Smalltalk y CuisUniversity

Introducción

El objetivo de este apunte es describir brevemente cómo es un ambiente de objetos Smalltalk y cómo se trabaja y se interactúa con él, particularmente con la distribución CuisUniversity.

Ambientes Smalltalk

Lo más importante y lo que más caracteriza a programar en Smalltalk es en el hecho de que en todo momento vamos a trabajar con un **ambiente de objetos vivos**. Esto es bastante diferente a lenguajes de programación convencionales, en donde escribimos código fuente en archivos de texto, y tenemos herramientas que nos sirven para manipularlos (syntax highlighting, refactorings, atajos de teclado, soporte para ejecución) que sumadas conforman lo que se denomina Entorno de Desarrollo Integrado, o IDE por sus siglas en inglés.

En un ambiente Smalltalk, no se trabaja sobre archivos. Más bien, se trabaja directamente sobre el ambiente en sí. El ambiente "es" el IDE. Por eso se dice que se trabaja con objetos "vivos". La gran particularidad es que los ambientes Smalltalk están escritos en Smalltalk. Por eso se dice que los ambientes Smalltalk son metacirculares. Pueden ser escritos, y modificados completamente en sí mismos.

Por este motivo, la experiencia de trabajar en Smalltalk es bastante diferente a otros lenguajes. Al abrir un ambiente, parecerá que estamos iniciando un sistema operativo diferente, con su propio manejador de ventanas, editores de texto y demás herramientas. Todas y cada una, formadas por objetos que están vivos.

Hay dos características que tienen estos ambientes: el **feedback inmediato** y la **manipulación directa** de objetos. El feedback inmediato tiene que ver con que cualquier interacción con el ambiente inmediatamente genera un resultado (visual o no); no hay un paso de compilación en donde hay que esperar para ver el resultado en algún otro lado. La manipulación directa se refiere a que cada objeto puede ser inspeccionado (es decir, ver cómo está compuesto) y ser modificado a nuestro gusto. No hay restricción de lo que se puede ver o editar. Todo es posible de cambiar, lo cual tiene sus grandes ventajas y también sus riesgos. Es muy fácil "romper" el ambiente modificando algo muy esencial (por ejemplo, cómo se suman los números, o cómo se comportan los booleanos). ¡Y es muy divertido también! Es algo muy común en el proceso de aprendizaje. Rápidamente podemos iniciar una versión nueva del ambiente si eso pasa, ya lo veremos más adelante.

Smalltalk "a secas"

Smalltalk es un **lenguaje de programación orientado a objetos puro**. Esto último quiere decir que (prácticamente) todo lo que se escriba va a ser en términos de la unidad esencial de la POO: la **colaboración** (el envío de un mensaje a un objeto).

Con Smalltalk se considera que se inició la programación orientada a objetos (hay un excelente artículo que explica cómo sucedió, "The Early History Of Smalltalk") y casi todos los lenguajes OOP se inspiraron en Smalltalk, aunque con el tiempo los lenguajes nuevos fueron incorporando más elementos que no son colaboraciones (lo cual los hace más "impuros").

Imagen y Máquina Virtual

¿Cómo se ejecuta un ambiente Smalltalk en nuestra computadora? Necesitamos de dos cosas:

- La **imagen**, que en la jerga Smalltalk significa el "mundo de objetos" con el que vamos a trabajar. Es una "foto" de todo el sistema, que se carga a memoria al iniciar el ambiente, se puede modificar y descartar los cambios, o guardarla, ya sea sobreescribiéndola, o generando una nueva. Generalmente se representa como un archivo binario de unas decenas de MB.
- La **máquina virtual** (VM) que es el programa capaz de interpretar una imagen y ejecutar todo lo que ocurre en ella.

Entonces para iniciar cualquier ambiente, lo que debemos iniciar es la máquina virtual, indicándole como parámetro con qué imagen lo queremos iniciar. Ya veremos detalles sobre esto más adelante con un ambiente en particular.

