Rhinoscripting y Grasshopper a través de sus instructores: un estudio de patrones y usos

Rhinoscripting and Grasshopper through the instructors: A study about patterns and conventions

Pablo C. Herrera Polo
Universidad Peruana de Ciencias Aplicadas, Perú.
pablo@espaciosdigitales.org

Resumen: *It is common today the use a graphical user interface and techniques to automate a process through computerization. By contrast, when trying to learn computational approaches, we have not overcome the learning curve and many of the workshops have not had the expected results to prolong their use. In this research we explore the trends in this process, from those that generate the generic object to those that appropriate it by modification. The set of case studies presents patterns and uses of those instructors who have come to be use algorithms intensively to solve a design problem.*

Palabras clave: Self-taught Programing; rhinoscripting; grasshopper; patterns

Antecedentes

Rhinoceros o *Rhino* de la empresa *McNeel* es una plataforma que centraliza tres entornos de trabajo. (1) La interfaz original de íconos desde 1998; (2) escribiendo el problema de diseño línea por línea usando Rhinoscripting (RS) desde alrededor del 2003; y (3) con componentes para controlar la historia y relaciones de un objeto usando Grasshopper (GH) desde finales del 2008. (Herrera, 2010, 215). El conjunto de *patrones* y *usos* de aquellos instructores que han realizado los principales talleres en los dos últimos entornos de trabajo es el punto de partida de esta investigación.

Los instructores y usuarios en general utilizan estas plataformas de acuerdo al tipo de control que interviene en sus diseños. Las técnicas que implican usar la interfaz principal son lineales y consecutivas, aportan con su resultado una solución hermética de representación, sin posibilidad de regresar a modificar alguna de sus partes si no se han archivado versiones previas del modelo. Este entorno permite representar formas muy complicadas usando técnicas simples y métodos cortos de memorizar. Los instructores aprenden a utilizar la interfaz interactiva sólo como base previa al uso de GH o RS; ésta es empleada sólo como plataforma complementaria. Cualquier usuario, al abrir o ver un archivo representado de esta manera, no puede entender cómo se hizo, porque, al igual que en el dibujo manual, todas las técnicas del proceso han sido eliminadas para dejar una posibilidad por vez.

Por el contrario, los siguientes entornos de trabajo, guardan el proceso de manera escrita (RS) y gráfica (GH), lo que permite a los instructores poder dejar abierta la revisión de los mismos cuando el archivo es abierto y ejecutado.

RS supera las técnicas secuenciales y no inicia con formas, convenciones tradicionales de representación, ni sirve para construir un modelo con un conjunto de librerías. Se identifican primero las variables que gobernarán el resultado; requiere que el problema sea matematizado, geometrizado y traducido con un vocabulario y gramática (sintaxis) a través en un algoritmo; permite explorar posibilidades cambiando cada línea, como se ha podido demostrar en los casos de estudio analizados. Hoy, la tendencia a escribir un problema de diseño usando *scripts* es una realidad que el parametricismo ha adoptado (Schumacher, 2009), justificado porque los íconos (estructuras de código prefabricadas por un programador) ajustan, a través de sus protocolos, limitaciones para representar un proyecto, efecto que se ha reducido en gran medida usando RS. "*Un usuario final programa para resolver tareas inusuales o repetidas de un problema bien descrito*" (Woodbury, 2010, 65, 185). Sin embargo, una desventaja común de aquellos que dejaron RS por GH, es que el primero no es interactivo, el código se ejecuta,

