked into town";
draw_hero_talk = true;
hero_talkx1 = 100;
hero_talky1 = 420;
hero_talkx2 = 60;
hero_talky2 = 430;
hero_talk_color = 1;
hero_talk_text = "Get down from those buildings!";
repaint();
theend = false;
}
if(frame == 4){
draw_day = true;
draw_alien1 = true;
alien1x = 300;
alien1y = 400;
draw_alien2 = true;
alien2x = 400;
alien2y = 450;
draw_alien3 = true;
alien3x = 500;
alien3y = 450;
draw_hero = true;
herox = 50;
heroy = 430;
draw_story = true;
storyx = 100;
storyy = 510;
story_color = 1;
story_text = "The citizens watched from their apartments as the hero challenged the aliens";
draw_hero_talk = true;
hero_talkx1 = 100;
hero_talky1 = 420;
hero_talkx2 = 60;
hero_talky2 = 430;
hero_talk_color = 1;
hero_talk_text = "I will teach you guys a lesson";
repaint();
theend = false;
```

```
}
if(frame == 5){
draw_day = true;
draw_boom = true;
boomx = 310;
boomy = 410;
draw_alien2 = true;
alien2x = 400;
alien2y = 450;
draw_alien3 = true;
alien3x = 500;
alien3y = 450;
draw_hero = true;
herox = 280;
heroy = 400;
draw_story = true;
storyx = 100;
storyy = 510;
story_color = 1;
story_text = "The hero killed the first alien";
repaint();
theend = false;
}
if(frame == 6){
draw_day = true;
draw_boom = true;
boomx = 400;
boomy = 450;
draw_alien3 = true;
alien3x = 500;
alien3y = 450;
draw_hero = true;
herox = 380;
heroy = 430;
draw_story = true;
storyx = 100;
storyy = 510;
story_color = 1;
story_text = "Then the second alien";
repaint();
theend = false;
}
if(frame == 7){
draw_day = true;
draw_boom = true;
boomx = 500;
```

```
boomy = 450;
draw_hero = true;
herox = 480;
heroy = 430;
draw_story = true;
storyx = 100;
storyy = 510;
story_color = 1;
story_text = "Then the hero finished off the third alien";
repaint();
theend = false;
}
if(frame == 8){
draw_day = true;
draw_hero = true;
herox = 600;
heroy = 440;
draw_story = true;
storyx = 100;
storyy = 510;
story_color = 1;
story_text = "The day was saved and the hero left to fight more crime elsewere.";
draw_hero_talk = true;
hero_talkx1 = 480;
hero_talky1 = 420;
hero_talkx2 = 610;
hero_talky2 = 440;
hero_talk_color = 1;
hero_talk_text = "Someone else is in trouble.  I must go!";
repaint();
theend = false;
}
if(frame == 9){
draw_night = true;
draw_story = true;
storyx = 100;
storyy = 510;
story_color = 1;
story_text = "The people of the city were able to sleep safe once again";
repaint();
theend = false;
}
if(frame == 10){
theend = true;
repaint();
}
```

```
if(frame > 10){
frame = 11;
repaint();
}
if(frame < 0){
frame = 0;
repaint();
}
}

}
```

20.3 Bad Semantics

Source Code: File name → badsemantics.txt

```
background jungle [jungle.jpg];
background jungle [milk.jpg];
item cat [cat.jpg];
item dog [dog.jpg];
item bird [bird.jpg];
item dog [mouse.jpg];
speech talk1 "hello";
speech talk2 "bye";
caption cap1 "they greet";
caption cap2 "they say farwell";
speech talk1 "bad speech";
caption cap2 "back cap";
cat <- "new text";
jungle <- "new text";
set cat;
position jungle 1 2;
spchloc dog 3 4 5 6 PURPLE;
caploc talk1 7 8 dog;
position cap2 2 4;
set jungle;
position manny 67 89;
```

Command Line Output: (Note – When errors are found in the program there is no code written to the output file)

```
C:\jdk1.3.1_01\bin>
C:\jdk1.3.1_01\bin>
C:\jdk1.3.1_01\bin>java Main < badsemantics.txt
line 22:1: expecting "end", found 'null'
 ( PRGM ( background jungle jungle.jpg ) ( background jungle milk.jpg ) ( item c
at cat.jpg ) ( item dog dog.jpg ) ( item bird bird.jpg ) ( item dog mouse.jpg )
( speech talk1 hello ) ( speech talk2 bye ) ( caption cap1 they greet ) ( captio
n cap2 they say farwell ) ( speech talk1 bad speech ) ( caption cap2 back cap )
( <- cat new text ) ( <- jungle new text ) ( set cat ) ( position jungle 1 2 ) (
 spchloc dog 3 4 5 6 PURPLE ) ( caploc talk1 7 8 dog ) ( position cap2 2 4 ) ( s
et jungle ) ( position manny 67 89 ) )
Error: identifier: jungle, was already declared

