THE DEVELOPMENT OF A DOMAIN SPECIFIC LANGUAGE FOR STORYTELLING MANAGEMENT

Valentina CRAEVSCAIA[1*], Viorel NOROC[1], Petru STRATULAT[1], Valentin FOROS[1]

[1]Department of Software Engineering and Automation, Group FAF-203, Faculty of Computers, Informatics and Microelectronics, Technical University of Moldova, Chisinau, Republic of Moldova

*Correspondent author: Valentina Craevscaia, valentina.craevscaia@isa.utm.md

Abstract. *This article presents an approach to the video game development, that is currently a very labor-intensive endeavor. The main purpose of this scientific paper is to present the created Domain Specific Languages (DSL) for Storytelling Management that differ from other alternative tools through the states of the character. More specific depending on the feelings and behavior, different events will be triggered and the character will pass through diverse changes and simulations.*

Keywords: *DSL, game, parse tree, grammar, storytelling, characters.*

Introduction

Movies, books, and other forms of popular culture have all been replaced by video games. It's possible that the industry's expansion is outpacing that of more traditional art forms. Much of the growing popularity of video games is due to advances in technology, such as alternate input devices and improved visuals [1,2]. Domain Specific Languages (DSL) are programming languages designed to enable a certain set of tasks in a given area.

This scientific research presents an approach to the Story management tool for operating in a deeper manner with the characters, beside the basic decision-making process. The proposed DSL differ from the other similar tools like Inky editor, scripting language based on the notion of using flow to mark-up pure-text in order to create interactive scripts [3], through the idea of operating with character, based on its condition the all-future events will be triggered, and vice-versa, based on the events happening to the character, its condition will be changing. An expert in the selected subject will use such a technology to try to boost his or her productivity.

Domain Analysis

People like level and story designers must communicate their concepts and ideas to programmers because all created material must finally be realized in a final software product. This can take a lot of time and effort, and it's simple for programmers to misinterpret the designers' genuine intentions. It would be much simpler, and indeed more efficient, if the designers could “code” their ideas in a way that is both natural for them, but also usable for the production of the game software [1]. A great solution for this problem, constitute the idea of creating a Domain Specific Languages (DSL) for storytelling. There were implemented several tools for operating with narrative text, for example Inky editor [3]. Analysing all existing Storytelling tools there were no tool that would be operating with the main character from the story, relying on the physical or mental condition and also on the events that are happening with it. Based on all these states, the idea is to trigger the future events and changes, that will influence the story development. The proposed DSL is mainly based on the Inky editor principle, an advanced script used for a narrative engine designed to slot into a game engine. However, the difference between Inky and the suggested DSL, is creating a simulation of the real life, in such way the game is more adaptive and interesting.

Language overview

It can be difficult and error-prone to develop a textual DSL by implementing its grammar. Since the programming language is based on Storytelling Management, the main focus was to make the writing process of the narration based on character condition. The language will provide an adaptive way for story and character development.

The proposed DSL consists of written representative samples of rules, where each rule refers to entities according to the data model, they are used to "validate" the program. Also, there are predefined keywords based on the grammar which help to the implementation of the lexer.

A programming language's grammar is a technical term for a set of formal rules that describes how the programming language is built and lists the valid tokens and lexemes. In Table 1 are described the meta-notation used in the Extended Backus-Naur Form, often used to present grammars.

Table 1.

EBNF Meta notation

<x>	means x is non-terminal
x or 'x'	means x is a terminal
[x]	means x is optional (0 or 1 occurrences of x)
x*	means 0 or more occurrences of x
x+	means 1 or more occurrences of x
"|" "{" and "}"	separates alternatives are used for grouping alternatives

The grammar of the DSL is G = (V_N, V_T, S, P) where:
Vn – nonterminal symbols,
Vt – terminal symbols,
S – starting symbol,
P – finite set of production rules.

The proposed language works around the player. It offers the possibility to add any stats and attributes to the player. If needed, there is also possibility for an inventory system and item equipment. The language also offers the possibility to create characters that the player can interact with and also enemies that the player fights with. To reduce the redundancy and ensure that every entity has the same defined stats and attributes, global stats and attributes can be defined. The language also has a dedicated part for items that are used in the story and the item has its stats and what will happen to the player if they use or equip the item.

In this language the user can declare global variables and functions. The supported data types are int, float, string, and bool. As for the functions, they can either return something or nothing. For the code logic, there is support for the basic mathematical operations, logic operations and the if statement. Now for the most important part of the language, the story flow.

