

Revista del Centro de Investigación. Universidad La Salle

Universidad La Salle

evm@ulsa.mx

ISSN (Versión impresa): 1405-6690

ISSN (Versión en línea): 1665-8612

MÉXICO

2008

Pablo C. Herrera Polo

SOLUCIÓN DE PROBLEMAS RELACIONADOS AL DISEÑO DE SUPERFICIES
COMPLEJAS: EXPERIENCIA DE PROGRAMACIÓN EN LA EDUCACIÓN DEL
ARQUITECTO

Revista del Centro de Investigación. Universidad La Salle, enero-junio, año/vol. 8

(sup), número 029

Universidad La Salle

Distrito Federal, México

pp. 55-60

Solución de problemas relacionados al diseño de superficies complejas: Experiencia de programación en la educación del arquitecto

Pablo C. Herrera Polo

Facultad de Arquitectura, Universidad Peruana de Ciencias Aplicadas

Escuela de Posgrado, FAUA, Universidad Nacional de Ingeniería. Lima, Perú.

E-mail: pablo@espaciosdigitales.org; <http://www.espaciosdigitales.org/santiago>

RESUMEN

En este artículo, el autor explica por qué los arquitectos que saben cómo utilizar el lenguaje script tienen una ventaja sobre los que sólo saben cómo manipular un software específico. La razón detrás de esta afirmación es saber cómo la programación libera al arquitecto de las reglas y el lenguaje del software interactivo y se puede convertir en un aliado. Para probar este punto, el autor organizó dos talleres donde estudiantes de una escuela de arquitectura aprendieron a utilizar *Rhinoscript*. Los estudiantes no crearon una nueva interfaz, sino que utilizaron una existente. Ellos adaptaron el programa (Rhino) a un problema de diseño que habían formulado al comienzo del taller. Los estudiantes podrían también utilizar *MaxScript* (3DS Max) y *MelScript* (Maya).

Palabras clave: lenguaje script; Rhino; educación arquitectónica, superficies complejas; Grupo de Fabricación de Diseño Digital.

ABSTRACT

In this paper, the author explains why architects who know how to use scripts have an advantage over the ones who just know how to manipulate specific software. The reason behind this affirmation is that knowing how to program frees the architect from the rules and language of the interactive software. To prove his point, the author organized two workshops where students at the school of architecture learned to use *Rhinoscript*. The students didn't create a new interface, but used an existing one. They adapted the program (Rhino) to a design problem they had formulated at the beginning of the workshop. Students could have also used *MaxScript* (3DS Max) and *MelScript* (Maya).

Key Words: Script language; Rhino; Architectural education, complex surfaces; Digital Design Fabrication Group.

ANTECEDENTES

Durante la última década los medios digitales usados en arquitectura han evolucionado en respuesta a lo que parece ser un complejo conjunto de influencias emergentes que han transformado el concepto de *representación* de la forma por el de *generación* de la forma (Oxman, 2005). Tanto la representación como la refiguración, se resuelven usando en las facultades de arquitectura un software interactivo. El código detrás del icono o la programación enmascarada por la interfase, permitió que los diseñadores

implementaran sin dificultad las computadoras a su trabajo. Esta dependencia a la interfase, trajo como consecuencia la adaptación de sus diseños al lenguaje propuesto por los programadores (Vinograd y Flores, 1987). Un lenguaje de reglas que el programador considera que podemos usar para diseñar y del que terminamos dependiendo. Si terminamos adaptando nuestras propuestas al software del que disponemos “la forma terminará siguiendo al software” (Serriano, 2003). Esta dependencia no es intencionada, muchos diseñadores que se involucran por primera vez con sistemas informáticos, asumen que usar una aplicación interactiva es el único camino para trabajar con el computador, lo que los aleja de otra manera de resolver problemas de diseño usando computadores (Kesson, 1995). ¿Pero cómo ir perdiendo esta dependencia?. En este documento se toma como base la propuesta de Mitchell (1987, 1989), para quien la programación es una alternativa para explorar sistemas generativos, trabajo que ha sido continuado por Larry Sass, Director del Digital Design Fabrication Group (DDFG) del MIT. Y es que la fabricación digital ha sido la responsable que se redescubra el uso de la programación en arquitectura (McCullough, 2006) al explorar también el diseño de superficies complejas usando VBScript, lenguaje diseñado para conectar e integrar componentes en un software interactivo. Un lenguaje con nombres como MaxScript en 3DS Max, MelScript en Maya, Rhinoscript en Rhino, o AutoLISP en AutoCAD.