Error: identifier: dog, was already declared

Error: identifier: talk1, was already declared
```

Error: identifier: cap2, was already declared

Error: Identifier: cat is not a valid "speech" or "caption" datatype
'<-' operator only operates on a "speech" or "caption" datatype

Error: Identifier: jungle is not a valid "speech" or "caption" datatype
'<-' operator only operates on a "speech" or "caption" datatype

Error: Identifier: cat is not a valid "background" datatype
the "set" operator only operates on "background" datatypes

Error: Identifier: jungle is not a valid "item" datatype
The "pos" operator only operates on "item" datatypes

WARNING: Did not specify a correct color in "spchloc";
Color will default to BLACK

Error: Identifier: dog is not a valid "speech" datatype
the "spchloc" operator only operates on "speech" datatypes

WARNING: Did not specify a correct color in "caploc";
Color will default to BLACK

Error: Identifier: talk1 is not a valid "caption" datatype
The "caploc" operator only operates on "caption" datatypes

Error: Identifier: cap2 is not a valid "item" datatype
The "pos" operator only operates on "item" datatypes

Error: Identifier: manny is not a valid "item" datatype
The "pos" operator only operates on "item" datatypes

<AST>:0:0: expecting "end", found '<ASTNULL>'
Error: "end" statement is missing

Exiting due to errors

C:\jdk1.3.1_01\bin>

Lessons Learned

21. Lessons Learned:

Designing a language is a very involving task. There are many different pieces that have to work together and this requires a lot of organization. The most important lesson learned was that there are many curve balls that get thrown into the design of the project. There were many occasions were I thought everything was going smoothly then later I realized that a small modification had to be made that ended up causing a large ripple effect of other modifications. The best advice I can give is to always have an overall vision and organizational structure of what the project is going to be. You need to know what software components are necessary and how they will work with one another. Another bit of advice is to never underestimate the time required to do such a project because, like stated earlier, there are many curve balls that will throw the project off course and the curve balls are usually impossible to foresee and it is unfeasible to predict the amount of time needed to get back on course.

Appendix 1: Front End Source Code

File Name: Dyog.g

```
/*Dyog Lexer*/
/*Author: Craig D. Vargas*/
/*October 13, 2004*/

class DyogLexer extends Lexer;
options{testLiterals = false; k=2; charVocabulary = '\3'..'\377'; exportVocab = Dyog;}

SEMI  :         ';';
PUSH  :         '<' '-';

protected LETTER:    ('a'..'z' | 'A'..'Z');
protected DIGIT:     ('0'..'9');
protected UNDER:     '_';
protected PERIOD:    '.';


ID options { testLiterals = true;}       :          LETTER (LETTER | DIGIT | UNDER)*;

NUMBER      :         (DIGIT)+;
FILE   :         '['! (LETTER|DIGIT|UNDER) (LETTER|DIGIT|UNDER)* PERIOD LETTER
LETTER LETTER ']'!;
STRING         :         ""! ('"' ""! | ~('"'))* ""!;
WS     :         (' ' | '\t' | '\n' {newline();} | '\r') {$setType(Token.SKIP);};
COMMENT   :         "//"(~('\r'|'\n'))* { $setType(Token.SKIP); };

/*-------------------------------------------------------------------------------------------------*/

/*Dyog Parser*/
/*Author: Craig D. Vargas*/
/*October 15, 2004*/

class DyogParser extends Parser;
options {buildAST = true; k = 2; exportVocab = Dyog;}

tokens{
  PRGM;
}

prgm: (bkgd | itm | spch | cptn | push | set | pos | spchloc | caploc | show)* end EOF!
  {#prgm = #([PRGM,"PRGM"],prgm);}
;
```

```
bkgd : "background"^ ID FILE SEMI!;

itm : "item"^ ID FILE SEMI!;

spch : "speech"^ ID STRING SEMI!;

/*spch_tail : SEMI!
            | STRING SEMI!;*/

cptn : "caption"^ ID STRING SEMI!;

/*cpt_tail : SEMI!
           | STRING SEMI!;*/

push : ID PUSH^ STRING SEMI!;

set : "set"^ ID SEMI!;

pos : "position"^ ID NUMBER NUMBER SEMI!;

spchloc : "spchloc"^ ID NUMBER NUMBER NUMBER NUMBER ID SEMI!;

caploc:  "caploc"^ ID NUMBER NUMBER ID SEMI!;

show : "show"^ SEMI!;

end: "end"^ SEMI!;

/*-------------------------------------------------------------------------------------*/

