

Release Notes of Fables IME

Feature Version: 1.0.04

Build id: M20080221-1800,

Compiler Build: 20080402.1922

Date: 04.03.2008

Release Notes.....	1
Grammar changes	2
New Keywords	2
Other changes	2
MassGUI changes	3
Standard Library extensions	4
Example Code.....	5

Grammar changes

New Keywords

1. `void` Is a new syntax element for the empty statement (“do nothing”).
2. `param` Is a new syntax element for notating model parameters. For complex models, the parameter list of the `startUp()` section would be hard to keep up-to-date. It’s an alternate way, the parameters can be declared either by enumerating them in the parameter list, or with the use of the new `param` keyword (or by using both ways).
3. `create` A new operator has been declared to clarify the creation of agents. From now on, the `new` operator always returns with the created agent, and the `create` operator just creates a new instance. This means that the new operator cannot be used in places where no return type is allowed (in the `startUp()` sections and event definition statements in the schedules).

Other changes

4. From now it is possible to explicitly notify the compiler that a constant should be generated as a function with the use of an empty parameter list (“()”):

```
const = 5; // is a constant
fnc() = 5; // is always generated as a function
```

5. Variable assignments can have initializations, and more than one variable declaration per definition is allowed:

```
var a; // Old form
var a := 0, b := 3 + c; // New form
```

6. For-each statements now accept a single statement or a block of statements:

```
for each i in [1..5] do println(i); // Old format
for each i in [1..5] do { // New format
    println(i); println(2*i);
};
```

7. Function expressions have to note their parameters exactly. Arbitrary usage of parentheses now counts as a compilation error. These structures caused many unambiguous problems.

```
f1(a, b) = a+b;
f2(c) = 5;

startUp() {

// f1 f2 0 0; // Old format: many problems, compiler parsed the
//parameters
// greedily. It was searching for the functions of f2(0, 0)
// and f1(f2(0, 0))
// From now these definitions are illegal!

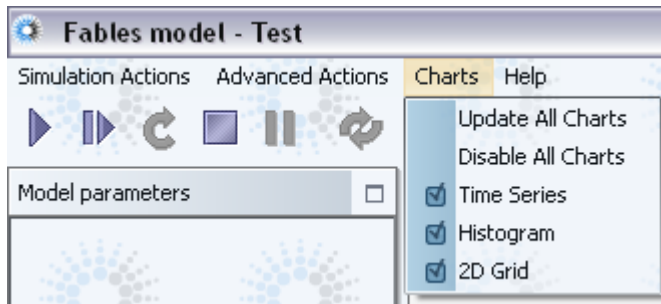
f1( f2(0), 0 ); // New format

};
```

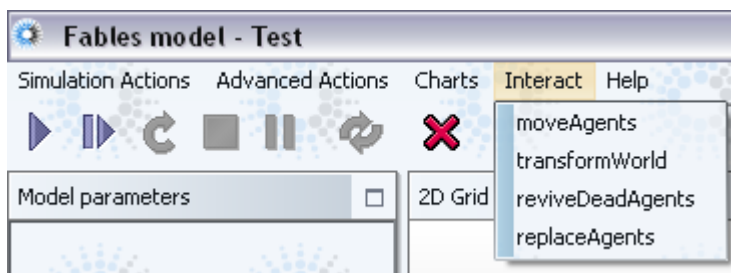
- The only exceptions from the above rule are the operators. Using the form of $* \{1, 2, 5\}$ is still a legal usage.

MassGUI changes

- Charts can be turned off and on separately or altogether in a new menu to be able to speed up the simulations during the unimportant phases:



- User interaction is now possible during the simulation runs. Each function which has no parameters can be executed anytime during the simulation when the simulation is paused. If the function has any return type, its value is printed on the Console.



Standard Library extensions

11. **Decrease:** The function returns the value of its first parameter decreased by 1 or by the second parameter if it has two parameters.

SYNTAX: dec(<Number>)
 dec(<Number>, <Number>)

12. **Increase:** The function returns the value of its first parameter increased by 1 or by the second parameter if it has two parameters.