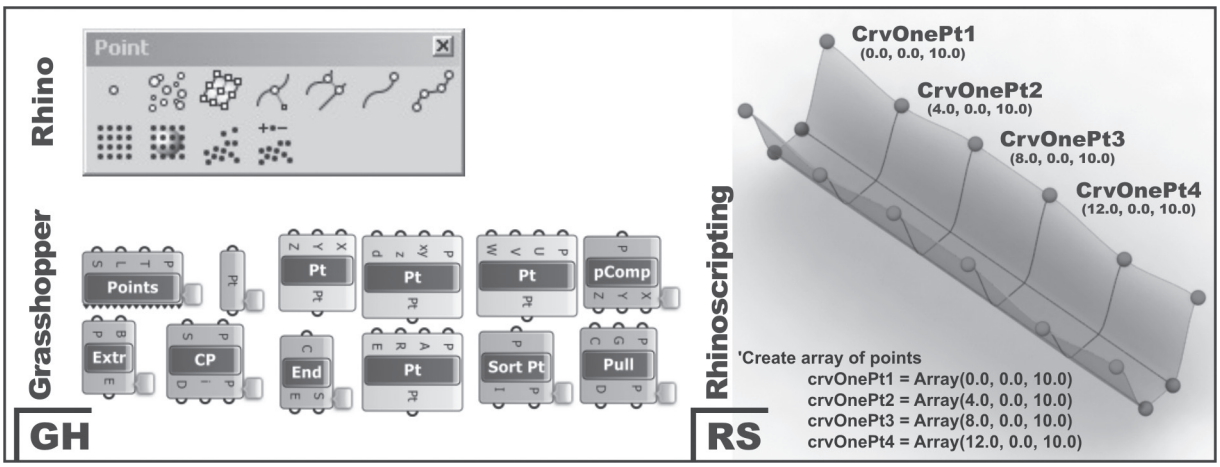

Fig. 1. Técnicas para representar un PUNTO: Usando la Interfaz principal de Rhino (izquierda), Grasshopper (centro) y Rhinoscripting (derecha).

se escribe y se corrige línea por línea, con Rhino como enlace para su visualización. *Python* abre posibilidades para compartir el código entre Microsoft Windows y Mac, pero aún sigue compartiendo un entorno de trabajo similar al de RS.

Por otro lado, GH se basa en un conjunto de reglas interactivas alojadas en un componente; cada una permite modificar su código para controlar las partes del modelo. Schumacher (2011,354) sostiene que "*la gran ventaja de GH es que traduce gran parte de la sintaxis de los scripts a un lenguaje gráfico de relaciones para cada uno de los componentes que la conforman*". Esa interactividad implícita para ejecutar el cambio de un parámetro y visualizar el resultado ha motivado que su uso y preferencia sobre RS sea más intenso en Latinoamérica que en otra región del planeta (Herrera, 2010, 341).

Casos de estudio

Hace dos años se documentó una primera cronología resumida, distribución geográfica y relación de métodos, patrones y convenciones de usuarios de RS (Herrera, 2009, 340). Al año siguiente, se presentó un análisis de preferencias entre usuarios de RS y GH, y se demostró que, en Latinoamérica, RS no tuvo ni tiene el impacto académico y práctico que el alcanzado por los arquitectos en proyectos realizados en EEUU y Europa. RS se ha utilizado con menos frecuencia y gradualmente ha sido reemplazado por GH (Herrera, 2010, 213).

La primera generación de instructores aparece entre 2003 y 2004: David Rutten en Europa, Stylianos Drit sas en EE.UU y el equipo de Roland Snooks y Robert Stuart-Smith en Australia. Ellos publicaron respectivamente *vbScripting/Rhinoscript Handout, Scripting and Computational Geometry* y *KokkugiaWiki*, que fueron las primeras referencias de programación usando RS y están disponibles desde esa fecha en internet. Las dos primeras son consultadas mensualmente por alrededor de 2500 usuarios según estadísticas disponibles en el portal http://issuu.com/pabloherrera/.

