Implementing UNU.RAN on WINDOWS systems

UNU.RAN is a collection of algorithms for generating non-uniform pseudorandom variate as a library of C functions, released under the GNU Public License (GPL). It provides an consistent and flexible interface to universal algorithm as well as to generators for standard distribution. It is applicable for continuous and discrete, univariate and multivariate distributions and for (re)sampling from empirical distributions.

The present guide present a short introduction to the following topics :

- Building C-language applications with calls to the UNU.RAN functions using a precompiled static library (unuran-0.5.0.lib) in Microsoft .NET framework

- Building Visual Basic applications that implement the UNU.RAN string-api interface using a dynamically linked library (unuran-0.5.0.dll).

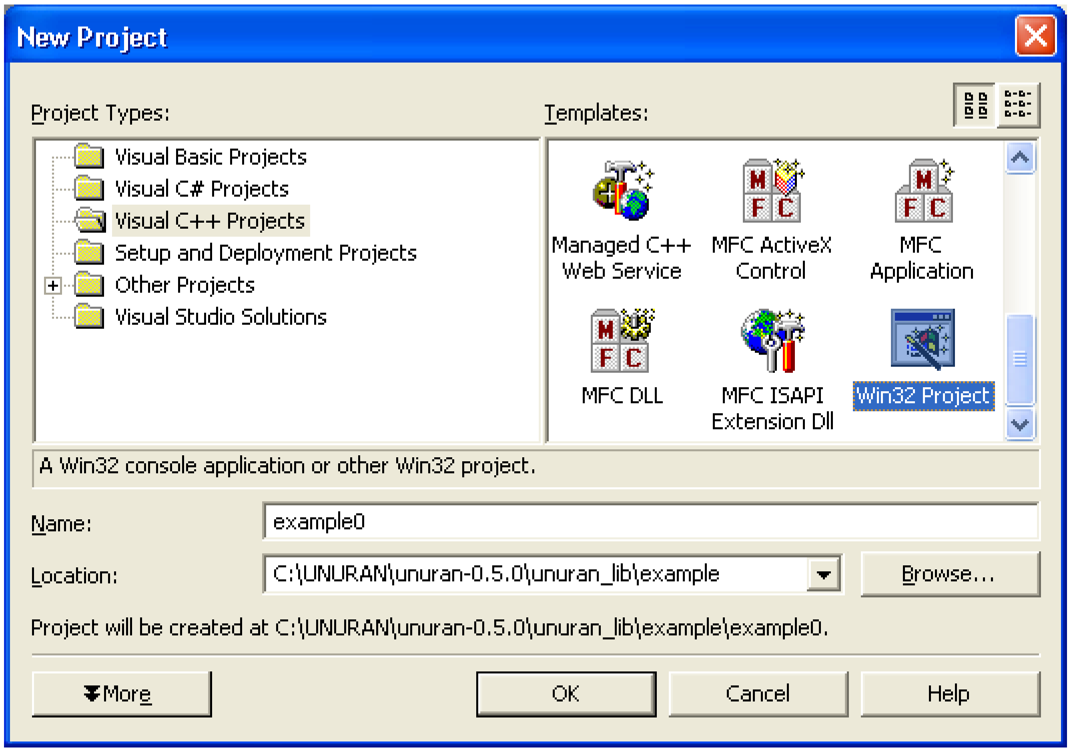
All VBA (Visual Basic for Applications) compatible programs like Excel, PowerPoint, CorelDraw etc. can thus also implement the UNU.RAN random number generators via the dynamically linked library.

The current windows distribution package unuran-0.5.0-win.zip containing the precompiled static and dynamic libraries along with short example programs can be downloaded from http://statistik.wu-wien.ac.at/unuran/download.html

Please keep in mind that UNU.RAN is work in progress. Thus although UNU.RAN is released under the GNU Public License (GPL) we ask all user of our library to send us an email unuran@statistik.wu-wien.ac.at . We are interested in any kind of feedback. This includes besides BUG reports also applications of our software.