/*Dyog TreeWalker*/

class DyogWalker extends TreeParser;

options{importVocab = Dyog; k=2;}

{DyogDataOut ddo = new DyogDataOut(); boolean error = false; boolean end = false;}

prgm {
  int coor1,coor2,coor3,coor4;
  String name, file, text;
  //System.out.println("Getting started\n");
}
  : #(PRGM (bkgd|itm|spch|cptn|push|set|pos|spchloc|caploc|show)* end)
  {
    //System.out.println("Here goes the final out calls\n");
```

```
    //ddo.outBackgrounds();
    //ddo.outItems();
    //ddo.outSpeechs();
    //ddo.outCaptions();
    if(!end){System.out.println("Error: \"end\" statement is missing\n\n"); error = true;}
    if(!error){ddo.writeFile("DyogStory.java");}
    else{System.out.println("Exiting due to errors\n");}
  }
  ;

bkgd {
  String name,file;
}
  : #("background" ID FILE)
  {
    name = #ID.getText();
    file = #FILE.getText();
    DyogBackground b = new DyogBackground(name,file);
    if(!ddo.isDuplicate(name)){ddo.addBackground(b);}
    else{System.out.println("Error: identifier: " + name + ", was already declared\n"); error = true;}
    //System.out.println("bakground's name: " + name + "  background's file: " + file + '\n');
  }
  ;
itm {
  String name,file;
}
  : #("item" ID FILE)
  {
    name = #ID.getText();
    file = #FILE.getText();
    DyogItem i = new DyogItem(name,file);
    if(!ddo.isDuplicate(name)){ddo.addItem(i);}
    else{System.out.println("Error: identifier: " + name + ", was already declared\n"); error = true;}
    //System.out.println("item's name: " + name + "  item's file: " + file + '\n');
  }
  ;
spch{
  String name,text;
}
  : #("speech" ID STRING)
  {
    name = #ID.getText();
    text = #STRING.getText();
    DyogSpeech s = new DyogSpeech(name,text);
    if(!ddo.isDuplicate(name)){ddo.addSpeech(s);}
    else{System.out.println("Error: identifier: " + name + ", was already declared\n"); error = true;}
```

```
    //System.out.println("speech's name: " + name + "  speech's text: " + text + '\n');
  }
  ;
cptn{
  String name,text;
}
  : #("caption" ID STRING)
  {
    name = #ID.getText();
    text = #STRING.getText();
    DyogCaption c = new DyogCaption(name,text);
    if(!ddo.isDuplicate(name)){ddo.addCaption(c);}
    else{System.out.println("Error: identifier: " + name + ", was already declared\n"); error = true;}
    //System.out.println("caption's name: " + name + "  caption's text: " + text + '\n');
  }
  ;
push{
  String name,text;
}
  : #(PUSH ID STRING)
  {
    name = #ID.getText();
    text = #STRING.getText();
    if(ddo.isSpeech(name) || ddo.isCaption(name)){ddo.push(name,text);}
    else{System.out.println("Error: Identifier: " + name + " is not a valid \"speech\" or \"caption\"
datatype\n\'<-\' operator only operates on a \"speech\" or \"caption\" datatype\n\n"); error = true;}
    //System.out.println("PUSH's ID: " + name + "  PUSH's text: " + text + '\n');
  }
  ;
set{
  String name;
}
  : #("set" ID)
  {
    name = #ID.getText();
    //System.out.println("background to set: " + name + '\n');
    if(ddo.isBackground(name)){ddo.addControl(name,0,0,0,0,1,1);}
    else{System.out.println("Error: Identifier: " + name + " is not a valid \"background\" datatype\n
the \"set\" operator only operates on \"background\" datatypes\n\n");  error = true;}
  }
  ;
pos{
  String name;
  int coor1,coor2;
}
  : #("position" ID p1:NUMBER p2:NUMBER)
```

