Common Lisp — Caso de uso real

Ana Secchi - Teoría de Lenguaje - FIUBA

Crash Bandicoot

Comienzos

Crash Bandicoot fue originalmente creado en 1996 para Play Station, por la desarrolladora de videojuegos Naughty Dog, fundada en 1984 por Andy Gavin y Jason Rubin. [1]

Gavin aprende Lisp por primera vez en la facultad, preparándose para un curso de Inteligencia Artificial y eventualmente se convirte en un fanático de Lisp. [2] En aquella época, una de las cosas interesantes de Lisp es que se trataba de un lenguaje relativamente fácil de parsear, interpretar y compilar.

1. Contexto

El código de controles de objetos, típicamente se ejecuta solo unas veces por frame. Para este tipo de implementación de código, Gavin explica que los requisitos más importantes son la velocidad de implementación, flexibilidad y la facilidad de poder modificarlo más adelante. Esto se debe a que todos los juegos tratan de tener un buen gameplay, y esto solamente es posible de lograr cuando uno escribre y reescribe una y otra vez los objetos a través del código. Como cualquier otra herramienta, Gavin dice que un lenguaje de programación tiene que ser el adecuado para cierto trabajo.

El lenguaje estándar de los últimos años era C. Sin embargo, muchos programadores encuentran que C tiene sintaxis inconsistente, un sistema de macros débil y una barrera importante entre tiempo de ejecución y tiempo de compilación. Para los que conocen muy bien C, puede llegar a ser conveniente ya que es muy bueno en expresiones y control de flujo básico, pero en cuanto a estructuras de datos más complejas, se requiere de un esfuerzo mucho mayor y con C es muy complejo transferir un tipo de dato a algún otro similar. Otra debilidad es el manejo de memoria. Desarrolladores se ven forzados a crear y destruir nuevos objetos de todo tipo, lo cual se vuelve un problema cuando en medio de un juego hay objetos que se crean y destruyen constantemente.

Ninguno de estos problemas estaban presentes en Lisp, que era uno de los lenguajes más flexibles para lograr el objetivo. Respaldado por la sintaxis consistente y por el sistema de macros que tenía, Lisp era extremadamente fácil de actualizar, personalizar, y expandir; todo esto era posible sin tener que pelear contra la estructura básica del lenguaje. Lisp sigue una simple regla: todo el código es una lista. Esta simple regla acaba con la ambigüedad en cuanto a sintaxis. Como las computadoras tienen un dilema en cuanto a la ambigüedad, programas que escriben otros programas eran mucho más fáciles de lograr en Lisp.

2. GOOL: Game Object Oriented Lisp

Entonces, para codear Crash Bandicoot, Gavin junto a su equipo de programadores, diseñaron GOOL [3]. Es el lenguaje usado para Crash 1, Crash 2 y Crash 3 y es un dialecto compilado de Common Lisp personalizado, especialmente desarrollado para la programación de objetos animados de juego interactivos.

Common Lisp provee un ambiente ideal para escribir compiladores porque ya contaban desde el comienzo con el recolector de basura, listas, árboles, y macros. GOOL como lenguaje, toma prestada la sintaxis y formas básica de Common Lisp: expresiones, aritmétrica, y operadores de control de flujo. GOOL varía en algunos aspectos por razones de simplicidad, pero básicamente puede ser entendido por cualquiera que haya codeado en Lisp alguna vez. GOOL cuenta con 56 primitivas y 420 macros que soportan el control de flujo y operaciones relacionadas a objetos de juego. Tener un lenguaje personalizado, cuyas primitivas y constructores realizan a una tarea general (en este caso programación de objetos de juego), hace que sea fácil escribir código limpio y descriptivo mucho más rápido. GOOL hace posible crear un prototipo de una nueva criatura u objeto, en tan solo 10 minutos. Nuevas cosas pueden ser probadas y elaboradas o descartadas rápidamente. Si finalmente el objeto no resultaba, simplemente se remueve del juego en segundos, sin dejar rastros difíciles de borrar.

Figura 1: The Great Gate

En este nivel Crash recorre plataformas de madera elevadas mientras escala un portón gigante para llegar a destino. Avanzar en altura requiere que salte cajas de hierro y plataformas de madera, e ir girándolas para acceder. El camino está lleno de trampas y enemigos, como pilares con púas que perforan al objeto y plataformas especiales que arrojan llamas. Además, Crash se encontrará con un terreno resbaladizo, lo que hará que resbale si permanece allí demasiado tiempo. Todo esto era posible gracias al diseño de GOOL.

"The secret to Crash's success was its Art.
And the secret to its Art was its Programming."

- Jason Rubin

Referencias

[1] Crash Bandicoot @

https://en.wikipedia.org/wiki/Crash_Bandicoot

[2] Lispings ala John McCarthy, Andy Gavin @

https://all-things-andy-gavin.com/2011/10/25/lispings-ala-john-mccarthy/

[3] Making Crash Bandicoot – GOOL – part 9, Andy Gavin @ https://all-things-andy-gavin.com/2011/03/12/making-crash-bandicoot-gool-part-9/