Building C-language applications with calls to the UNU.RAN functions using the precompiled static library (unuran-0.5.0.lib) in Microsoft .NET framework

In the Microsoft .NET framework create a new project :



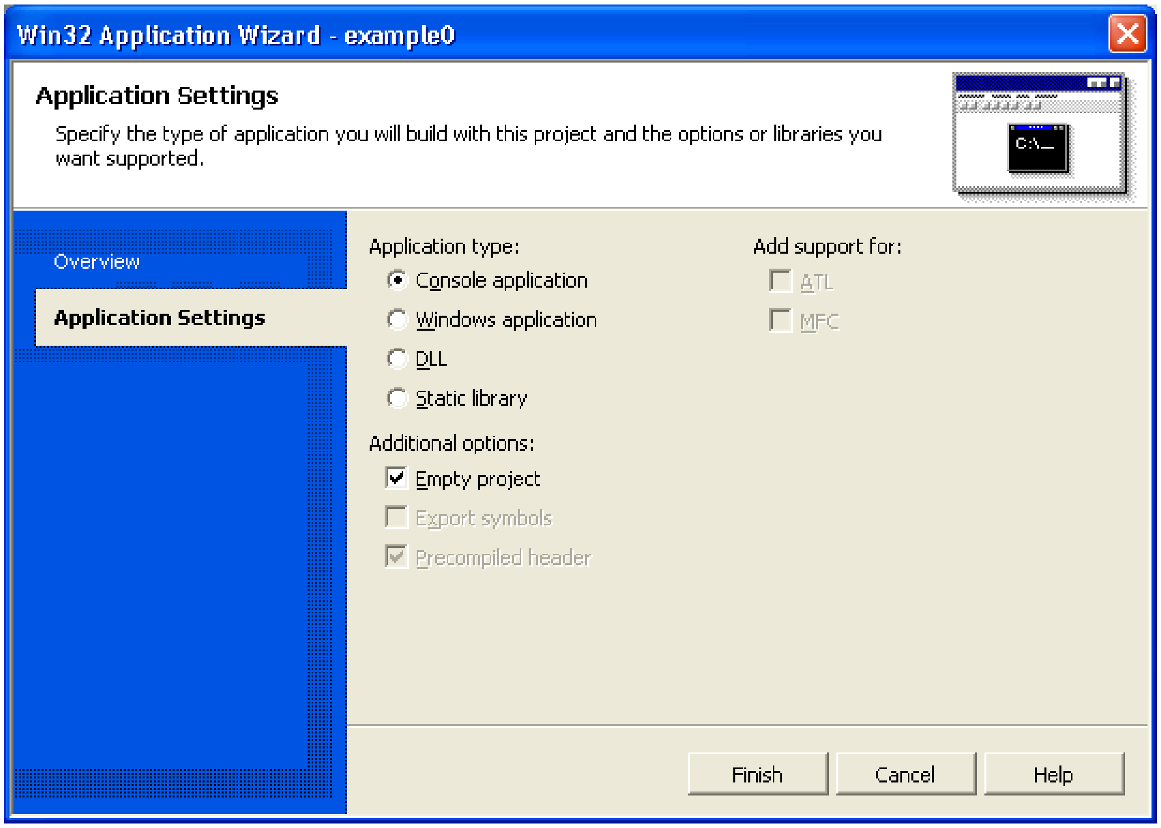
A Win32 console application or other Win32 project.

Name: example0

Location: C:\UNURAN\unuran-0.5.0\unuran_lib\example

Project will be created at C:\UNURAN\unuran-0.5.0\unuran_lib\example\example0.

In the present example we wish to build a console application starting with an empty project :



