# Skeleton – easy simulation system

*Takashi Yamamiya*
propella@yuri.sakura.ne.jp

## Abstract

Skeleton is a visual scripting environment as an extension of Squeak etoy [1] with Connectors [2] system to make mathematical and physical simulation for non professional computer user. Squeak etoy system is a unique attempt to provide effective way of programming for children in learning environment. Skeleton makes logical relationships among graphical objects in the etoy system with spreadsheets-style interface, and users can describe object's behavior in declarative representation.

Concreteness is one of the key words of end user scripting. Direct manipulation of objects on the screen is helpful to understand what happened in your computer. Skeleton's spreadsheets style interface realizes this concreteness to show any input and output data same time.

Sometime user scripting system like etoy has a problem of modularity. This aspect is important as a basis of reusing, thus Skeleton has some features for reusing. Tree structured naming system like ECMAScipt [3] is used to access to Skeleton object by name. And modularity is realized by sheet-card mechanism that is possible to reuse a behavior of objects in another context in Skeleton.

## Background

There are a lot of works in end-user scripting or visual programming. ThingLab [4] emphasizes declarative relationship among objects as idea of "constraint" and, allows user to build application with concrete manipulation of graphical objects. Fabrik [5] has a metaphor of wired components. These wires are representation of multiple path data flow.

Spreadsheets are widely used in end-user computing today. They have a matrix format and each cell is connected by formula that represents a relationship among each cell. As all values deal with calculation include input and output are shown to users all the time. This character is fit to logical programming as interface of prolog [6]

Spreadsheet has not only used as business application, but also as user friendly interface in the experiment of end-user scripting. ASP [7], NoPumpG [8], and C32 [9] are attempts to apply a power of spreadsheet's understandability to build graphical interface.

Squeak etoy system provides user friendly and instance based programming interface. Instance base means that each objects behavior are described by each object itself, not by abstract class definition. The aim of the Skeleton project is to mix etoy's flexibility with logical perspicuity and verifiability borrowing spreadsheet style interface.

## User interface

This section is described about user interface of Skeleton using a geometric example borrowed from ThingLab [4]. This example is to demonstrate a geometric theorem, which states given an arbitrary quadrilateral, if one bisects each of the sides and draws lines between the adjacent midpoints, the new lines form a parallelogram.

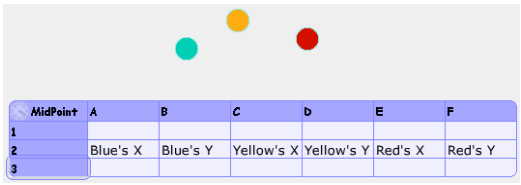



Fig. 1 Skeleton sheets and morphic points

Morphic is basic graphical object system in Squeak. User can manipulate morphs by mouse operation directly, or by tile scripts with slots of the morph. Skeleton has capability dealing with these morphic slots. Now we pick up a Skeleton sheet and three points on the screen from "supply flap". Then we write object's name on the sheet as labels like "Blue's X, Blue's Y …"

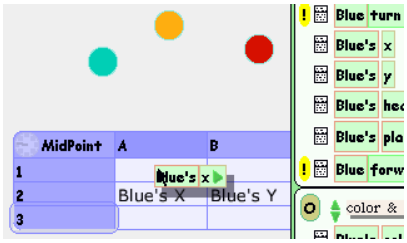



Fig. 2 drop a slot of morph to cell

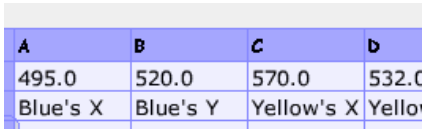



Fig. 3 attached cells

Using drag-and-drop, a slot of morph can be connected with a sheet's cell. We attach each x-y coordinate of point into cells.

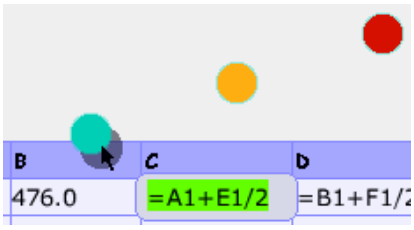



Fig 4. formula keeps midpoint

When we put a formula in an attached cell, related morph is moved by the formula. In this case, **"= A1 + E1 / 2"** means **"yellow's x = (blue's x + red's x) / 2"** (in Smalltalk syntax, all binary operators have same priority). The formula is used to calculate coordinate of midpoint of blue and red. As ordinary spreadsheets, first character '=' in cell is indicated as formula.


Fig 5. card made from sheet

Formulas on top row of the sheet have special meaning. These formulas can be reused on other sheets as card. A card is made by dragging change-size halo to become the sheet small.

