Script for Visualization of Algorithms: Framework for Animation Environment and Composite Structures

A synopsis of the thesis to be submitted in partial fulfillment of the requirement for the degree in

MASTER OF TECHNOLOGY

In

COMPUTER SCIENCE & ENGINEERING

Submitted by

Dibyahash Bordoloi
(06CS6008)

Under the guidance of

Professor Abhijit Das
DEPARTMENT OF CSE IIT KHARAGPUR
1. Introduction

The field of Algorithm Animation (AA) has remained a research and educational interest over the decades. However, for many educators the benefits of an Algorithm Animation system still remains a vague idea.

A simple definition of algorithm animation is that it is a dynamic visualization with effects of all the processes that make up algorithms and the associated data structures. As such, any tool that captures the execution of a given algorithm can be termed as Algorithm Animation. In broad terms, an Algorithm Animation will be a composite set of tools in multimedia form such as graphics, associated animation, labels, voice, video, program code etc that simulates the working of the algorithm. In that it uses the abstractions of data, different operations and the semantics of the program that works behind the algorithm. Many different systems, tools and applications exist today in this domain. Educators and students also interested in deploying Algorithm Animation can utilize any or more of these tools by making an informed decision about a specific system or tool, or even can decide to develop their own system. The existing systems differ greatly in terms of how the system is built, how it will execute and how it will perform.

Algorithm animation concerns itself with illustration or dynamic execution of the intended behavior of a program in terms of visualizing the absolute fundamental operations of the program as it go through its run. Various studies show that such displays are quite handy in the field of education and research in the domain of design and analysis of different algorithms.

2. Objective and Motivation

The objective is to develop a scripting language that will be used to display the changing states of an algorithm in execution. The state changes are to be reflected on a cross platform browser for easy deployability. A Graphical User Interface is to be associated with the display with user controls.

Our motivation for building a scripting language for animation algorithm stems from the fact that animation, as a tool for learning has not been explored to that extent as much as it has been done on focusing on new innovations of visualization technology. There have been incredible advances in animation technology, but a full fledged widely used system for learning algorithms has not yet been deployed.

Given the fact that animation is an effective way for both teaching and learning algorithms, an easy way to create animations is in the interest for both teachers and students. If we concentrate solely on Computer Science algorithms and if we look around
for resources for algorithm animations we will come across variety of computer programs that animate a particular algorithm. Most of these programs are of stand alone variety, quite associated with the animated algorithm at large. So for each and every algorithm there exists a particular program in any of the available language. From the programmers point of view he has to write a separate program for a different algorithm. From the user’s point of view he is just a viewer of the output and is not involved in any way in the creation of the animation.

Here the motivation comes: Let both the teacher and student have a system where both feels that they have their hands on creation of the animation and understand it in a better perspective. And let the system be an easy one without being tied to a particular program or language. We can have a single set of predefined operations using which we can animate anything we want.

A scripting language is a easy way to accomplish complicated task. We focus on building a scripting language that will contain a minimal set of commands that will enable a teacher to animate any state and sequence of an algorithm. The student, when he feels that he now has some understanding of that particular algorithm can use the same set of commands to generate the sequence.

3. Users of an Algorithm Animation System

Roughly four groups of users can be targeted by an Algorithm Animation System. Most of the systems normally targets more than one type of users. The broad categories of users are

**Programmer**: This category of user is usually in the least targeted group. Since the role of the programmer is to implement, he is to code a given algorithm without any specific regard for the animation purpose. The programmer may, infact, be totally unaware of any future plans of animating the code in particular.

**Developer**: The developer plays an active role in the Algorithm Animation system. He has to design and build the system to bring out the desired animation.

**Visualizer**: The visualizer will generate the content of the animation and also edit the content.

**User**: The user or end-user will use the system and interact with the animation

4. Features of an Algorithm Animation System:

Some of the important features of a good Algorithm Animation System are:

- Good User Interface
- Accepting Program Data from users
- Availability of user controls
5. Types of Animation System

As of now there are around 300 Algorithm Animation Systems available, however only a very few of them have been extensively used and researched upon. After Java was introduced as a programming language around the mid 90s it has become a hot favorite for designers and programmers of Animation Systems. Virtually all of the systems available now are programmed in java, some requires downloading the system to the local machine and installing it, some run in client server architecture. A rough breakup of how the Algorithm Animation system has been implemented is given below as available from [9]

<table>
<thead>
<tr>
<th>Implementation Strategy</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java Applets</td>
<td>65 %</td>
</tr>
<tr>
<td>Java Local Application</td>
<td>30 %</td>
</tr>
<tr>
<td>OS specific binaries</td>
<td>3 %</td>
</tr>
<tr>
<td>Cross platform binaries</td>
<td>0.5 %</td>
</tr>
<tr>
<td>Flash</td>
<td>0.5 %</td>
</tr>
</tbody>
</table>

Table: Implementation strategy of Algorithm Animation Systems

6. Animation System based on Script

The system implemented is a script based system. There are various advantages that come from using a script. Some of them are:

- Scripts are simple and straightforward and easy to learn
- The user will have total control on the creation and movement of actors and objects.
- No special editor is required.
- The script can be generalized for the algorithm animation purpose

7. The Algorithm Animation System

The Algorithm System developed is built with the three types of user in mind: educators who would like to develop animations to teach animations, students who would develop their own animations and experiment with different data sets giving them a broad understanding of the underlying algorithm and the developers who would like to add extra features and capabilities to the whole system. The system offers a friendly
Graphical User Interface, supports creation of scripts in simple text files, a scripting language that provides generic components and operational effects for creation of animation, option for configuring to run as a local application or from a server, cross browser support and interaction with the animation at any instance of time during the run and changing the course of the animation.