Application Settings

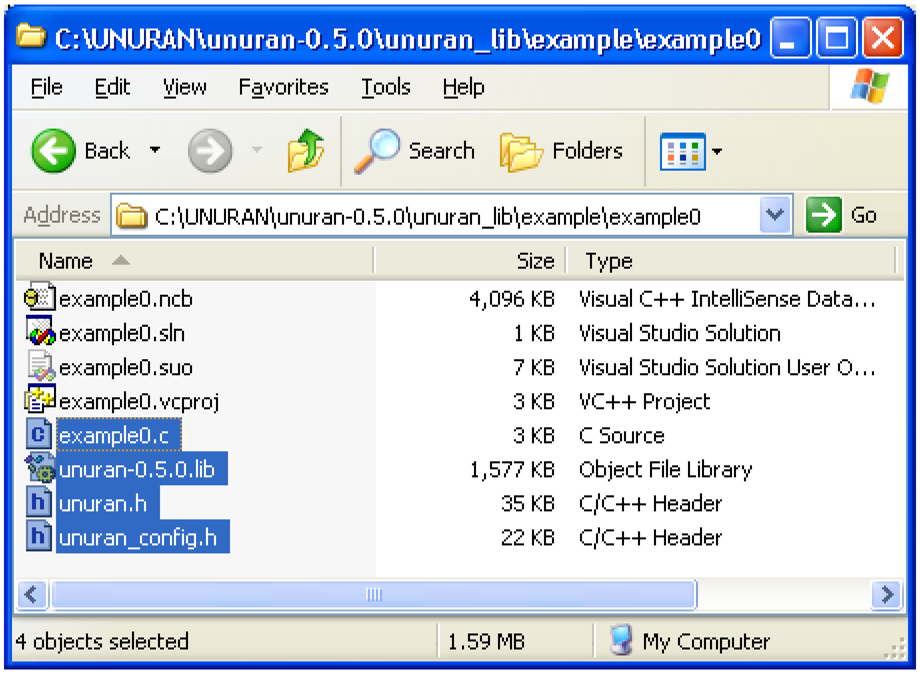
Specify the type of application you will build with this project and the options or libraries you want supported.

Application type:
- [✓ Console application]
- Windows application
- DLL
- Static library

Additional options:
- [✓ Empty project]
- Export symbols
- [✓ Precompiled header]

Add support for:
- ATL
- MFC

We now copy the necessary files example0.c, unuran.h, unuran_config.h and unuran-0.5.0.lib into our project folder :

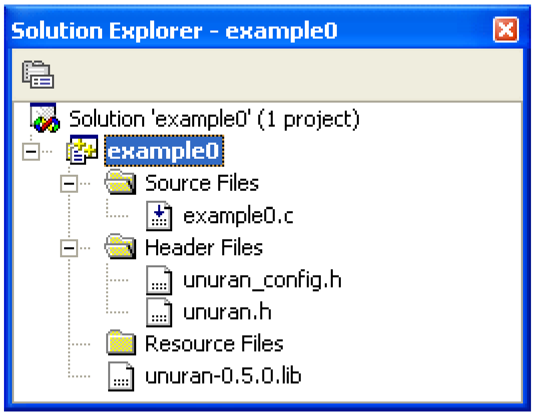


C:\UNURAN\unuran-0.5.0\unuran_lib\example\example0

Name	Size	Type
example0.ncb	4,096 KB	Visual C++ IntelliSense Data...
example0.sln	1 KB	Visual Studio Solution
example0.suo	7 KB	Visual Studio Solution User O...
example0.vcproj	3 KB	VC++ Project
example0.c	3 KB	C Source
unuran-0.5.0.lib	1,577 KB	Object File Library
unuran.h	35 KB	C/C++ Header
unuran_config.h	22 KB	C/C++ Header