Una segunda generación de instructores aparece en el 2006 y se caracteriza por publicar todo tipo de trabajo personal utilizando RS. Toman como referencia los manuales, códigos y ejercicios de la primera generación. Se han identificado más de una docena de métodos y cientos de bitácoras con información de diferentes autores. A esta generación, pertenecen los primeros instructores que llegan a Latinoamérica (Chile) hacia el año 2006. Fueron experiencias de programación, bajo un esquema académico muy similar a los primeros talleres de EE.UU. o Europa (de 3 a 10 días de duración). En la Región Andina, las experiencias se inician en el Perú (2008) y en Colombia (2009), bajo el mismo esquema de implementación del Cono Sur. En toda la región, la implementación se estableció en un ambiente académico, diferente al realizado en el hemisferio norte del planeta y Australia, regiones en donde, "*lo más importante, fue que (el scripting) se desarrolló y perfeccionó en la práctica profesional y no en la academia*" (Leach, 2010). La implementación en Latinoamérica se basó en metodologías elaboradas en el *Digital Design Fabrication Group* (DDFG) de MIT, la *Architectural Association Design Re-*

search Laboratory (AADRL) o en el *Instituto de Arquitectura Avanzada de Catalunya* (IaaC), lugares de donde provienen los primeros instructores.
En todos estos grupos y en diferentes regiones del planeta, se utilizaron como temas, algunas de las técnicas de producción digital clasificadas por Iwamoto (2009): *Sectioning*, *Tesellating*, *Folding*, *Contouring*, y *Forming* o las documentadas por Sass (2010): *Bricks*, *Shapes* y *Structures*.

Patrones de implementación

Dentro de espacios académicos, los instructores han alternado su experiencia bajo tres *patrones* de implementación: a) Desde su motivación por el autoaprendizaje, b) Desde su experiencia como estudiantes de maestría y doctorado, y c) Realizando visitas académicas externas.

a) Autoaprendizaje

Motivados principalmente por estudios de postgrado en sus países e influenciados en cursos promovidos por implementaciones locales previas, se reúnen en este grupo, arquitectos que no sólo buscan un camino para darle solución a cuestionamientos personales sino que comparten su experiencia e investigaciones en bitácoras, cursos y tutoriales, con la única motivación de difundir sus exploraciones.
En el resto del mundo Davide del Giudice (madeincalifornia 2005) con una bitácora de 683 entradas, Dave Pigram y Iain Maxwell (supermanoeuvre 2006), Lucio Santos (2007) con 55 entradas, Marco Vanucci (opensysdesign 2007) con 43 entradas, Tobias Schwinn (2008), Andrew Kudless (matsysdesign 2008), Karl Daubmann (paramod2 2008) con 101 entradas o Andrea Bugli (2009).
En Latinoamérica, Chile es uno de los países de la región que ha contribuido de manera significativa a utilizar estas técnicas localmente: Guillermo Parada (gt2p 2006), Diego Pinochet (sCRIPT-O 2007), Francisco Calvo y Katherine Cáceres (Tectónicas Digitales 2010) que realizaron la primera traducción al español del manual de Andy Payne de GH. En México, Rodrigo Medina (designplaygrounds 2008), ha realizado en casi dos años alrededor de 11 talleres en diferentes ciudades de México y Barcelona.

b) Experiencia de estudiantes de maestría y doctorado

Es el grupo que más ha publicado su trabajo; alternan por lo general con el grupo a) y c). El retorno a Latinoamérica de estudiantes de maestría y doctorado durante la década pasada, ha reforzado la integración de tecnologías emergentes en ciudades importantes.
Uno de los casos que cubre el cono sur es el Laboratorio que incorpora proyectos financiados por el gobierno chileno bajo la dirección de doctores egresados de la Universidad Politécnica de Catalunya: Rodrigo García Alvarado de la Universidad de Bio Bio (Chile 2008, 2009), Underlea Bruscatto de UNISINOS (San Leopoldo 2008), Mauro Chiarella de la Universidad Nacional del Litoral (Santa Fé 2008). Ernesto Bueno (Curitiba 2009) es otro instructor que luego realizar sus estudios en Europa se trasladó a Brasil para desarrollar una actividad importante con GH.
Otro grupo de iniciativas están asociadas a egresados del IaaC. Gabriel Ochoa (FABLab Medellin 2010), el también colombiano Rodrigo Toledo, han llevado la experiencia académica FABLab Open Workshop (FLOW) a dos versiones en la ciudad de Medellín. En Perú, Luis Odiaga (2010), regresó a Lima en el 2009 y durante el 2010 promovió dos talleres bajo el título *Artificial Wrong* (AW).