To start an algorithm animation the user will write the script in a simple text file. The script is interpreted and displayed over a web browser. The GUI has all the user controls that are used to initiate, pause, resume or stop the animation. Control for speeding up or delaying the animation is also provided. Users are informed of any error in the scripting language whenever an error point is reached.

8. The Language

The language for animation is composed of commands that create objects and apply behavioral actions on those objects. Both primitive objects and composite objects that are directly applicable in the field of computer science are supported. Creation of an object is achieved by simply naming the object and passing the associated parameters. Generation of an animation is achieved by applying control commands on the objects.

The most primitive of the objects are presented directly by mapping some aspects of the core programming language to types of images that are the best representation of those objects. Composite structures are obtained by encapsulating some of the details which are basically values associated with a more primitive object and presenting a direct representation from the other available information.

The commands can be written for both sequential and parallel execution. For parallel execution the commands are to be written in a block that is characterized by a start-stop signifying keyword. Delays can be injected in between the commands.

Creating the script assumed the fact that the user has an understanding of the exact sequence of state change of the particular algorithm he wants to animate.

9. Design and Implementation

The total system has been developed by a team of four. Described in the following sections is the contribution made by the author.

9.1 The Framework:

The framework of the system has been developed with regarding to the following perspective:

- State changes are reflected over a widely used medium so that no particular application is required to download and install.
• A simple GUI that does not clutter the user’s workspace.
• A simple and generic script that does not require any concentrated effort to learn. The script is integrated with basic behavioral control.
• The repositories of content generation are in the same place.
• The GUI or Virtual Environment has user controls and these controls do not share space with content generators in the animation workspace.

Diagram: Framework of Script based Algorithm Animation System

9.2 The Object Perspective: Structures

Objects are static actors that require the actions of the Animation Generator for presenting behavioral aspects in the GUI interface. In other words behavioral control can be performed on the objects. The objects are in a lower level of the content hierarchy, just above the basic building blocks. Structures are special composite objects that are composed of one or more primitive actors that are in the lowest hierarchical level.
Structure objects interact with other objects and basic blocks in the virtual environment creating a series of Interactive behavior. While the user has all the behavioral controls at hand he needs to use those controls according to the targeted algorithm. An example of interactive behavior is ‘shiftArrow’. In this case one object (an Arrow) will have to locate some other object in the display environment and interact with it. The relative position of the object with reference to other objects all along will change dynamically and the control data is made available at run time.

The objects, and so the structures are loaded in the virtual environment by a ‘control performer’. The shape of the structure object does not interfere with user controls.

Structure objects are independent in nature and the control behaviors on them are mutually exclusive. It is possible to apply behavioral control on the structures in sequence or in a block of composite behaviors for creating a parallel execution.

For developers the structure objects present another set of group which they can work upon to produce more composite structures.

**9.2.1 Implementation of the Structure Objects:** Each of the objects is implemented as a separate java class in individual files. They are contained in a single package. A special class is provided which can group a number of structures so that a basic behavior can be applied on that group as a whole. Classes are provided for creating an abstraction in scaling the objects. Functions have been provided to assist users in accessing state data of the objects in action.

**9.3 The Execution Environment perspective:**

The guiding philosophy behind the Execution Environment is simple: present the user with a dynamic display of algorithm in execution and arm the user with enough widgets to control different aspects of the execution.

In the hierarchical level of total system framework this environment sits at the top most level. This is where the user will interact directly with the objects and the behavior of the objects. The system is capable of providing two logical levels of workspace, one dedicated to the user and one to the algorithm in execution. While these two workspaces do not overlap to create behavioral problems, there is enough support to provide a clean interaction between the two.

The salient features in this environment are:

- A simpler GUI that can give the user a ease in learning devoting less time in the intricacies of the environment and more time in the actual programming and animation.

- The delay between presenting the script and running the execution sequence is totally negligible.
• Giving the user a view that the GUI itself is the main tool for generating animations.

9.3.1 Implementation of the Execution Environment:

Providing a simpler Executing Environment requires a finer level of programming. The implementation in Java has been arranged in two different packages. One package is used to create the bare frame for the environment. This package also contains class files that extend the basic java Applet class for creation of animation. The main package contains all the classes that create the GUI and the performing subsequent interactions with objects and controls. A specific class creates the user controls. As soon as the user interacts with the environment using a control another class in this package initiates the execution. Multiple threads are spawn to maintain a fine level of interaction as more and more objects come into play with their associated behavior. To keep track of the animation sequence is also an important implementation as the user may want to go back and forth. Classes are provided for injecting delays and providing user controlled speed is provided.

10. Comparisons with other systems

We have compared our system with some of the other available scripting language in terms of features and execution. A table is appended below which summarizes the findings.

<table>
<thead>
<tr>
<th></th>
<th>High Level style of coding</th>
<th>Smart Placement of objects</th>
<th>Generic</th>
<th>Timing</th>
<th>Debug</th>
<th>Compatibility with other scripting languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>SALSA</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>None</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SAMBA</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Delay</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ANIMAL</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Delay, Duration</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>DsCats</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Matrix</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Our system</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Delay</td>
<td>Yes</td>
<td>Provided</td>
</tr>
</tbody>
</table>
11. Further Work

In the future more composite and complex structure can be created either from the existing components or from scratch. Complex behaviors which can take advantage of improved graphics capabilities both at hardware and the programming language level can also be added further. User interaction with the animation at run time can be improved upon including provision of extra windows. Another added feature can be embedding of sound in the animation.

Bibliography


