Universidad de Castilla-La Mancha
Escuela Superior De Informática

Memoria de Prácticas

Blendify: Lenguaje para la definición de simulaciones físicas

Autores:
Alberto Aranda García
Cristian Gómez Portes
Daniel Pozo Romero
Eduardo Sánchez López

Fecha:
7 de octubre de 2017

Índice

1. Presentación del problema

Dentro de la enseñanza de Física, la visualización de los problemas es una herramienta muy importante para que los estudiantes comprendan mejor los conceptos teóricos existentes tras ellos. En este contexto, la simulación es una opción que lleva disponible durante años, pero que está siendo muy infrautilizada.

Esto tiene una fácil explicación: la mayoría de programas de generación de simulaciones físicas imponen interfaces gráficas que requieren una inversión de tiempo grande por parte del usuario: partiendo de un enunciado textual de un problema, el usuario, que puede ser tanto un profesor como un estudiante de física, se ve obligado a traducirlo manualmente a un formato gráfico, lidiando muchas veces con interfaces de usuario tediosas que hacen perder mucho tiempo, tanto para aprenderlas, como por las operaciones repetitivas que deben realizarse.

Lo ideal sería que, a partir de un enunciado de un problema, fuese posible la generación automática de su simulación. Dejando de lado las dificultades que el Procesado de Lenguaje Natural supone (aún dentro de un dominio tan acotado), una primera aproximación puede ser la creación de un lenguaje con un poder expresivo similar al de los enunciados de problemas, al que sea lo más sencillo posible traducir (para un humano) dichos enunciados escritos originalmente en lenguaje natural.

Como plataforma de simulaciones físicas, vamos a considerar Blender, ya que incorpora, entre muchas otras funcionalidades, un motor de físicas muy potente, y, tantoo más importante, una API para Python que expone prácticamente todas las funcionalidades del programa.

Para nuestra práctica, por tanto, consideramos la definición de un lenguaje que permita escribir simulaciones físicas, y la construcción de un procesador de lenguajes que lo traduzca a código Python que use la API de Blender, que posteriormente podrá ser interpretado por este programa para generar una simulación. Nuestro objetivo principal, pues, es simplificar al máximo posible la generación de simulaciones que de otra forma serían muy tediosas de realizar.

2. Descripción del lenguaje

Como se ha dicho anteriormente, nuestro lenguaje, llamado Blendify, deberá tener el mismo poder expresivo los enunciados de problemas, y posiblemente, permitir definir ciertos hechos sobre el contexto que se dan por supuesto (tipo de problema, dimensionalidad, etc.). Si analizamos la estructura de los enunciados, nos encontra-

mos con que, primero, se nos da información sobre los elementos que intervienen en el problema (cubo, esfera, polea, rampa, electrón, planeta...) y sus parámetros (masa, velocidad, fuerzas que actúan sobre este...); posteriormente, se suele dar una condición que podemos considerar como de parada de la simulación, y una pregunta (que podemos felizmente obviar) sobre algún parámetro de un elemento en determinado momento. Nuestro lenguaje, pues, deberá permitir declarar diferentes objetos y sus correspondientes atributos, así como ciertos parámetros globales que afecten a toda la simulación, alguna condición de parada, y posiblemente, información sobre qué quiere hacer el usuario con la simulación (ejecutarla, generar una animación, etc.).

2.1. Tabla de Tokens

Tokens	**Lexeme**	**Pattern**
Assign	=	=
Comma	,	,
Open Bracket	{	{
Closed Bracket	}	}
Case	Case	Case
scene	scene	scene
start_simulation	start_simulation	start_simulation
condition	AND |OR	AND |OR
Open parentheses	(	(
Closed parentheses	)	)
Type Figure	static, dynamic	static, dynamic
Type Value	position, rotation, scale weight, speed	position |rotation |scale |weight |speed
From Figure	Cube, Sphere, Cone Cylinder, Force_field, Ramp	Cube |Sphere |Cone Cylinder |Force_field |Ramp
Coordinates	(1.2, 1.2, 1.2)	'(' real ',' real ',' real ')'
Alphabetic	a ,..., z , A ,..., Z	lower_case |upper_case
lower_case	a ,..., z	[a-z]
upper_case	A ,..., Z	[A-Z]
Real	23.21	digit + ['.' digit +)]
Digit	0 ,..., 9	1 |2 |3 |4 |5 6 |7 |8 |9 |0

Cuadro 1: Tabla de tokens.

2.2. EBNF

program ::= **begin** id_program body_program **end**

body_program ::= **declaration** body_declaration **scene** body_scene **action** body_action

body_declaration ::= '{' { **(static** attribute_declaration '=' value_static) |(**dynamic** attribute_declaration '=' value_dynamic)} '}'

attribute_declaration ::= type_figure id_attribute

type_figure ::= **Cube** |**Sphere** |**Cone** |**Cylinder** |**Force_field** |**Ramp** |**Plane**

body_scene ::= '{' {(attribute_case '=' goal)} '}'

attribute_case := **Case** id_case

goal::= gplane |gspeed |gcollision |goal2

goal2 ::= '(' goal { condition goal } ')'

body_action ::= '{' **start_simulation** '}'

value_static ::= [position] [rotation] [scale]

value_dynamic ::= [position] [rotation] [scale] [weight] [speed]

id_attribute ::= alphabetic {alphanumeric}

id_program ::= alphabetic {alphanumeric}

position ::= **position** coordinates

rotation ::= **rotation** coordinates

scale ::= **scale** coordinates

weight ::= **weight** real

speed ::= **speed** coordinates

coordinates ::= '(' real ',' real ',' real ')'

condition ::= **AND** |**OR**

alphabetic ::= lower_case |upper_case

lower_case ::= [a-z]

upper_case ::=[A-Z]

alphanumeric ::= alphabetic |digit

*rea*l ::= digit + [('.' digit +)]