El uso de la programación para resolver problemas en superficies complejas tiene un enorme potencial para los diseñadores, pero uno de los factores adversos es la inversión considerada para aprenderla. Senske (2005) observó el aprendizaje en programación de estudiantes del MIT y encontró que debido a que la velocidad de implementación tecnológica supera a la implementación pedagógica, la auto instrucción es la principal fuente de conocimiento para estar actualizado en técnicas de programación. Pero esta afirmación es válida para estudiantes de postgrado que ya tienen un conocimiento de exploración previa con diferentes técnicas y métodos de diseño. Sass (2003) propuso una experiencia pedagógica en arquitectura: entender los diseños y luego enseñar como generar nuevas alternativas e ideas usando el lenguaje de scripts. Para Sass, los scripts permiten obtener resultados rápidamente, dándonos mayores oportunidades para desarrollar la creatividad pues nos permiten iterar diferentes alternativas que finalmente evaluaremos.

Pero en Latinoamérica no ocurre lo mismo, de modo general aún mantenemos dependencia al software interactivo. Salvo casos particulares como el taller de programación y arquitectura usando Visual Basic realizado en Brasil por Celani (2002), entonces estudiante del PhD de MIT implementado bajo el sistema propuesto por Mitchell, Liggett y Kvan (1987).

En Enero del 2006, participé en el Workshop Computational Design Solutions en la Escuela de Arquitectura del MIT, dirigido por Kenfield Griffith y John Snavely (ambos también del MIT) en el que se usaba Rhinoscript para explorar soluciones de diseño. Mi objetivo era encontrar respuestas a lo siguiente: 1) ¿Es posible explorar ideas y formas geométricas usando la programación sin depender directamente de un software interactivo?. 2) ¿La programación es sólo para programadores o puede ser incluida en una facultad de arquitectura como respuesta al problema de generación de formas complejas? y; 3) Si un software interactivo no es cien por ciento paramétrico. ¿Será posible que al usar el script para la generación de superficies estemos frente a una solución paramétrica?. La primera respuesta fue afirmativa, la segunda concluía en que no sólo era para programadores para para saber si se podía incluir en una facultad de arquitectura era necesario desarrollar un taller con un público heterogéneo y no sólo de postgrado y, la tercera respuesta resultó positiva. Estas respuestas fueron el motivo que propició realizar este proyecto e implementarlo en una facultad de arquitectura en Latinoamérica, en donde el acercamiento a estas experiencias son limitados y sólo se alcanzan a través de estudios de postgrado fuera de la región.

En Febrero del 2006 tomé contacto con la Facultad de Arquitectura y Urbanismo de la Universidad de Chile y se concretó realizar el Taller Soluciones de Diseño Computacional en noviembre del 2006 y agosto del 2007 en el que participaron como instructores, Griffith, Snavelly, Cardoso y Herrera. La experiencia y los resultados son los que se presentan en este congreso. El sistema de enseñanza propuesto se basa en las experiencias de Sass y los talleres impartidos por equipos de investigación del DDFG del MIT en Estados Unidos y Europa principalmente, usando Rhino como plataforma y Rhinoscript como lenguaje.

OBJETIVOS

Acercar a los participantes a la programación básica para explorar la forma y el espacio. El objetivo fue proporcionarles herramientas conceptuales y técnicas que les permitan decidir cuándo y de qué manera la programación puede volverse un aliado del proceso de diseño, entendiendo primero cómo podrían resolver el problema sin un software interactivo, en vez de inducirlos a resolverlo con las técnicas de uno.

Cada acción o diseño propuesto, simple o complejo se puede abstraer a una secuencia de pasos. En matemática y programación, una secuencia es un algoritmo. Un algoritmo contiene un número determinado de pasos (o conjunto de instrucciones) para encontrar una posible solución a un problema (Terzidis, 2006). Cuando tenemos una forma muy compleja e intentamos representarla, llevamos al límite las técnicas que aprendimos sobre un software interactivo de modelado, y llegamos a un punto en que estas ya no son suficientes para responder rápidamente a cambios constantes en un proyecto. Es ahí donde propiciamos en los participantes el ejercicio del pensamiento computacional a través del lenguaje interpretado y el lenguaje script.

DESARROLLO

El taller fue implementado en agosto del 2007 basándonos en los resultados del primer taller de noviembre del 2006. Participaron 38 estudiantes de grupos heterogéneos: No graduados (25), graduados (4) y asistentes de taller (9). La duración del taller fue de diez días consecutivos en el que se establecieron tres fases.