```
  {
    name = #ID.getText();
    coor1 = Integer.parseInt(#p1.getText());
    coor2 = Integer.parseInt(#p2.getText());
    //System.out.println("item to place: " + name + "  at coordinates: " + coor1 + " " + coor2 + '\n');
    if(ddo.isItem(name)){ddo.addControl(name,coor1,coor2,0,0,1,2);}
    else{System.out.println("Error: Identifier: " + name + " is not a valid \"item\" datatype\nThe
\"pos\" operator only operates on \"item\" datatypes\n\n");  error = true;}
  }
  ;
spchloc{
  String name, color;
  int coor1,coor2,coor3,coor4,text_color;
}
  : #("spchloc" n:ID s1:NUMBER s2:NUMBER s3:NUMBER s4:NUMBER wb:ID)
  {
    name = #n.getText();
    coor1 = Integer.parseInt(#s1.getText());
    coor2 = Integer.parseInt(#s2.getText());
    coor3 = Integer.parseInt(#s3.getText());
    coor4 = Integer.parseInt(#s4.getText());
    color = #wb.getText();
    if(color.equals("WHITE")){text_color = 2;}
    else{text_color = 1;if(!color.equals("BLACK")){System.out.println("WARNING: Did not
specify a correct color in \"spchloc\";\nColor will default to BLACK\n\n");}}
    //System.out.println("color = " + color + '\n');
    //System.out.println("speech to place: " + name + "  at coordinates: " + coor1 + " " + coor2 + " "
+ coor3 + " " + coor4 + '\n');
    if(ddo.isSpeech(name)){ddo.addControl(name,coor1,coor2,coor3,coor4,text_color,3);}
    else{System.out.println("Error: Identifier: " + name + " is not a valid \"speech\" datatype\n the
\"spchloc\" operator only operates on \"speech\" datatypes\n\n");  error = true;}
  }
  ;
caploc{
  String name, color;
  int coor1,coor2,text_color;
}
  : #("caploc" n:ID c1:NUMBER c2:NUMBER wb:ID)
  {
    name = #n.getText();
    coor1 = Integer.parseInt(#c1.getText());
    coor2 = Integer.parseInt(#c2.getText());
    color = #wb.getText();
    if(color.equals("WHITE")){text_color = 2;}
    else{text_color = 1;if(!color.equals("BLACK")){System.out.println("WARNING: Did not
specify a correct color in \"caploc\";\nColor will default to BLACK\n\n");}}
```

```
    //System.out.println("color = " + color + '\n');
    //System.out.println("caption to place: " + name + "  at coordinates: " + coor1 + " " + coor2 +
'\n');
    if(ddo.isCaption(name)){ddo.addControl(name,coor1,coor2,0,0,text_color,4);}
    else{System.out.println("Error: Identifier: " + name + " is not a valid \"caption\" datatype\nThe
\"caploc\" operator only operates on \"caption\" datatypes\n\n");  error = true;}

  }
  ;
show
  : ("show")
  {
    //System.out.println("Its time to show the screen\n");
    ddo.addControl("dummy",0,0,0,0,1,5);
  }
  ;
end
  : ("end")
  {
    //System.out.println("the program is over\n");
    if(!end){end = true;}
    else{System.out.println("Error: Duplicate end statements\n\n"); error = true;}
    ddo.addControl("dummy",0,0,0,0,1,6);
  }
;
```

Apendix 2: Back End Source Code

File Name: DyogBackground.java

```
public class DyogBackground{

  private String name = "";
  private String file = "";

  public DyogBackground(String n, String f){
    name = n;
    file = f;
  }

  public void setname(String n){
    name = n;
  }

  public void setfile(String f){
    file = f;
  }

  public String getname(){
    return name;
  }

  public String getfile(){
    return file;
  }
}
```

File Name: DyogItem.java

```
public class DyogItem{

  private String name = "";
  private String file = "";

  public DyogItem(String n, String f){
    name = n;
    file = f;
  }

  public void setname(String n){
    name = n;
```

```
  }

  public void setfile(String f){
    file = f;
  }

  public String getname(){
    return name;
  }

  public String getfile(){
    return file;
  }
}
```

File Name: DyogCaption.java

```
public class DyogCaption{

  private String name = "";
  private String text = "";

  public DyogCaption(String n, String t){
    name = n;
    text = t;
  }

  public void setname(String n){
    name = n;
  }

  public void settext(String t){
    text = t;
  }

  public String getname(){
    return name;
  }

  public String gettext(){
    return text;
  }
}
```

File Name: DyogSpeech.java

```
public class DyogSpeech{

  private String name = "";
  private String text = "";

  public DyogSpeech(String n, String t){
    name = n;
    text = t;
  }

  public void setname(String n){
    name = n;
  }

  public void settext(String t){
    text = t;
  }

  public String getname(){
    return name;
  }

  public String gettext(){
    return text;
  }
}
```

File Name: DyogDataOut.java

```
import java.util.*;
import java.io.*;
import java.lang.Object.*;

public class DyogDataOut{

  private Vector backgrounds;        //Holds all variables of type background
  private Vector items;              //Holds all variables of type item
  private Vector captions;           //Holds all variables of type caption
  private Vector speechs;            //Holds all variables of type speech
  private String control;
  private int frame;

  public DyogDataOut(){
```

```
  backgrounds = new Vector();
  items = new Vector();
  captions = new Vector();
  speechs = new Vector();
  control = "public void Control(){\r\nif(frame == 0){\r\n";
  frame = 0;
}

//function to add a new background variable to the data structure
public void addBackground(DyogBackground b){
  backgrounds.addElement(b);
}

//function to add a new item variable to the data structure
public void addItem(DyogItem i){
  items.addElement(i);
}

//function to add a new caption variable to the data structure
public void addCaption(DyogCaption c){
  captions.addElement(c);
}