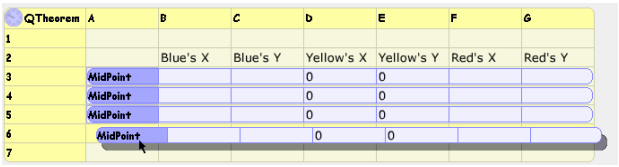



Fig 6. embded card in other sheet

Card can be embedded any sheet. We make four copies of the card and put into new sheet named "QTheorem".


Fig 7. attaching othe slot to card

After we assign all card with more morphic objects and joining by Connectors' [2] line. The project can show a quadrilateral theorem.

Fig 8. demonstrate theorem

## Basic feature

Skeleton has basic functions of spreadsheet. Like commercial spreadsheets, it has matrix includes cells with formulas. To describe a formula, Skeleton uses mix syntax of Excel-like expression and normal Smalltalk expression. Each cell is pointed as 'A1' style position or user defined string name. Any numbers, strings or Smalltalk objects are stored in cells. Additionally, etoy's slots can be attached with cells, and any morphs are controlled by Skeleton.

Relative reference and listing reference of cells is not supported. For this purpose, card mechanism is used for reusing formula. After user makes a formula in a spreadsheet, the user can reuse the formula to make its card from the spreadsheet. The relationship among card and spreadsheets (sheet) is similar to instance and class in object oriented concepts.

Some predefined library cards are provided. They include mathematical function like sum and average. Name is used for specifying these library cards. User defined named cells, sheet, and library cards are in same name space, and the system bounds their name using same mechanism. A formula can use the name with tree structured path to search these names.

## Architecture

One of the most important requirements of Skeleton was flexibility. Skeleton is not stand-alone system, yet it needs to connect to existing etoy's base system, and needs various kind of ability to make interesting application. So Skeleton consists of several parts for this requirement.

### Kernel and library

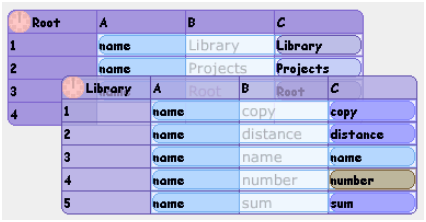
Kernel provides basic services include handling spreadsheets-style interface and naming protocol. But any concrete data storage and computation is not included. Kernel defines just how an object shows and what name is the object. Therefore any different data structure and

updating strategy can be mixed. This is because to suppose to be written various logical primitive library as building block for specific demand. Skeleton's standard spreadsheet SkSheet is just one of the libraries.

## Name

SkObject (Skeleton object) is basic element in Skeleton. All SkObject can have spreadsheet like user interface and a name. Basic interaction between user and SkObject is based on a text sequence with its name, not direct object reference. It is because portability and transparency. All objects in Skeleton must be able to access by string name or cell position to be accessed.

Cell position is used for local access in object, but using name can be got across other object. Any SkObjects don't hold its own name, but they are named by parent object. To keep the parent object, a SkObject has a special property named 'parent'. Property is a kind of cell it has a name but doesn't have a position; a property is used for storing information for system. All SkObjects construct one tree as follow.




Root
  Root
  Library
    Libraries …
  Projects
    User projects…
      User defined sheets …

Fig. 9 system sheet and naming structure

Root is a start object for the name tree. It has a reference for Root itself, and for Library and Projects. Library keeps each library names and objects. Projects are associated with Squeak Project's object. Each user defined sheet is associated in its project, and all sheets on same project shared same namespace.

To find a name, the system searches SkObject as dictionary, then searches parent object if it couldn't find. Library that is for common predefined library is searched after Root.

## Updating

Skeleton has own updating system extended original Squeak updating mechanism. Its brief flow is followed.

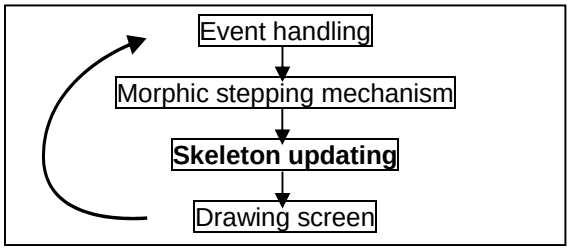



Fig. 10 updating flow in a frame

Each frame, the sequence is executed by UI process. Skeleton objects are updated just before drawing screen because it must be synchronized with morphic stepping cycle to draw smooth graph or animation in etoy's project.

## Card

Card is basic mechanism for reusing in Skeleton. Any Skelton object can make its own card. Then the card is used to embed other object. A card holds original object's behavior and most messages are delegated to original object. First row of the object has a special meaning. When a card is embedded in other sheet, the card tries to update cells under the card as these are top rows of the original object.

In card execution, the data flow related cells can be bidirectional and there are no distinction of input and output to show on spreadsheet. But some library assumes that first cell is output and others are input customary (e.g. sum). First cell is left end cell by ordinary. But user can change the direction of card.