The story is divided into nodes. The first node is always called START. Each node has the next properties: text – text that is printed when the node is entered, options – the options for this node that have their own properties and node type and a node can be either an encounter with a person, a battle or just showing the text and providing some options. Also, each node can have local variables and can go to other node without choosing an option. An option has a label, the text that is printed and optionally a condition for showing the option or changing the node. When an option or a node goes to node END, it finishes the story. The proposed language offers a simple flow that will make the developing of storytelling games easier.

Program example

```
GLOBAL_STATS{hp: 100; atk:0;}
GLOBAL_ITEM_STATS{value: 1;}
GLOBAL_ATTRIBUTES{name: "name";}
GLOBAL_EQUIPABLE{
   right_hand: NOTHING;
}
ITEMS{sword{EQUIPED{atk:10;}}}
PLAYER{
   stats{atk: 1;}
   attributes{name: "John";}
   inventory{money: 100; }
   equipable{right_hand: sword;}
}
NPCS{alexander{}}
ENEMIES{skeleton{}}
GLOBAL_VARIABLES{}
STORY{
   START{
      text: "you lost your way in the woods";
      options{
         1{
            label:"go left";
            text:"you decided to go left";
            goto: START;
         }
         r{
            label:"go right";
            text:"you decided to go right";
            goto: went_right;
         }
      }
   }
   go_right{
      text: "you found your way out";
      goto: END;
   }
}
```

Parsing tree

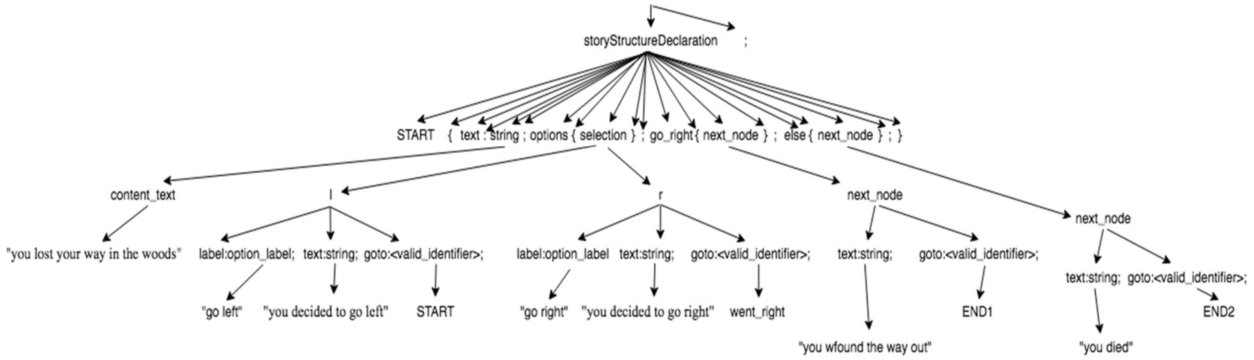
sourceCode
statement
storyStructureDeclaration
;
START { text : string ; options { selection } ; go_right { next_node } ; else { next_node } ; }
content_text
l
r
next_node
next_node
"you lost your way in the woods"
label:option_label; text:string; goto:<valid_identifier>;
"go left"
"you decided to go left"
START
label:option_label text:string; goto:<valid_identifier>;
"go right"
"you decided to go right"
went_right
text:string; goto:<valid_identifier>;
"you wfound the way out"
END1
text:string; goto:<valid_identifier>;
"you died"
END2

Figure 1. Parsing tree for a part of the program

Conclusions

Domain Specific Languages can be used for a variety of tasks. They can be utilized in a variety of situations and by a variety of people. This article, introduces a new DSL concept, which develop the idea of the Storytelling Management. This project presents an approach to the Story management tool for operating in a deeper manner with the characters, beside the basic decision-making process. The grammar implementation, define the general rules of the DSL, that later will perform the target-language code generation.

References

1. BEYAK, Lucas, CARETTE, Jacques. *SAGA: A Story Scripting Tool for Video Game Development* Ph.D. Thesis, Hamilton, Ontario (Canada): McMaster University, April 21, 2011. [accessed 28.02.2022]. Available: https://www.researchgate.net/publication/51935203_SAGA_A_DSL_for_Story_Management.
2. BANGEMAN**,** Eric*, Growth of gamming in 2007 far outpaces movies, music*[online]. 2008, [accessed 28.02.2022]. Available: https://arstechnica.com/gaming/2008/01/growth-of-gaming-in-2007-far-outpaces-movies-music/
3. Writing web-based interactive fiction with ink, [online]. [accessed 28.02.2022]. Available: https://www.inklestudios.com/ink/web-tutorial/.