//function to add a new speech variable to the data structure
public void addSpeech(DyogSpeech s){
        speechs.addElement(s);
}

public void push(String name, String text){
        String temp;
        //System.out.println("preforming a push operation\n");
        //System.out.println("I am looking for " + name + '\n');
        for(int x=0; x<speechs.size();x++){
          DyogSpeech s =(DyogSpeech)speechs.elementAt(x);
          //System.out.println("Comparing " + s.getname() + " to " + name + '\n');
          temp = s.getname();
          //System.out.println("temp = " + temp + '\n');
          if(temp.equals(name)){
                  //System.out.println("found it. Name is " + s.getname() + '\n');
                  s.settext(text);
                  //System.out.println("new text is " + s.gettext() + '\n');
          }
        }
        for(int x=0; x<captions.size();x++){
          DyogCaption c =(DyogCaption)captions.elementAt(x);
          //System.out.println("Comparing " + c.getname() + " to " + name + '\n');
```

```
            temp = c.getname();
            //System.out.println("temp = " + temp + '\n');
            if(temp.equals(name)){
                    //System.out.println("found it. Name is " + c.getname() + '\n');
                    c.settext(text);
                    //System.out.println("new text is " + c.gettext() + '\n');
            }
           }
}



//Function to output all backgrounds
public String outMyScreenClassBackgrounds(){
          String out = "//list of background variables\r\n";
          for(int x=0; x<backgrounds.size();x++){
            DyogBackground b =(DyogBackground)backgrounds.elementAt(x);
            //System.out.println(b.getname() + " " + b.getfile() + '\n');
            out += "ImageIcon " + b.getname() + ";\r\n";
            out += "boolean draw_" + b.getname() + ";\r\n";
  }
  return out;
}

//Function to output all Items
public String outMyScreenClassItems(){
          String out = "//list of Item variables\r\n";
          for(int x=0; x<items.size();x++){
            DyogItem i =(DyogItem)items.elementAt(x);
            //System.out.println(i.getname() + " " + i.getfile() + '\n');
            out += "ImageIcon " + i.getname() + ";\r\n";
            out += "boolean draw_" + i.getname() + ";\r\n";
            out += "int " + i.getname() + "x, " + i.getname() + "y;\r\n";
  }
  return out;
}

public String outMyScreenClassCaptions(){
  String out = "//list of caption variables\r\n";
  for(int x=0; x<captions.size();x++){
    DyogCaption c =(DyogCaption)captions.elementAt(x);
    //System.out.println(c.getname() + " " + c.gettext() + '\n');
    out += "boolean draw_" + c.getname() + ";\r\n";
    out += "int " + c.getname() + "x, " + c.getname() + "y;\r\n";
    out += "int " + c.getname() + "_color;\r\n";
    out += "String " + c.getname() + "_text;\r\n";
```

```
    }
    return out;
  }

    //Function to output all Speechs
  public String outMyScreenClassSpeechs(){
    String out = "//list of speech variables\r\n";
          for(int x=0; x<speechs.size();x++){
            DyogSpeech s =(DyogSpeech)speechs.elementAt(x);
      //System.out.println(s.getname() + " " + s.gettext() + '\n');
      out += "boolean draw_" + s.getname() + ";\r\n";
      out += "int " + s.getname() + "x1, " + s.getname() + "y1, " + s.getname() + "x2, " + s.getname()
+ "y2;\r\n";
      out += "int " + s.getname() + "_color;\r\n";
      out += "String " + s.getname() + "_text;\r\n";
    }
    return out;
  }


  public String outConstructorBackgrounds(){
          String out = "//set background variables\r\n";
          for(int x=0; x<backgrounds.size();x++){
            DyogBackground b =(DyogBackground)backgrounds.elementAt(x);
            //System.out.println(b.getname() + " " + b.getfile() + '\n');
            out += b.getname() + " = new ImageIcon(\"" + b.getfile() + "\");\r\n";
            out += "draw_" + b.getname() + " = false;\r\n";
          }
    return out;
  }

  //Function to output all Items
  public String outConstructorItems(){
          String out = "//set Item variables\r\n";
          for(int x=0; x<items.size();x++){
            DyogItem i =(DyogItem)items.elementAt(x);
            //System.out.println(i.getname() + " " + i.getfile() + '\n');
            out += i.getname() + " = new ImageIcon(\"" + i.getfile() + "\");\r\n";
            out += "draw_" + i.getname() + " = false;\r\n";
    }
    return out;
  }//Function to output all Captions

  public String outConstructorCaptions(){
          String out = "//set caption variables\r\n";
          for(int x=0; x<captions.size();x++){
```

```
            DyogCaption c =(DyogCaption)captions.elementAt(x);
            //System.out.println(c.getname() + " " + c.gettext() + '\n');
            out += "draw_" + c.getname() + " = false;\r\n";
            out += c.getname() + "_color = 1;  //Defalut to BLACK\r\n";
            out += c.getname() + "_text = \"" + c.gettext() + "\";\r\n";