c) Visitas Académicas Externas

Iniciativas organizadas por arquitectos investigadores e involucrados en la práctica, que viajan hacia diferentes partes del mundo para desarrollar talleres por temporadas breves o semestres enteros. La organización es realizada por la misma universidad o por iniciativas de alumnos. La totalidad de la primera generación pertenece a este grupo y ejercieron la programación en un entorno de sintaxis escrita (RS). De la segunda generación tenemos a Marc Fornes, uno de los fundadores de *theVerymany* (2005), le siguen *Live Architecture Network* o *LaN* conformada por Luis Fraguada, Monika Wittig, Shane Salisbury, CarloMaria Ciamploli y Aaron Willette.
Para el caso Latinoamericano, las primeras experiencias fueron en la Universidad de Chile. (Santiago, 2006, 2007) a cargo de miembros del DDFG de MIT que fueron alternándose entre Kenfield Griffith, John Snavely, Daniel Cardoso y Skylar Tibbits. Ese mismo grupo realizó otros talleres: en la Universidad Peruana de Ciencias Aplicadas (Lima, 2008) y en la Universidad de los Andes (Bogotá, 2009); el único taller de Marc Fornes en la región fue en la Universidad Técnica Federico Santa María (Valparaíso, 2009)
Otro grupo de experiencias se han promovido utilizando entornos de una sintaxis gráfica (GH) como principal herramienta. Sistemáticamente se inician con el DOF (Design Optimization and Fabrication) de McNeel y

se ha realizado en Colombia, Chile, Argentina, Perú y Brasil desde el 2009. Le siguen los talleres del Tooling Group (Barcelona, 2009) conformado por el brasileño Affonso Orciuoli y el colombiano Pablo Baquero que han realizado experiencias en la Universidade Presbiteriana Mackenzie. (Sao Paulo, 2009) y la Universidad Javeriana (Bogotá, 2010).

Conclusiones

Carpo (2011, 126) sostiene *"que todo diseño paramétrico, implica inevitablemente dos niveles de autoría: el autor principal que diseña el objeto genérico (el programa o serie o notación generativa), y el autor secundario, que hace específico ese objeto genérico con el fin de usarlo con un propósito particular"*. El caso latinoamericano, de manera general, pertenece al segundo tipo; según la cantidad de casos de estudio revisados, EEUU, Europa, Asia y Oceanía representan al primer tipo.

Se logró identificar un proceso, en el que el autor principal publica QUE?? EL SCRIPT??, el secundario lo mejora y adapta hasta volverlo suyo, para nuevamente publicarlo y dejar que el ciclo continúe. Lo que cada instructor publica es un *patrón*, "*una solución genérica a un problema compartido"* (Woodbury, 2010, 8).

Weinberg (1971, 193) sostiene que *"para aprender a programar uno debe aprender de sí mismo y de sus problemas, impulsado por la motivación y el entrenamiento"*.

Según Graham (2004,148) "*una de las razones por las que no podemos solucionar por nuestra cuenta un error del sistema operativo Microsoft Windows es porque no tenemos el código fuente"*. Raymond (1999), asoció este modelo al de la construcción de una catedral, donde la solución sólo pertenece a los que la producen. Diferente es el modo de trabajo en el sistema operativo Linux, donde se evidencia los errores al exponer su código, y los usuarios siguen contribuyendo a mejorarlo.

