

Object Oriented Modeling and Simulation with OOSimL

Random Numbers

Dr. José M. Garrido C.

Department of Computer Science
College of Computing and Software Engineering
Kennesaw State University

Spring 2017

Stochastic Models

- Random numbers are used to model uncertainty in simulation stochastic models.
- These models use probability distributions for the generation of random numbers.

OOSimL provides several types of random number generators that use different probability distributions.

Common Probability Distributions in OOSimL

- *Randint*
- *NegExp*
- *Poisson*
- *Normal*
- *Uniform*
- *Binomial*
- *Bernoulli*
- *Erlang*
- *Gamma*
- *Geometric*
- *HyperGeometric*
- *Triangular*
- *Weibull*

Random Numbers with Randint

- The *Randint* random number generator provides the most basic type of random number generation, a uniform distribution with numbers between 0.0 and 1.0.
- The **create** statement is used to create and initialize a random number generator object.

The following lines declare and create a random number generator object, referenced by *ran_gen1*, with title “random val” and 7 as the value of the seed.

```
define ran_gen1 of class Randint // random generator
...
create ran_gen1 of class Randint using
    ``random val'', 7
```

Generating Random Numbers

The **generate random** statement generates a random number from the specified random number generator object. The general form of this statement is:

```
generate random value { variable }  
    from { ref_var }
```

The following lines generate a random number using the generator object *ran_gen1* and stores this number in variable *ran_value*.

```
define ran_value of type double  
...  
generate random value ran_value from ran_gen1
```

Random Numbers with Negative Exponential

- The *NegExp* random number generator provides random numbers using a negative exponential distribution.
- The **create** statement is used to create and initialize a random number generator object.
- The arguments supplied are the name (of type **string**) assigned to the random number generator, the mean value of the samples (of type **double**), and the seed (of type **long**, optional argument).

Example Negative Exponential

The following lines declare and create a random number generator object, referenced by *ran_gen2*, with title “random time” and *mtime* as the mean value.

```
define ran_gen2 of class NegExp // random generator
...
create ran_gen2 of class NegExp using
    ``random time'', mtime
```

Generating Negative Exponential

the following lines generate a random number from the generator object *ran_gen2* defined above, and stores the random number in variable *ran_value*.

```
define ran_value of type double
...
generate random value ran_value from ran_gen2
```


Random Numbers with Normal Distribution

The *Normal* random number generator provides random numbers using a normal distribution.

Creating Normal Objects

The **create** statement is used to create and initialize a random number generator object. The arguments to be supplied are:

- the name (of type **string**) to be assigned to the random number generator
- the mean value of the samples (of type **double**)
- the value of the standard deviation (of type **double**)
- the seed (of type **long**, optional argument).

Example with Normal Distribution

the following lines declare and create a random number generator object *norm_gen* with title “Service time”, mean of value 12.5 and standard deviation of value 2.3.

```
define norm_gen of class Normal
...
create norm_gen of class Normal using
    ``Service time'', 12.5, 2.3
```

Generating Normal Random Numbers

The following lines generate a random number from the generator object *norm_gen* and stores the random number in variable *ran_value*.

```
define ran_value of type double
...
generate random value ran_value from norm_gen
```

Random Numbers with Triangular Distribution

The *Triangular* random number generator provides random numbers using a Triangular distribution, which can be a good approximation to work carried out by persons or machines.

Creating Triangular Objects

The **create** statement is used to create and initialize a random number generator object of class *Triangular*. The arguments to be supplied are:

- the name (of type **string**) to be assigned to the random number generator
- the minimum value of the samples (of type **double**)
- the most likely value (of type **double**)
- the maximum value of the samples (of type **double**)
- the seed (of type **long**, optional argument).

Example with Triangular Distribution

The following lines declare and create a random number generator object *triang_gen* with title “Service time”, minimum value 1.25, the most likely value of 1.75, and maximum value of 2.35.

```
define triang_gen of class Triangular
...
create traing_gen of class Triangular using
    ``Service time'', 1.25, 1,75, 2.35
```

Generating Triangular Random Numbers

The following lines generate a random number from the generator object *triang_gen* and stores the random number in variable *ran_value*.

```
define ran_value of type double
...
generate random value ran_value from triang_gen
```

Random Numbers with Uniform Distribution

The *Uniform* random number generator provides random numbers using a Uniform distribution.

Creating Uniform Objects

The **create** statement is used to create and initialize a random number generator object of class *Uniform*. The arguments to be supplied are:

- the name (of type **string**) to be assigned to the random number generator
- the minimum value of the samples (of type **double**)
- the maximum value of the samples (of type **double**)
- the seed (of type **long**, optional argument).

Example with Uniform Distribution

The following lines declare and create a random number generator object *unif_gen* with title “clean interval”, minimum value 1.25, and maximum value of 2.35.

```
define unif_gen of class Uniform
...
create unif_gen of class Uniform using
    ``clean interval'', 1.25, 2.35
```


Generating Uniform Random Numbers

The following lines generate a random number from the generator object *unif_gen* and stores the random number in variable *cl_value*.

```
define cl_value of type double  
...  
generate random value cl_value from unif_gen
```