        }
        return out;
    }

    //Function to output all Speechs
  public String outConstructorSpeechs(){
    String out = "//set speech variables\r\n";
          for(int x=0; x<speechs.size();x++){
            DyogSpeech s =(DyogSpeech)speechs.elementAt(x);
     //System.out.println(s.getname() + " " + s.gettext() + '\n');
     out += "draw_" + s.getname() + " = false;\r\n";
     out += s.getname() + "_color = 1; //Default to BlACK\r\n";
     out += s.getname() + "_text = \"" + s.gettext() + "\";\r\n";

    }
    return out;
  }

public String outConstructorHeader(){
          String out = "//Constructor\r\n";
          out += "MyScreen(JButton b1,JButton b2){\r\n";
          out += "this.b1 = b1;\r\n";
          out += "this.b2 = b2;\r\n";
          out += "frame = 0;\r\n";
          out += "theend = false;\r\n";
          out += "this.b1.addActionListener(this);\r\n";
          out += "this.b2.addActionListener(this);\r\n";
          //out += "this.Control();\r\n";
          return out;
  }

  public String outConstructorEnd(){
          String out = "this.Control();\r\n";
          out += "}\r\n";
          return out;
  }

  public String outMyScreenClassHeader(){
          String out = "class MyScreen extends JComponent implements ActionListener{\r\n";
          out += "int frame;\r\n";
```

```
        out += "boolean theend;\r\n";
        out += "JButton b1,b2;\r\n";
        return out;
  }

  public String outMyScreenClassEnd(){
        return "}\r\n";
  }

  public String outPaintComponentHeader(){
        String out = "public void paintComponent (Graphics g) {\r\n";
        out += "Graphics2D g2 = (Graphics2D) g;\r\n";
        out += "g2.setColor(Color.white);\r\n";
        out += "g.fillRect(0,500,700,100);\r\n";
        out += "g2.setColor(Color.black);\r\n";
        out += "if(theend == true){g.drawString (\"THE END\",100,200);}\r\n";
        return out;
  }

  public String outPaintComponentBackgrounds(){
    String out = "//draw backgrounds\r\n";
        for(int x=0; x<backgrounds.size();x++){
          DyogBackground b =(DyogBackground)backgrounds.elementAt(x);
          //System.out.println(b.getname() + " " + b.getfile() + '\n');
          out += "if(draw_" + b.getname() + " == true){" + b.getname() +
".paintIcon(this,g,0,0);}\r\n";
          out += "draw_" + b.getname() + " = false;\r\n";
        }
    return out;
  }

  public String outPaintComponentItems(){
        String out = "//draw itmes\r\n";
    for(int x=0; x<items.size();x++){
          DyogItem i =(DyogItem)items.elementAt(x);
          //System.out.println(i.getname() + " " + i.getfile() + '\n');
          out += "if(draw_" + i.getname() + " == true){" + i.getname() + ".paintIcon(this,g," +
i.getname() + "x," + i.getname() + "y);}\n\r";
          out += "draw_" + i.getname() + " = false;\r\n";}
        return out;
  }

  public String outPaintComponentCaptions(){
    String out = "//draw captions\r\n";
    for(int x=0; x<captions.size();x++){
      DyogCaption c =(DyogCaption)captions.elementAt(x);
```

```
    //System.out.println(c.getname() + " " + c.gettext() + '\n');
    out += "if(draw_" + c.getname() + " == true){\r\n";
    out += "if(" + c.getname() + "_color == 2){g2.setColor(Color.white);}\r\n";
    out += "g.drawString(" + c.getname() + "_text," + c.getname() + "x," + c.getname() + "y);\r\n";
    out += "}\r\n";
    out += "draw_" + c.getname() + " = false;\r\n";
    out += c.getname() + "_color = 1;\r\n";
    out += "g2.setColor(Color.black);\r\n";
   }
   return out;
  }

    //Function to output all Speechs
  public String outPaintComponentSpeechs(){
   String out = "//list of speech variables\r\n";
   for(int x=0; x<speechs.size();x++){
    DyogSpeech s =(DyogSpeech)speechs.elementAt(x);
    //System.out.println(s.getname() + " " + s.gettext() + '\n');
    out += "if(draw_" + s.getname() + " == true){\r\n";
    out += "if(" + s.getname() + "_color == 2){g2.setColor(Color.white);}\r\n";
    out += "g.drawString(" + s.getname() + "_text," + s.getname() + "x1," + s.getname() +
"y1);\r\n";
    out += "g.drawLine(" + s.getname() + "x1," + s.getname() + "y1," + s.getname() + "x2," +
s.getname() + "y2);\r\n";
    out += "}\r\n";
    out += "draw_" + s.getname() + " = false;\r\n";
    out += s.getname() + "_color = 1;\r\n";
    out += "g2.setColor(Color.black);\r\n";
   }
   return out;
  }

  public String outPaintComponentEnd(){
        String out = "Control();\r\n";
        out += "}\r\n";
        return out;
  }

  public String outActionPreformed(){
         String out ="//Action Preformed function\r\n";
         out += "public void actionPerformed(ActionEvent e){\r\n";
         out += "if(e.getSource() == b1){frame++;}\r\n";
         out += "if(e.getSource() == b2){frame--;}\r\n";
         out += "}\r\n";
         return out;
  }
```