Esta estructura es bastante conveniente para que los ambientes puedan ser **multiplataforma**, en muchas ocasiones una misma imagen se puede utilizar para GNU/Linux, Mac o Windows, ya que tenemos diferentes máquinas virtuales para cada sistema operativo.

Distribuciones de Smalltalk

Smalltalk a secas no es más que una **especificación**, que luego tiene diferentes **implementaciones** de ambientes, que varían según sistema operativo, propósito, o si se distribuye con licencia libre o privativa.

Entre las distribuciones más conocidas se encuentran: Squeak, VA Smalltalk, VisualWorks, Pharo, GNU Smalltalk, Cuis, Amber. Hay muchísimas más.

Cabe destacar que las diferentes implementaciones se suelen tomar ciertas "licencias" y se permiten modificar ciertos aspectos de lenguaje, principalmente extendiendo su sintaxis, es por eso que a veces un programa escrito en una distribución de Smalltalk no puede ser portado fácilmente a otra distribución. No obstante, hay un estándar que se denomina Smalltalk-80 que muchas distribuciones cumplen.

Herramientas básicas: Browser, Inspector, Debugger

Todo ambiente posee al menos 3 herramientas básicas con las que vamos a trabajar gran parte del tiempo: el **Browser**, el **Inspector** y el **Debugger**.

El **Browser** es el lugar en donde podemos, como indica su nombre, navegar por el sistema y ver los diferentes objetos que existen junto con los mensajes que saben responder, y las implementaciones

(métodos) para dichos mensajes. No sólo podemos ver sino que también podemos modificar métodos, agregar y borrar objetos, entre otras opciones.

El **Inspector** es quizá la herramienta más fundamental de un Smalltalk, ya que es la forma de ver y manipular un objeto concreto. Cuando se habla de inspeccionar un objeto, se refiere a abrir una ventana de inspector, donde el objeto inspeccionado es el que queremos. Ya lo veremos más adelante, pero es muy sencilla a nivel visual. Podemos ver la estructura interna del objeto, con sus colaboradores internos, una representación en texto de dichos colaboradores y del propio objeto, y contamos con un panel interactivo en el cual podemos enviarle mensajes y ver sus resultados.

El **Debugger** es otra herramienta fundamental, donde podemos ejecutar paso por paso cualquier colaboración. Lógicamente, en un ambiente puro, cada "paso" no es más que una colaboración. El Debugger es muy útil para entender cómo funciona un programa, y no sólo para encontrar errores (el único propósito que se suele asociar a esta herramienta) sino para construir de a pequeños pasos nuestro programa.

CuisUniversity

Introducción

La distribución de Smalltalk que utilizaremos en este curso se llama CuisUniversity, y es una distribución diseñada específicamente para aprender programación orientada a objetos. Contiene herramientas específicas para ir aprendiendo conceptos de manera incremental, y agregados pedagógicos que sirven para suavizar la curva de aprendizaje.

CuisUniversity está basado en la distribución Cuis Smalltalk, creada en el año 2009 por Juan Vuletich como un fork de Squeak, y que hoy en día cuenta con una fuerte comunidad centrada en Argentina. Es una distribución completamente open source, lo cual es muy favorable para ser utilizada en un contexto académico. Además Cuis tiene como objetivo ser un Smalltalk simple y fácil de entender, con un conjunto mínimo de herramientas pero con el suficiente poder para realizar cualquier tarea de desarrollo.

Estructura de un ambiente CuisUniversity

CuisUniversity puede descargarse desde el siguiente enlace: http://cuisuniversity.org y viene organizado en un archivo .zip para las plataformas más conocidas, que contiene todo lo necesario para empezar. Es un ambiente completamente portable, no se necesita de un software adicional para ejecutarlo.

