

Script Synthesis Tool for Non-Experienced Programmers

Koji Yokokawa
Mamezou Co., Ltd.
koji.yokokawa@riverside.skr.jp

Abstract

This paper describes the design of a script synthesis tool, named “Syncriptor”, for non-experienced programmers in Squeak/Tweak. With this tool, users can synthesis or analyze multiple Etoys scripts of an object.

Syncriptor is designed with Programming by Example and Visual Programming. Therefore, this tool can be used by non-expert users like children. All operations and effects of the tool are visual. Users manipulate the tool easily by a mouse and realize the effect of complex combined scripts with real time feed back.

1. Introduction

Many children take apart a mechanical clock to figure out how it works. In so doing, it satisfies their curiosity and they learn through actual experience. If the user handles computer programs in the same way, it is a great help especially for non-experienced programmers.

Scripting systems are a popular way in end-user programming. Squeak/Tweak [1] [2] has a tile scripting system, named ‘Etoys’ [3]. Users have the ability of programming combining visual tiles of statements provided by Etoys. Etoys is a good scripting system to make a simple script but it is not easy to combine multiple scripts. To add or remove a script, a user should open a script viewer and operate switches that turn on or off. In addition, users cannot move a script for an object to another object easily. When these operations about multiple scripts can be done more easily, Etoys will have greater usability.

Visual Programming is a good design to understand and manipulate mechanisms of computation. Text based interfaces require a high-level of ability in computer operations by the user. Visual interfaces are easier than text based interfaces. Even if a user cannot manipulate through the keyboard, they can modify some part of a program visualized in the display through a mouse.

Programming by Example is a powerful design for non-expert users. In this design, a user makes or finds an example object that does what the user wants, and instructs a computer to do the same as the example object. A user can check the behavior by the examples. Not all the examples are made by the user. Pre-programmed examples can be used as parts for the new program. Therefore, when a user cannot create a program from the beginning, the user can create a new program by including other programmed objects.

1.1. Related Works

Several approaches have been aimed at creating visual programming tools that were supposed to give not just adults but children the power to create their own programs on the computer, for example, ToonTalk [4], Stagecast Creator [5], HANDS [6], Viscuit [7], JiVE [8], and Etoys. These all have original scripting mechanisms to make programming easy. Stagecast and Viscuit are also designed with Programming by Example. However, these easy scripting systems have simple and static script structures that enable the user to achieve the manipulation of scripts easily. But they lack an easy interface that combines or divides scripts, and have difficulty in understanding complex scripts. These systems also have a lack of time concerned behavior changing mechanisms like the time line in Syncriptor.

The time line concept is used in Flash [9]. The time line in Syncriptor is able to combine two or more scripts logically. However, the time line in Flash is not used for programming logic but only for visual effects like moving positions, changing shapes and other such related actions.

2. Concept

Syncriptor is a tool to give scripts or shapes to an object in accordance with a time line. Syncriptor is not a tool only to make graphic animations but also it is a tool to experiment with scripts. It makes complex scripts easier and so users can combine multiple scripts into one object and add or remove parts of pre-

programmed scripts visually with this tool. All these effects are operated visually by the users. Users put in or take out visualized scripts and watch the effects of these scripts and then users can understand the meaning of the scripts.

2.1. Use-case

A simple use-case description is like this. A user makes ‘mold objects’, which will be models of parts for a synthesized object. Then the user can set another object as a ‘screen object’. When the start button is pushed the time cursor of the Syncraptor runs and the aspects of the mold objects are projected into a screen object according to the time cursors motion.

For example, a user draws two fish with the normal paint tool in Squeak/Tweak and adds scripts (see Figure 1). One of the fish has a ‘forward by’ script and the other has a ‘turn by’ script. Next, the user sets these two fish as mold objects and another sketch as a screen object. These scripts and graphics of two fish are projected into the screen object according to the time cursor runs down. The result of this is that the screen object changes its shape to become like the fish and swim around in a circular motion (see Figure 2).

The user can modify the parameters of the projection process by mouse through the visualized user interfaces. The speed of the time cursor is changed by the arrow size of the start button. The duration and timing of the projection of a mold object is modified by size and position of the green bar on the syncrript tile.