```
public void writeFile(String fn){
  //System.out.println(output1());
  try {
    FileWriter fw = new FileWriter(fn);
    PrintWriter pw = new PrintWriter(new BufferedWriter(fw));
    pw.println(outFileHeader());
    pw.println(outDyogStoryClass());
    pw.println(outMyScreenClassHeader());
    pw.println(outMyScreenClassBackgrounds());
    pw.println(outMyScreenClassItems());
    pw.println(outMyScreenClassCaptions());
    pw.println(outMyScreenClassSpeechs());
    pw.println(outConstructorHeader());
    pw.println(outConstructorBackgrounds());
    pw.println(outConstructorItems());
    pw.println(outConstructorCaptions());
    pw.println(outConstructorSpeechs());
    pw.println(outConstructorEnd());
    pw.println(outPaintComponentHeader());
    pw.println(outPaintComponentBackgrounds());
    pw.println(outPaintComponentItems());
    pw.println(outPaintComponentCaptions());
    pw.println(outPaintComponentSpeechs());
    pw.println(outPaintComponentEnd());
    pw.println(outActionPreformed());
    pw.println(control);
    pw.println(outMyScreenClassEnd());

    //pw.println();
    pw.close();
  }
  catch (Exception e)
  {}
}

public String outFileHeader(){
        String out = "";
        out = "import java.awt.Container;\r\n";
        out += "import java.awt.FlowLayout;\r\n";
        out += "import java.awt.event.ActionListener;\r\n";
        out += "import java.awt.event.ActionEvent;\r\n";
        out += "import javax.swing.JFrame;\r\n";
        out += "import javax.swing.JButton;\r\n";
```

```
        out += "import java.awt.*;\r\n";
        out += "import java.awt.event.*;\r\n";
        out += "import javax.swing.*;\r\n";
        out += "import javax.swing.border.*;\r\n";
        return out;
}

public String outDyogStoryClass(){
        String out = "public class DyogStory extends JFrame{\r\n";
        out += "public static void main (String [] args) {\r\n";
        out += "new DyogStory();\r\n";
  out += "}\r\n";
  out += "public DyogStory(){\r\n";
  out += "setLocation (0,0);\r\n";
  out += "setSize (800, 600);\r\n";
  out += "setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);\r\n";
  out += "Container dyog_cont = getContentPane();\r\n";
  out += "JPanel dyog_controls = new JPanel();\r\n";
  out += "dyog_controls.setLayout (new GridLayout(5,1));\r\n";
  out += "dyog_controls.setSize(150,250);\r\n";
  out += "JButton button1 = new JButton (\"Next\");\r\n";
  out += "dyog_controls.add (button1);\r\n";
  out += "JButton button2 = new JButton (\"Previous\");\r\n";
  out += "dyog_controls.add (button2);\r\n";
  out += "MyScreen screen = new MyScreen(button1, button2);\r\n";
  out += "dyog_cont.add (screen, BorderLayout.CENTER);\r\n";
  out += "dyog_cont.add (dyog_controls, BorderLayout.EAST);\r\n";
  out += "show();\r\n";
  out += "}\r\n";
  out += "}\r\n";
  return out;
}



public void addControl(String id, int coor1, int coor2, int coor3, int coor4, int color, int type){
        if(type == 1){
          control += "draw_" + id + " = true;\r\n";
          //Set background
          }
        else if(type == 2){
          control += "draw_" + id + " = true;\r\n";
          control += id + "x = " + coor1 + ";\r\n";
          control += id + "y = " + coor2 + ";\r\n";
          //position item
   }
```