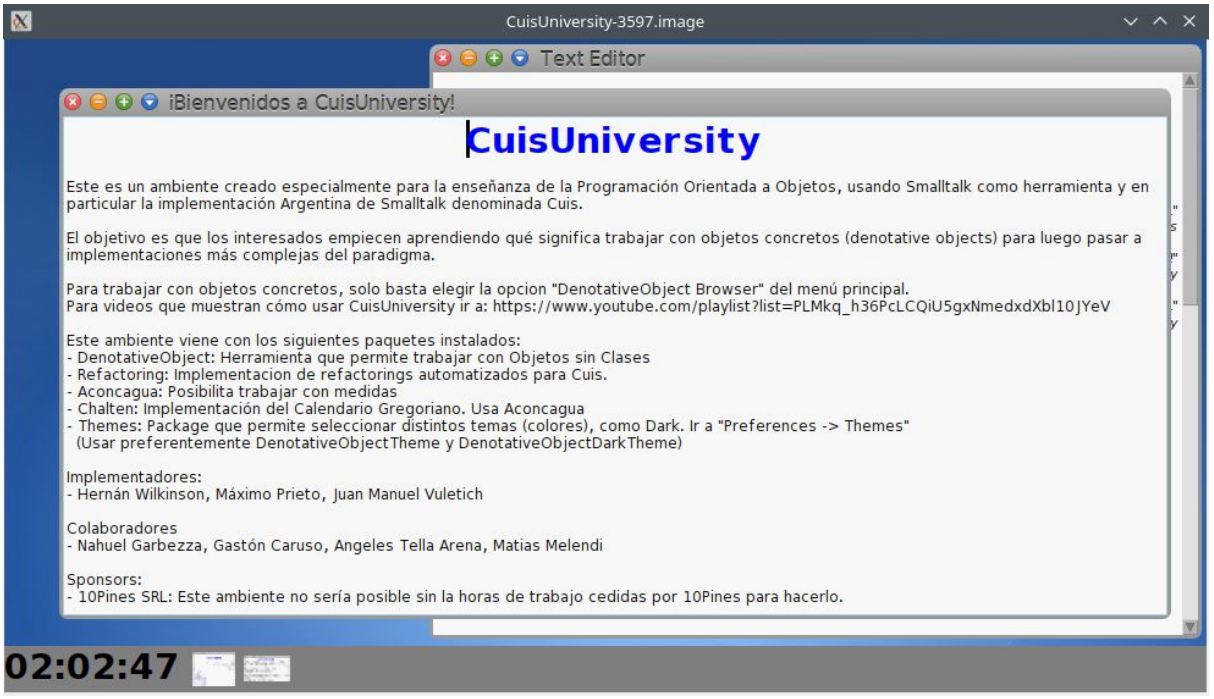
Al descomprimir este paquete, nos encontraremos con una estructura de carpetas y archivos como la siguiente. Veamos para qué sirve cada cosa:

- CoreUpdates: carpeta que contiene archivos con actualizaciones al ambiente, que pueden instalarse bajo demanda y sobre el ambiente ya funcionando.
- Documentation: carpeta que contiene varios documentos que explican el uso de Cuis y algunas de sus principales bibliotecas

- Packages: aquí están algunas de las extensiones que se pueden instalar a Cuis: desde bibliotecas para sonido e imagen, o diferentes herramientas o soporte para programación web. Hay más packages disponibles en Internet, estos son sólo los más conocidos y de los que se conoce su correcto funcionamiento en Cuis
- vm: contiene la máquina virtual.
- CuisUniversity-xxxx.image: El archivo que representa la imagen. el número xxxx indica la versión de Cuis y se condice con el número de versión de los próximos archivos
- CuisUniversity-xxxx.changes: Es un complemento vital para la imagen, contiene el código fuente de los métodos que están guardados en la imagen. Lo que está en el archivo .image es código ya compilado. Si no disponemos de un .changes, lo que veremos va a ser código decompilado que es un poco más difícil de leer.
- CuisUniversity-xxxx.user.changes: Similar al .changes pero contiene sólo los cambios que el usuario haya realizado. Es por ello que inicialmente este archivo está vacío.
- CuisV5.sources: Al igual que el .changes contiene código fuente que se termina asociando a la imagen.
- (varía según el sistema operativo) un ejecutable que se encarga de iniciar el ambiente con la máquina virtual y la imagen. En GNU/Linux, se llama run.sh.