La primera fase, de *implementación teórica*, fue realizada en dos días de 9 horas cada uno y se utilizó la guía que pueden descargar desde <http://www.espaciosdigitales.org/santiago/2>. El objetivo de la primera sesión fue el de la exploración geométrica. Por espacio de una hora, los alumnos sin conocimiento previo de programación entendieron la lógica de convertir actividades cotidianas en un listado de instrucciones, lo que se entiende como pseudo código (donde aún no hay programación), sólo una secuencia de pasos que conducen a realizar una tarea. Seguidamente, lo cotidiano o problemas diarios fueron cambiados por operaciones algebraicas en el que se introdujo el concepto de declaración y asignación de datos de una variable. Al terminar la sesión los scripts no llegaban a tener más de quince líneas y se logró relacionar datos con componentes de Rhino. En la segunda sesión, se trabajó la asignación de valores a los elementos de un arreglo o matriz, imprimir el resultado de un elemento de un arreglo, obtener un valor de un usuario así como leer el contenido de una posición específica en un arreglo e imprimirlo. Las operaciones algebraicas fueron reemplazadas por las geométricas lo que condujo al alumno a desarrollar la habilidad de relacionar puntos, líneas y superficies para explorar diferentes alternativas de diseño al terminar las dos sesiones. En esta segunda sesión los scripts superaban las cuarenta líneas de código en promedio. Esta primera fase, concluyó con la tercera sesión. Para ello se les pidió un día antes, que formaran grupos con un máximo de cuatro participantes y un mínimo de dos y que trajeran una propuesta de trabajo. Se solicitó que definieran el problema pensando cómo lo resolverían si no tuvieran un computador. Es importante destacar que la

experiencia de haber intentado resolver el problema por medios tradicionales era una condición, de lo contrario no podrían analizar el problema sin tener conocimiento de posibles soluciones. Era necesario al menos que conocieran los parámetros que crean las geometrías propuestas, sin importar si las habían o no construido en la realidad. Los resultados de esta presentación pueden ser vistos en el enlace <<http://www.espaciosdigitales.org/santiago/2/Grupos/indexp1.html>>.

La segunda fase, de *supervisión y seguimiento* se realizó en tres sesiones de 8 horas cada una (de la cuarta a la sexta sesión) y fueron decisivas para guiar a los 12 grupos. Se desarrolló el análisis del problema, aprender cómo conectar las ideas y geometría en un algoritmo usando los componentes de Rhino.

Cada grupo realizó una presentación al final de la sexta sesión que puede ser revisado desde <<http://www.espaciosdigitales.org/santiago/2/Grupos/indexp3.html>>.

La tercera fase, de *crítica y revisión final* se realizó en tres sesiones de carga horaria variable. Los alumnos solucionaron algunos problemas en sus scripts, tenían que preparar una presentación de un máximo de diez diapositivas, un video con el script ejecutando y un panel con su propuesta. El objetivo de la presentación final a la comunidad académica, fue el de motivar la inclusión de estas herramientas en los talleres de diseño.

CONCLUSIONES

Algunas comparaciones entre el taller de noviembre (2006) y agosto (2007). Los participantes se duplicaron de 15 a 38. El grupo femenino se incremento considerablemente de 1 a 7. La encuesta respondida por 33 de los 38 participantes dió los siguientes resultados. 18 nunca habían usado Rhino anteriormente, 5 lo hacían eventualmente y 10 lo habían usado siempre. Lo que afirma que este tipo de exploración puede darse sin tener conocimientos de un software interactivo, dado que el script genera modelos e ideas que parten de la propuesta del participante. Ocho tenían algún conocimiento de programación (incluido 4 que participaron en el primer taller) y 25 se acercaban a la programación por primera vez, lo que nos indica que no es necesario tener conocimientos previos de programación para participar. Los 33 participantes manifestaron sentirse al terminar con más conocimiento y satisfechos con los resultados, así como haber explorado posibilidades de diseño. En su totalidad recomendaron difundir el taller. Ante la pregunta sobre si depende o no de las herramientas propuestas por el programador: 19 manifestaron que no, 4 respondieron que dependen ahora de ambas, y 10 respondieron que sentían dependencia (hay que indicar que el motivo de esa dependencia como lo indicaron es la falta de práctica, pero que en definitiva superada esa limitación estarían en camino de su independencia).

Entre algunas de las exploraciones están: *Reptar* (Rossel, Serres, Solar), tuvo por objetivo evidenciar patrones de comportamiento en un proyecto y determinar si el resultado obedecía a decisiones y adaptaciones basadas en condiciones ambientales.

¿Que regula la porosidad en una superficie?