```
else if(type == 3){
        control += "draw_" + id + " = true;\r\n";
        control += id + "x1 = " + coor1 + ";\r\n";
        control += id + "y1 = " + coor2 + ";\r\n";
        control += id + "x2 = " + coor3 + ";\r\n";
        control += id + "y2 = " + coor4 + ";\r\n";
        control += id + "_color = " + color + ";\r\n";
        for(int x=0; x<speechs.size();x++){
          DyogSpeech s =(DyogSpeech)speechs.elementAt(x);
    if(id.equals(s.getname())){
            control += id + "_text = \"" + s.gettext() + "\";\r\n";
          }
        }
        //speech location
}
else if(type == 4){
        control += "draw_" + id + " = true;\r\n";
        control += id + "x = " + coor1 + ";\r\n";
        control += id + "y = " + coor2 + ";\r\n";
        control += id + "_color = " + color + ";\r\n";
        for(int x=0; x<captions.size();x++){
          DyogCaption c =(DyogCaption)captions.elementAt(x);
          if(id.equals(c.getname())){
            control += id + "_text = \"" + c.gettext() + "\";\r\n";
                }
        }
        //caption location
}
else if(type == 5){
        control += "repaint();\r\n";
        control += "theend = false;\r\n";
        control += "}\r\n";
        frame++;
        control += "if(frame == " + frame + "){\r\n";

        //end if structure and start new one for next frame
}
else{
        control += "theend = true;\r\n";
        control += "repaint();\r\n";
        control += "}\r\n";
        control += "if(frame > " + frame + "){\r\n";
        control += "frame = " + (frame + 1) + ";\r\n";
        control += "repaint();\r\n";
        control += "}\r\n";
        control += "if(frame < 0){\r\n";
```

```
            control += "frame = 0;\r\n";
            control += "repaint();\r\n";
            control += "}\r\n";
            control += "}\r\n";
    }
}

public boolean isDuplicate(String name){
        String temp;
        //check all backgrounds
        for(int x=0; x<backgrounds.size();x++){
          DyogBackground b =(DyogBackground)backgrounds.elementAt(x);
          temp = b.getname();
          if(temp.equals(name)){
                return true;
     }
    }

    //check all items
    for(int x=0; x<items.size();x++){
          DyogItem i = (DyogItem)items.elementAt(x);
          temp = i.getname();
          if(temp.equals(name)){
                return true;
     }
    }

    //check all speechs
    for(int x=0; x<speechs.size();x++){
          DyogSpeech s =(DyogSpeech)speechs.elementAt(x);
          temp = s.getname();
          if(temp.equals(name)){
            return true;
          }
        }

    //check all captions
    for(int x=0; x<captions.size();x++){
          DyogCaption c =(DyogCaption)captions.elementAt(x);
          temp = c.getname();
          if(temp.equals(name)){
                return true;
          }
        }

        //Identifier was not found at all
```

```
        return false;
}

public boolean isBackground(String name){
        String temp;
        //check all backgrounds
        for(int x=0; x<backgrounds.size();x++){
          DyogBackground b =(DyogBackground)backgrounds.elementAt(x);
          temp = b.getname();
          if(temp.equals(name)){
                return true;
          }
  }
  return false;
}

public boolean isItem(String name){
        String temp;
        for(int x=0; x<items.size();x++){
          DyogItem i = (DyogItem)items.elementAt(x);
          temp = i.getname();
          if(temp.equals(name)){
                return true;
          }
  }
  return false;
}

public boolean isSpeech(String name){
        String temp;
  for(int x=0; x<speechs.size();x++){
          DyogSpeech s =(DyogSpeech)speechs.elementAt(x);
          temp = s.getname();
          if(temp.equals(name)){
            return true;
          }
        }
        return false;
}

public boolean isCaption(String name){
        String temp;
  for(int x=0; x<captions.size();x++){
          DyogCaption c =(DyogCaption)captions.elementAt(x);
          temp = c.getname();
          if(temp.equals(name)){
```

```
                return true;
            }
          }
          return false;
  }
}
```

File Name: Main.java

```
import java.io.*;
import antlr.CommonAST;
import antlr.collections.AST;
import antlr.debug.misc.ASTFrame;



public class Main {
  public static void main(String args[]) {
    try {
      DataInputStream input = new DataInputStream(System.in);

      // Create the lexer and parser and feed them the input
      DyogLexer lexer = new DyogLexer(input);
      DyogParser parser = new DyogParser(lexer);
      parser.prgm(); // "file" is the main rule in the parser

      // Get the AST from the parser
      CommonAST parseTree = (CommonAST)parser.getAST();

      // Print the AST in a human-readable format
      System.out.println(parseTree.toStringList());

      // Open a window in which the AST is displayed graphically
      //ASTFrame frame = new ASTFrame("AST from the Dyog parser", parseTree);
      //frame.setVisible(true);

      DyogWalker walker = new DyogWalker();
            walker.prgm(parseTree);

    } catch(Exception e) { System.err.println("Exception: "+e); }
  }
}
```

Appendix 3: .bat files

File Name: dyogc.bat

```
set classpath=.;c:\antlr\antlr-2.7.4\antlr.jar
java Main < %1
javac DyogStory.java
```

FileName: dyogcr.bat

```
set classpath=.;c:\antlr\antlr-2.7.4\antlr.jar
java Main < %1
javac DyogStory.java
java DyogStory
```