## Library's protocol

All Skeleton's library object is a subclass of SkObject that defines relationship among cells in Skeleton. For kernel, Skeleton object behaves just as a dictionary. The dictionary's key can be a string and/or a point. A user defines new Skelton object to make Smalltalk class or to construct from standard Skelton spreadsheet (SkSheet) as normal spreadsheet applications.

#at: and #resolveOn: messages are the most common protocol of SkObject. A parameter of #at: is point or string. String key is used for named access, and it is applied tree structured search path. But a point is noted in the object locally. #resolveOn: message is send when the object should do something. In many cases, an object reads its cells, solves some problems, and stores the answer to a cell. The augment of #resolveOn: is the SkObject itself or another SkObject that is used to reuse.

## Planning

A lot of researchers have proposed methods for determining execution order for declarative language so far. In Skeleton, any SkObject can have its own strategy, but most common used is SkSheet's planner. It is simple

enough but can handle circulation dependency. This planner has two phases.

### Building phase

When a user updates a formula, the building phase is invoked. System scans all cells left to right, and top to bottom, and then makes a graph showed about cell's dependency. Circulation dependency found afterward is taken out simply. Then an order list of cell execution and a reverse reference map is made from the tree.

### Updating phase

A SkObject is called each frame before drawing, but any formulas are not executed until it meets conditions. In example, each cell of SkSheet is updated just when dependent cell is updated by other force using a reverse reference map. In this strategy, circulation constraint is resolved even if the dependence graph was not complete.

## Programming Style

Code reading is regarded as an important issue. Once a program is written, the code comes out and is read by other people widely. Despite reading code is harder than writing code, much effort must be gone for maintenance code after developed. The concrete oriented scripting style of Skeleton helps writing code as readable documentation. Using common example of "Fahrenheit / Celsius unit converter" that is also in ThingLab, we describe a typical programming style for example in Skeleton.

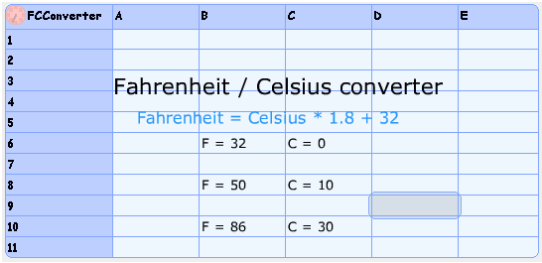



Fig. 11. Describe specification

First, to make sure what the code is, some documentation and examples are written the sheet. These examples show how the code does in concrete form.

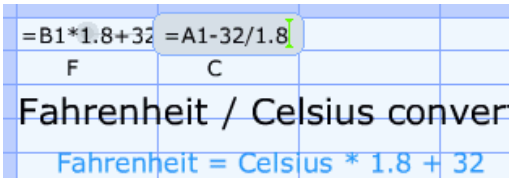



Fig. 12 writing code the top of sheet

Along the specification, we write codes on top row of the sheet.

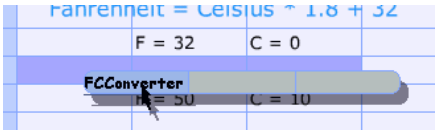



Fig. 13 embed card into itself

When the code is complete, we can make a card of the sheet, and embed the card into original sheet itself. The card is used to test if the code satisfies given specification, and used as live documentation for someone or yourself in future.


Fig. 14 using on other sheet

## Conclusion

The Skeleton system provides simple and flexible extension of Squeak etoy system. It controls any etoy's object with easy spreadsheet style interface. The spreadsheet has abundant ability to use dynamic nature of Smalltalk directly, and has modularity with card system.

## References


[1]. http://www.squeakland.org/

[2]. Connectors http://nedkonz.dhs.org:8080/Ned/8

[3]. http://www.ecma-international.org/publications/standards/Ecma-262.htm

[4]. A. Borning. "The Programming Language Aspects of ThingLab, A Constraint-Oriented Simulation Laboratory" 1981

[5]. D. Ingalls, S. Wallace, Yu-Ying Chow, F. Ludolph and K. Doyle, "`Fabrik: a visual programming environment" 1988

[6]. Michael Spenke and Christian Beilken. "A spreadsheet interface for logic programming" 1989

[7]. K.W. Piersol, "Object Oriented Spreadsheets: The Analytic Spreadsheet Package" 1986.

[8]. C. Lewis. NoPumpG: "Creating interactive graphics with spreadsheet machinery" 1990.

[9]. B. Myers, "Graphical Techniques in a Spreadsheet for Specifying User Interfaces" 1991.

[10]. Gregg Rothermel, Margaret Burnett, Lixin Li, Christopher DuPuis, Andrei Sheretov "A Methodology for Testing Spreadsheets" 2001

[11]. Marc Stadelmann. "A spreadsheet based on constraints" 1993



This project was sponsored as Exploratory Software Project by Information-technology Promotion Agency Japan.