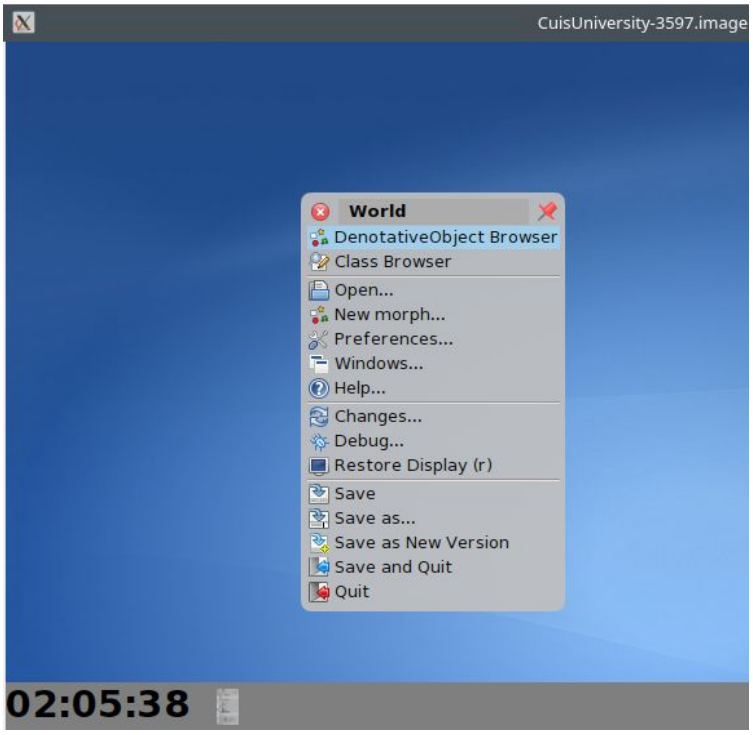
Primeros pasos

Al ejecutar el ambiente, nos encontraremos con una ventana como la siguiente:

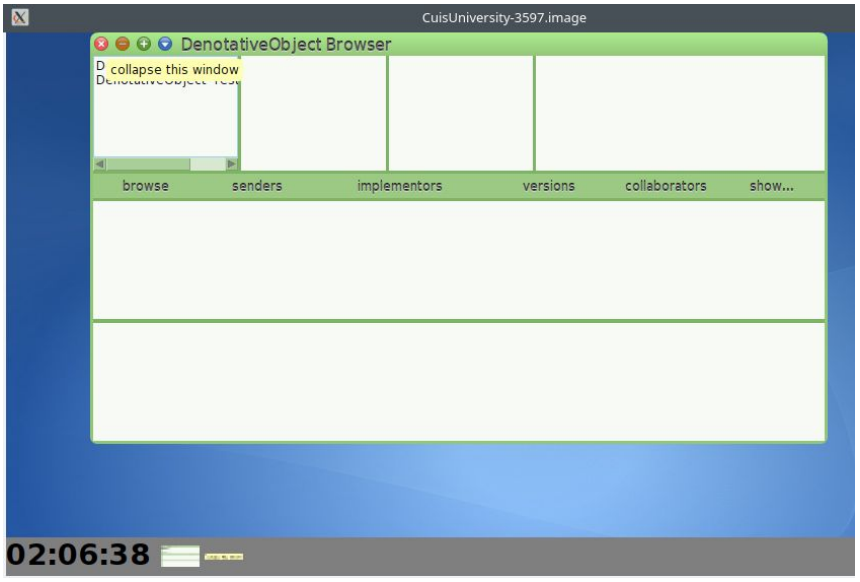


Aquí ya se puede observar cómo el aspecto es como el de un sistema operativo independiente. Las dos ventanas que están abiertas por defecto se pueden minimizar o cerrar, sólo sirven a modo de introducción.