4 objects selected 1.59 MB My Computer

And adding these “existing items” in the solution explorer

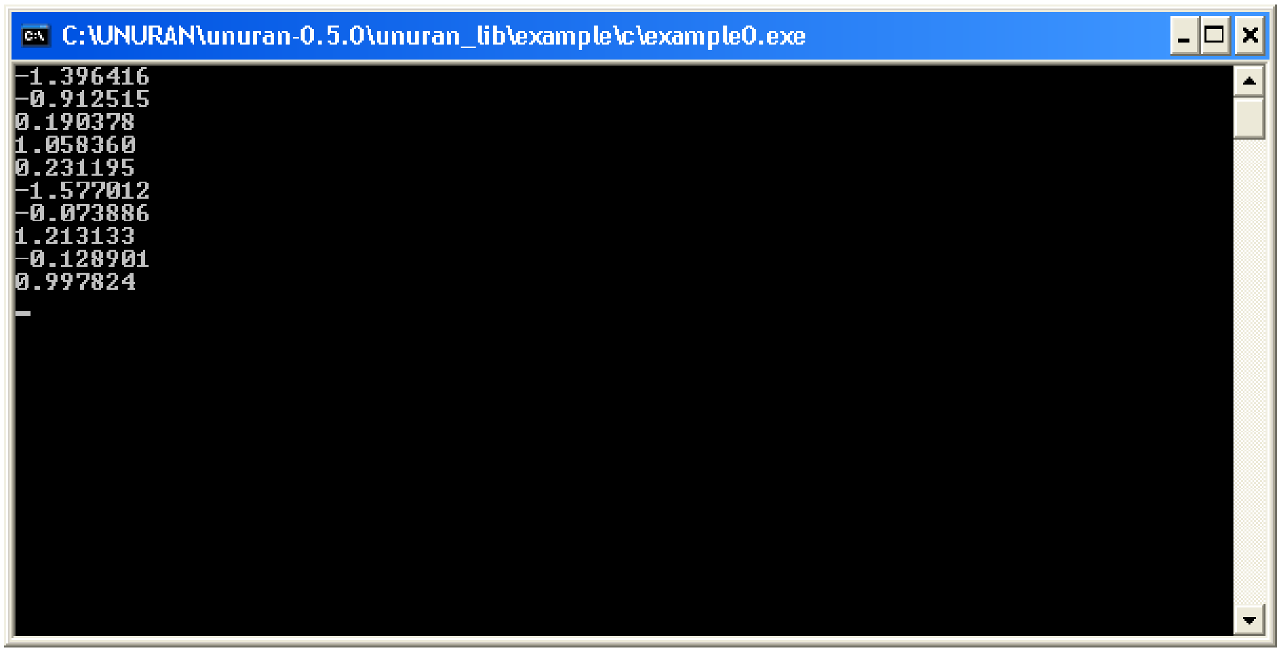


```
Solution 'example0' (1 project)
example0
    Source Files
        example0.c
    Header Files
        unuran_config.h
        unuran.h
    Resource Files
    unuran-0.5.0.lib
```

Inspecting the content of our example0.c file :

```c
/* ------------------------------------------------------------- */
/* File: example0.c                                              */
/* ------------------------------------------------------------- */

/* Include UNURAN header file.                                   */
#include "unuran.h"

int main()
{
  int    i;     /* loop variable                                 */
  double x;     /* will hold the random number                   */

  /* Declare the three UNURAN objects.                           */
  UNUR_DISTR *distr;    /* distribution object                   */
  UNUR_PAR   *par;      /* parameter object                      */
  UNUR_GEN   *gen;      /* generator object                      */

  /* Use a predefined standard distribution:                     */
  /*   Gaussian with mean zero and standard deviation 1.         */
  /*   Since this is the standard form of the distribution,      */
  /*   we need not give these parameters.                        */
  distr = unur_distr_normal(NULL, 0);

  /* Use method AUTO:                                            */
  /*   Let UNURAN select a suitable method for you.              */
  par = unur_auto_new(distr);

  /* Now you can change some of the default settings for the     */
  /* parameters of the chosen method. We don't do it here.       */

  /* Create the generator object.                                */
  gen = unur_init(par);

  /* Notice that this call has also destroyed the parameter      */
  /* object `par' as a side effect.                              */
  /* It is important to check if the creation of the generator   */
  /* object was successful. Otherwise `gen' is the NULL pointer  */
  /* and would cause a segmentation fault if used for sampling.  */
  if (gen == NULL) {
     fprintf(stderr, "ERROR: cannot create generator object\n");
     exit (EXIT_FAILURE);
  }

  /* It is possible to reuse the distribution object to create   */
  /* another generator object. If you do not need it any more,   */
  /* it should be destroyed to free memory.                      */
  unur_distr_free(distr);

  /* Now you can use the generator object `gen' to sample from   */
  /* the standard Gaussian distribution.                         */
  for (i=0; i<10; i++) {
    x = unur_sample_cont(gen);
    printf("%f\n",x);
  }

  /* When you do not need the generator object any more, you     */
  /* can destroy it.                                             */
  unur_free(gen);
  getchar(); /* pausing ... */
  exit (EXIT_SUCCESS);
} /* end of main() */
```

That’s it – we can now build and run our example :



```
-1.396416
-0.912515
0.190378
1.058360
0.231195
-1.577012
-0.073886
1.213133
-0.128901
0.997824
```

The static library unuran-0.5.0.lib contains all UNU.RAN functions and methods as found in the documentation (http://statistik.wu-wien.ac.at/unuran/doc.html)

Building Visual Basic applications that implement the UNU.RAN string-api interface using the dynamically linked library (unuran-0.5.0.dll).