De entre muchas, Rodolfo Ugarte y Tomas Cox tomaron el paso de la luz dentro de un ambiente así como la densidad estructural de un objeto. En base a ello, el script *Superficie esponja* (Figura 1), crea y organiza puntos sobre una superficie como si fuera una cuadrícula. Los puntos son reorganizados según el paso de luz o densidad. Los puntos resultantes son triangulados para crear polígonos cerrados, que son la base de los anillos interiores. Al iterar, se escoge la solución tridimensional resultante.



Figura 1. Superficie esponja

Otro grupo se planteó **¿Cómo adaptar formas complejas a cualquier superficie también compleja en 3D?** *Girascript* (Basaez, Enríquez, Pulgar), parten del principio geométrico que configura y ordena las semillas del girasol en el corazón de una flor para definir un módulo (semilla). La semilla producida algorítmicamente, podía adaptarse sobre cualquier forma existente 3D. (Figura 2).

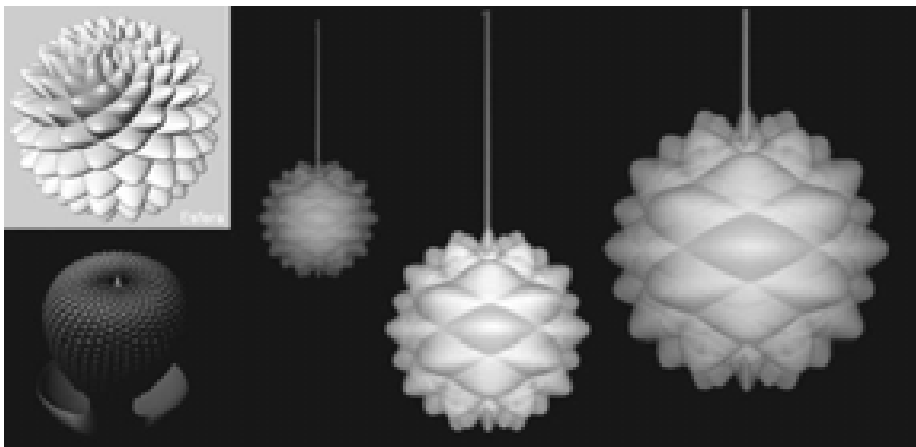


Figura 2. Girascript

OBSERVACIONES

Aunque los resultados no pueden ser generalizados para otros contextos geográficos, la experiencia sí permitió que los participantes obtengan una solución basada sólo en conocimientos básicos de programación, explorando alternativas para un problema planteado, dejando que el análisis sobre el problema permita crear un programa que de solución al modelado tridimensional tradicional de formas complejas. Ello les permite ahora tener la posibilidad de elegir en qué momento la programación puede ser un aliado.

RECONOCIMIENTOS

A Pedro Soza, Pilar Barba y Marcelo Valenzuela de la Universidad de Chile por llevar adelante el proyecto y el financiamiento de los dos talleres. A la disposición de Kenfield Griffith, John Snavely y Daniel Cardoso del Massachusetts Institute of Technology por su intensa colaboración y experiencia. Y a cada uno de los que participaron en los dos talleres que son desde ahora, la semilla que permitirá que esta propuesta continúe en Latinoamérica.

REFERENCIAS

Celani, G.: 2002, *Beyond analysis and representation in CAD: a new computational approach to design education*, tesis, Departamento de Arquitectura, MIT, Cambridge.

Kesson, M.: 1994, *An Introduction to 3D Computer Graphics*. Wellington.

McCullough.: 2006, 20 Years of Scripted Space en M. Silver (ed), *Programming Cultures: Art and Architecture in the Age of Software*, *Architectural Design*, 76, 4, pp. 12-15.

Oxman, R.: 2005, Conferencia no publicada *Digital design: The theoretical foundations of digital architecture*, FAU, Universidad de Chile.

Senske, N.: 2005, *Fear of Code: An Approach to integrating Computation with Architectural Design*, tesis, Departamento de Arquitectura, MIT, Cambridge.

Sass, L.: 2003, *An Interview with Professor Larry Sass*, <http://web.media.mit.edu/~federico/creativity/sass/sass_trans.htm>.

Serrano, P.: 2003, Form follow the software en *ACADIA 22, Connecting*, K. Klinger (ed.), The Association for Computer Aided Design in Architecture, Mansfield.

Terzidis, K.: 2006, *Algorithmic Architecture*, Elsevier Ltd., Oxford.

Winograd, T., y Flores F.: 1987, *Understanding computers and cognition: A new foundation for design*, Addison-Wesley, Reading.