Luego al clickear en el "escritorio" de este ambiente accedemos al menú World a través del cual podemos visitar las diferentes herramientas que existen:

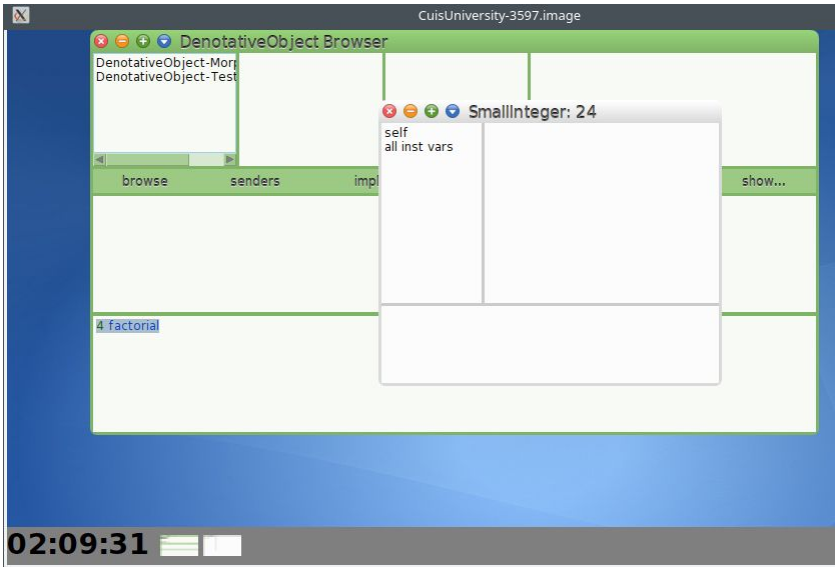


Vamos a comenzar con la primera opción, el DenotativeObject Browser:



El DenotativeObject Browser es un agregado de CuisUniversity y no es más que un Browser, especializado en objetos concretos, con los cuales vamos a arrancar nuestro curso.

Probemos nuestra primera colaboración: en el panel de más abajo, escribamos `4 factorial .` Luego, presionamos Ctrl+i o desde el menú contextual, la opción Inspect it (inspeccionar). Esto debería abrir un inspector sobre el resultado de esta colaboración (enviarle el mensaje `factorial` al objeto 4):



¡Este es un gran paso! Aquí ya podemos ver en acción el feedback inmediato y la manipulación directa. Antes de avanzar, veamos las opciones posibles para ejecutar una colaboración:

- Do it (ejecutarlo): Esto sólo envía el mensaje y NO muestra el resultado. Se activa con Ctrl+d
- Print it (imprimirlo): Esto envía el mensaje e imprime una representación textual del resultado a la derecha de la expresión escrita. Es una opción un poco más limitada que el inspector pero un poco más rápida porque no se abre una ventana extra. Se activa con Ctrl+p.
- Inspect it (inspeccionarlo): Esto es lo que hicimos en el paso previo: se envía el mensaje y se abre un inspector con el resultado. Es bastante poderoso ya que podemos visualizar bastante lo ocurrido y continuar interactuando si es necesario. Se activa con Ctrl+i
- Debug it (debuggearlo): Abre un Debugger en el cual el primer paso (no ejecutado aún) es evaluar la expresión seleccionada.

En el panel inferior del DenotativeObject Browser se puede escribir más de una expresión, siempre y cuando estén separadas por un punto. Esta es la forma que Smalltalk tiene para separar "statements".

Más ejemplos

Esta lista de videos en YouTube tiene más tutoriales explicando las diferentes herramientas de CuisUniversity. La versión utilizada en los videos es bastante menor a la actual, así que puede haber algunas diferencias visuales, pero conceptualmente no hubo cambios importantes, así que deberían ser igual de útiles.

CuisUniversity:
https://www.youtube.com/watch?v=Ke8YAiz2OO8&list=PLMkq_h36PcLCQiU5gxNmedxdXbl10JYeV