The dynamic link library (unuran-0.5.0.dll) is a wrapper library to the UNU.RAN String-API

The following functions are available

- unuran_create (ByVal API_String As String) As Long

  Providing an API_String, this function returns a handle to the unuran-object.
  This handle is a positive integer in case of successful creation of the generator object and is used as parameter in the following calls to unuran_sample and unuran_destroy.

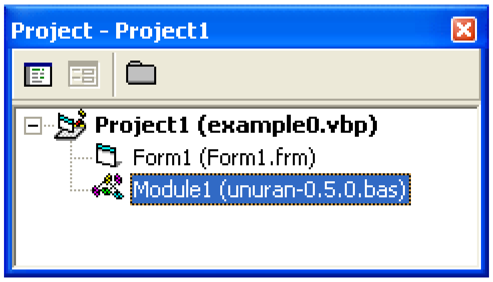
- unuran_sample (ByVal handle As Long) As Double

  This function return a random number according to the API_String in the unuran_create call.

- unuran_destroy (ByVal handle As Long)

  Memory cleanup.

These functions are declared in the provided unuran-0.5.0.bas file, which should be included in each Visual Basic project.

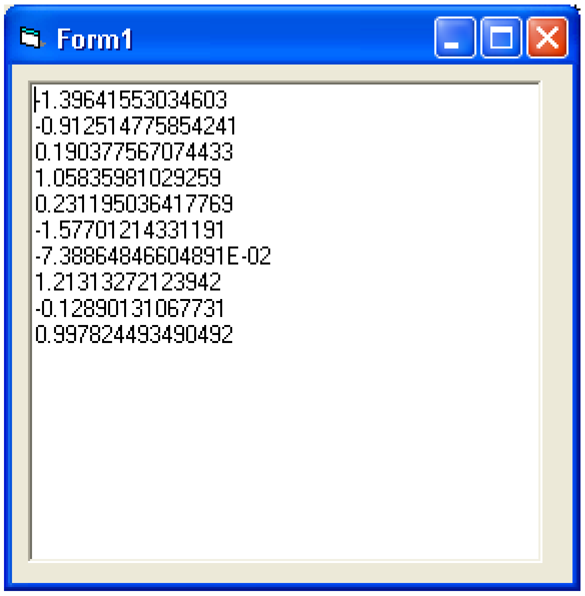


Furthermore the unuran-0.5.0.dll should be placed in the current directory of the Visual Basic project. (Alternatively, this file could be placed in the central windows\system32 directory instead).

Inspecting the content of our Visual Basic source code :

```
Private Sub Form_Load()
  Dim string_api As String
  Dim x As Double
  ' building a string using syntax from the "String-API"
  api_string = "normal()"
  ' obtain a handle to the generator object
  unuran_handle = unuran_create(api_string)

  If (unuran_handle > 0) Then
    For i = 1 To 10
      ' sample and show the generated random numbers
      x = unuran_sample(unuran_handle)
      Text1 = Text1 & x & vbCrLf
    Next i
    ' freeing the generator object
    Call unuran_destroy(unuran_handle)
  Else
    Text1 = "ERROR: random-generator could not be initialized"
  End If
End Sub
```

Upon execution we see our generated random numbers :



```
-1.39641553034603
-0.912514775854241
0.190377567074433
1.05835981029259
0.231195036417769
-1.57701214331191
-7.38864846604891E-02
1.21313272123942
-0.12890131067731
0.997824493490492
```

It is thus also possible to implement the UNU.RAN random number generators in VBA (Visual Basic for Applications) compatible programs like Excel, CorelDraw etc. by importing the declarations as found in the unuran-0.5.0.bas module and invoking calls to the unuran_create, unuran_sample and unuran_destroy functions.