SYNTAX: inc(<Number>)
 inc(<Number>, <Number>)

13. **Greatest common divisor:** The function returns the greatest common divisor of the integers given by the parameter(s).

SYNTAX: gcd(<Number>, <Number>)
 gcd(<Collection of Numbers>)

14. **Least Common Multiple:** The function returns the least common multiple of the integers given by the parameter(s).

SYNTAX: lcm(<Number>, <Number>)
 lcm(<Collection of Numbers>)

15. **Probability:** Returns true or false with given probability. When the function has only one parameter then it returns true if a random number chosen from the [0,1) interval is smaller than the parameter. In case of two parameters the first parameter determines the random generator used by the function. When there are three parameters the function returns true if the first parameter is greater than a random number chosen from the [second parameter, third parameter) interval. In case of four parameters the first parameter determines the random number generator.

SYNTAX: prob(<Number>)
 prob(<GeneratorName>, <Number>)
 prob(<Number>, <Number>, <Number>)
 prob(<GeneratorName>, <Number>, <Number>, <Number>)

16. **Uniform Sample:** Returns a uniform sample of a given collection with the given length (which has a default value of 1). An *exclude set* can be passed to the function, and it guarantees that the result not contain those elements.

SYNTAX: uniformSample(<Collection of Numbers>)
 uniformSample(<Collection of Numbers>, <Number>)
 uniformSample(<Collection of Numbers>,
 <Collection of Numbers>, <Number>)

Example Code

```
model StdLibExtensions {  
    startUp() {  
        println( "---- DEC ----" );  
        println( dec(2) ); // 1  
        println( dec(2, 3) ); // -1  
        println( dec(2.5) ); // 1.5  
        println( dec(2, 3.1) ); // -1.1  
        println( dec(23.1, 3) ); // 20.1  
        println( dec(1.1, 2.1) ); // -1.0  
  
        println( "---- INC ----" );  
        println( inc(2) ); // 3  
        println( inc(2, 3) ); // 5  
        println( inc(2.5) ); // 3.5  
        println( inc(2, 3.1) ); // 5.1  
        println( inc(23.1, 3) ); // 26.1  
        println( inc(1.1, 2.1) ); // 3.2  
  
        println( "---- GCD ----" );  
        println( gcd(42, 56) ); // 14  
        println( gcd( [12, 18] ) ); // 6  
        println( gcd( {-4, 14} ) ); // 2  
        println( gcd(-5, 0) ); // 5  
  
        println( "---- LCM ----" );  
        println( lcm(4, 6) ); // 12  
        println( lcm( [21, 6] ) ); // 42  
        println( lcm( {8, 9, 21} ) ); // 504  
        println( lcm(3, 4.0) ); // 12 [strict conversion to int]  
  
        //-----  
        seed(10); // To make sure randoms are working correctly  
        //-----  
  
        println( "---- PROB ----" );  
        println( prob(0.5) ); // false  
        println( prob(10, 1, 100) ); // false  
        addUniform( "A", 1112 );  
        println( prob("A", 0.5) ); // false  
        println( prob("A", 10, 1, 100) ); // false  
  
        println("---- UNIFORM_SAMPLE ----");  
  
        println("uniformSample({1,2,3}): " ++ uniformSample({1,2,3}));  
        // uniformSample({1,2,3}): 3  
  
        println("uniformSample({1.1,2,3}, 2): " ++ uniformSample({1,2,3},2));  
        // uniformSample({1.1,2,3}, 2): { 2, 1 }  
  
        println("uniformSample({1,2,3,4,5}, {2,3}, 2): " ++  
            uniformSample({1,2,3,4,5},{2,3},2));  
        // uniformSample({1,2,3,4,5}, {2,3}, 2): { 1, 4 }  
  
        println( "uniformSample([1,2,3], 2): " ++ uniformSample([1,2,3],2));  
        // uniformSample([1,2,3], 2): [1, 3]  
    }  
};
```