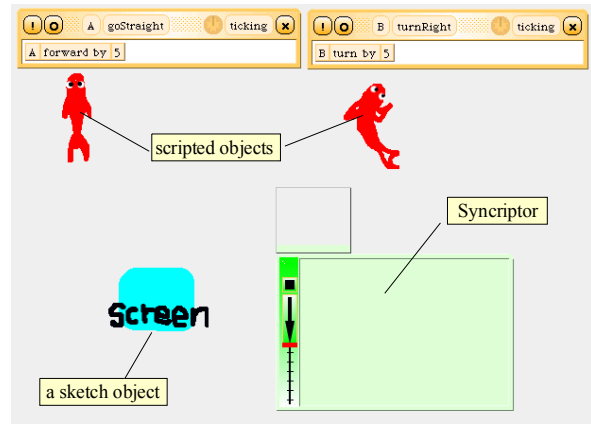


Figure 1: An example configuration, a user prepares two scripted red fish and a non scripted sketch object.

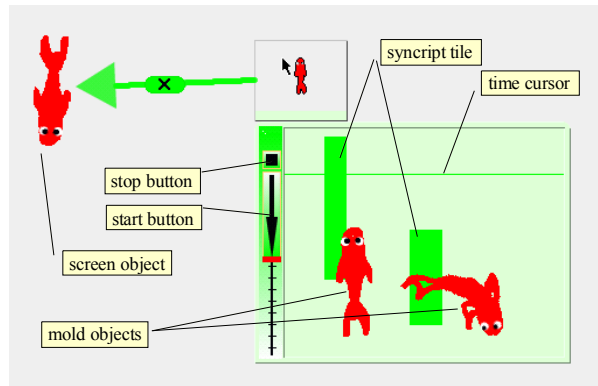


Figure 2: The result of the example, the user sets the two fish as mold objects and the sketch as a screen object.

3. Design of Syncraptor

3.1. Base Architecture

The Syncraptor has syncrript tiles and a time cursor (see Figure 3). The time cursor indicates which mold objects are used to project aspects into the screen objects at any given moment. A time cursor runs down on it and it changes and activates the tiles.

A Syncraptor is given two types of objects by a user, one is a ‘mold object’ and the other is a ‘screen object’. Some of the mold objects aspects are projected into screen objects through syncrript tiles at the time indicated by the time cursor. Both types of objects are normal Squeak/Tweak Player objects.

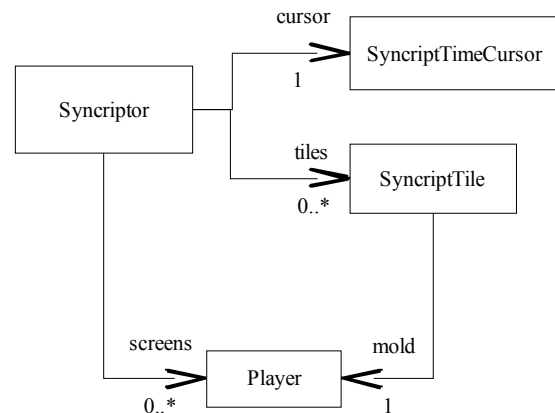


Figure 3: Class diagram of Syncraptor base architecture.

3.2. Syncrript Tile and Time Cursor

A syncrript tile consists of a vertical bar and a modal object. A tile represents the projection parameters of a modal object. The tile is also a user interface of these parameters. The mold object is a normal Tweak object. Therefore, the objects are made up of drawn shapes and scripted with Etoys tiles by the users. When the normal object is dropped on a Syncrriptor window, the object becomes a mold object of a new syncrript tile.

During the time it is running, the screen object is changing as the mold objects, with the time line. Aspects of mold objects are projected into screen objects when the tile crosses the time cursor.

The time cursor touches the bottom of the window then restarts from the top. The projection time line is repeated until the user pushes the stop button.

Users can choose aspects of the mold object to project into the screen object through switches in each of the syncrript tiles. These projection aspects are graphic, size, heading, nose, and scripts.

3.3. Script Synthesis

The Syncrriptor projects aspects periodically. State type aspects like graphics or size are just copied to screen objects during each projection. Scripts are not copied during each projection because compiling takes a long time. Scripts of a mold object are copied once and this is done the first time the syncrript tile crosses the time cursor. After that, the copied scripts remain in the screen object and start/pause messages are sent to the scripts repeatedly corresponding to the syncrript tile crossing the time cursor or not. When a syncrript tile is removed or the screen object is detached, the copied scripts are removed from the screen objects.

Syncrriptor copies a script with its trigger event. A ticking script is copied with the same ticking timing. Each ticking script has its own ticking timing so that the copied scripts run in the screen object with its own ticking timing. A script that has mouse down triggers is copied with the same mouse down trigger to a screen object, and then the copied script is activated by a click of the screen object.