Raymond sostiene que "*a mayor cantidad de revisores, los errores en un problema serán encontrados con facilidad"*. A esta observación se le conoce como la Ley de Linus, y fue tomada del modelo de trabajo utilizado en el sistema Linux creado por Linus Torvalds, en donde *"el código fuente, evoluciona constantemente y no es un objeto acabado ni fijo, porque mantiene una relación recíproca entre solución y descubrimiento de problemas"*. (Sennett, 2008, 39).

Aprenderemos a resolver con más rapidez un problema y contribuiremos a potenciar la exploración de soluciones en la medida que expongamos ese problema a nuestros pares. Los instructores evidenciaron sus errores, y permitieron a cualquier usuario plantear correcciones tanto de sus códigos como del diagrama que los relacionaba. Un modelo con o sin parámetros, que no almacena ni muestra su proceso o las variables que lo gobiernan, sólo producirá en su hermetismo una solución concreta en la que el o los autores son los únicos que la conocen. Aprendemos de nuestros procesos no del objeto. Sennett (2008, 20) sostiene que "*para desarrollar una habilidad en alto grado son necesarias alrededor de 10.000 horas de experiencia"*. Podemos emplear ese tiempo para ejercitar nuestras técnicas, pero de cada experiencia sólo queda en el objeto que la representa. Por el contrario, si la experiencia es almacenada en un código (RS) o en una definición (GH) estaremos seguros que podremos volver a ella tantas veces como necesitemos reutilizarla o emplearla como punto de partida para nuevos proyectos.

En esta investigación se puso en evidencia que ambas generaciones de instructores compartieron y publicaron sus trabajos, como una actividad cotidiana que permite potenciar el autoaprendizaje otros grupos. Ello ha contribuido, a que técnicas tan complejas como la programación sigan siendo utilizadas por comunidades académicas y profesionales como una importante tecnología emergente de base para las siguientes generaciones.

Referencias

- Carpo, M. 2011. *The Alphabet and Algorithm.* Cambridge: MIT Press
- Graham, P. 2004. *Hackers & Painters: Big Ideas from the Computer Age.* Sebastopol: O'Reilly Media Inc.
- Herrera, P. 2009. *Patterns and conventions using Rhinoscripting language*. Proceedings of the 13th Congress of the Iberoamerican Society of Digital Graphics, Sao Paulo, Brazil, November 16-18, 2009. Disponible en: http://cumincades.scix.net/data/works/att/sigradi2009_1085.content.pdf
- Herrera, P. 2010. *Disruptive Technologies: Scripting and Digital Fabrication in Latin America*. SIGraDi 2010 Proceedings of the 14th Congress of the Iberoamerican Society of Digital Graphics, pp. Bogotá, Colombia, November 17-19, 2010, pp. 213-216. Disponible en: http://cumincades.scix.net/data/works/att/sigradi2010_213.content.pdf
- Leach, N. 2010. *Definitions: Parametric and Algorithmic Design*. Recuperado en marzo de 2010, de http://parasite.usc.edu/?p=443

- Iwamoto, L. 2009. *Digital Fabrication*. New Jersey: Princeton Architectural Press
- Raymond, E. 1999. *The Cathedral & the Bazaar.* Sebastopol: O'Reilly Media Inc.
- Sass, L. 2010. *Printing Architecture: Digitally Fabricated Buildings*. SIGraDi 2010 Keynote Lecture of the 14th Congress of the Iberoamerican Society of Digital Graphics. Bogotá, Colombia, November 17-19, 2010.
- Schumacher, P. 2009. Parametricism: A New Global Style for Architecture and Urban Design. *AD Architectural Design,* 79 (4), 14-23.
- Schumacher, P. 2011. *The Autopoiesis of Architecture: A New Framework for Architecture. Vol. I.* West Sussex: Wiley.
- Sennett, R. 2008. *The Craftman.* New Haven: Yale University Press.
- Weinberg, G. 1971. *The psychology of computer programming.* New York: Van Nostrand Reinhold
- Woodbury, R. 2010. *Elements of Parametric Design.* New York: Routledge.