Script copying is done with dynamic compiling. It duplicates the tile structure of the original script and replaces all 'receiver' references with the screen object then compiles the copied tiles.

When a time cursor intersects more than two syncrript tiles, then all scripts of each mold object are copied into a screen object and run.

For example, there are two mold objects which has a ticking script each. One is 'forward by', the other is 'turn by'. When the first time of the time cursor crosses the 'forward by' object, the 'forward by' script is copied and compiled to the screen object and the copied script is started. During the process, the syncrript

tile bar is under the time cursor and the script stays active and gets ticks from the Tweak system. When the cursor passes the tile off, the copied script is stopped, not removed from the screen. Therefore, the copied script is reused. The script of the 'turn by' object is copied in the same manner. Start / stop timing of these scripts are defined by each bar of the syncrript tile. The user can change the duration of each script by a mouse. For instance, when some part of the 'forward by' bar intersects the 'turn by' bar, at the intersected area these two scripts are activated same time, so that the screen object turns and goes straight simultaneously. At the other area, only the 'forward by' is activated, so that the screen just goes straight. (see Figure 4)

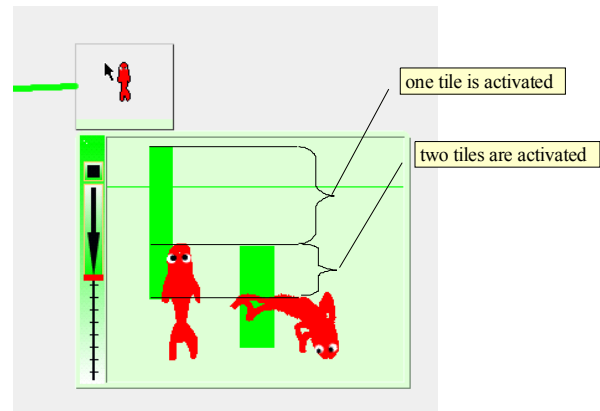


Figure 4: The user can control the tile activities through the position and size of each tile.

4. Conclusions

Syncrriptor is an experimental tool to investigate the possibility and effect of script synthesis. This implementation confirms script synthesis makes complex multi-scripts easier. The users can directly view the sequence of programs. Visualized logics of the complex scripts will have many educational effects for non-experienced programmers. Users can learn how to construct complex scripts with multiple scripts.

In addition, users can assign scripted objects as a part of other object. Therefore, users will make programs that are more complex by simple integration of scripted objects without editing scripts.

Copying scripts to other objects is a useful technique as Programming by Example. The users make scripts on an object and test it then use it as a part of another program. That sequence makes programming intuitive and easy.

The time line is good way to make animations easy. On the other hand, some scripts are not fit for changing

according as time passing. It can express these static scripts as a full sized syncrript tile bar. However, the time line mechanism might have been separated from the script synthesis to simplify it.

5. Future work

Syncrriptor will be open for the public under the Squeak License after cleaning the code up.

We will make more samples and documentation for non-expert users to get feedback from them and so be able to refine it further.

Acknowledgements

I would like to thank Alan Kay and Kim Rose for the support they have given me through out the development of this project. I am also grateful to the generosity of the Information-technology Promotion Agency Japan.

This paper is partially supported by Exploratory Software Project grant of Information-technology Promotion Agency Japan.

6. References

[1] Squeak, <http://www.squeak.org>

[2] Tweak, <http://tweak.impara.de/>

[3] Etoys, <http://www.squeakland.org/author/etoys.html>

[4] Ken Kahn, "ToonTalk - An Animated Programming Environment for Children", *Journal of Visual Languages and Computing*, June 1996.

[5] Stagecast Creator, <http://www.stagecast.com/>

[6] J.F. Pane, B.A. Myers, and L.B. Miller, "Using HCI Techniques to Design a More Usable Programming System," *Proceedings of IEEE 2002 Symposia on Human Centric Computing Languages and Environments (HCC 2002)*, Arlington, VA, September 3-6, 2002, pp. 198-206.

[7] Viscuit, <http://www.viscuit.com/>

[8] Jana Hintze, Maic Masuch. "Designing a 3D Authoring Tool for Children," *c5*, pp. 78-85, Second International Conference on Creating, Connecting and Collaborating through Computing (C5'04), 2004.

[9] Flash, <http://www.macromedia.com>