



Étapes du Projet « First PlugIn » avec Unreal Engine 4.12.5

Éric JACOPIN

`eric.jacopin@st-cyr.terre-net.defense.gouv.fr`

[Plugins Programming Guide \(Epic Games\)](#)
[Custom Blueprint Node Creation \(Epic Games\)](#)

Objectif : Maîtriser le processus de création d'un Plugin pour l'Unreal Engine 4

Application : Une librairie de nœuds Blueprint pour générer aléatoirement des nombres entiers

Table des matières :

[Partie 1 : Générer les fichiers d'un Plugin](#)

Partie 1.1 : Créer un projet C++ basique

Partie 1.2 : Créer un nouveau Plugin

Partie 1.3 : Tester le Plugin

[Partie 2 : Modifier le Plugin](#)

Partie 2.1 : Changer la valeur retournée

Partie 2.2 : Utiliser le paramètre

Partie 2.3 : Ajouter une opération

[Partie 3 : Un Plugin pour la génération de nombres aléatoires](#)

Partie 3.1 : Préparation du code

Partie 3.2 : Ajouter plusieurs opérations pour générer aléatoirement des nombres entiers

Exercices : Valeurs initiales et combinaison de deux LCGs

[Partie 4 : Distribuer un Plugin](#)

Partie 4.1 : Choisir des méta-données appropriées

Partie 4.2 : Mettre le Plugin à disposition d'un autre projet UE4

Partie 4.3 : Mettre le Plugin à disposition de l'Unreal Engine 4

Partie 4.4 : Finalisation de la distribution du Plugin

[Annexe 1 : Fichiers .h et .cpp de la Partie 1 tels que générés par UE4](#)

Annexe 1.1 : Fichier « BP_Lib_V3BPLibrary.h »

Annexe 1.2 : Fichier « BP_Lib_V3BPLibrary.cpp »

[Annexe 2 : Fichiers .h et .cpp de la Partie 1](#)

Annexe 2.1 : Fichier « BP_Lib_V3BPLibrary.h »

Annexe 2.2 : Fichier « BP_Lib_V3BPLibrary.cpp »

[Annexe 3 : Fichiers .h et .cpp de la Partie 3](#)

Annexe 3.1 : Fichier « BP_Lib_V3BPLibrary.h »

Annexe 3.2 : Fichier « BP_Lib_V3BPLibrary.cpp »

[Annexe 4 : Propositions de corrections](#)

Annexe 4.1 : Entêtes des opérations du générateur combiné

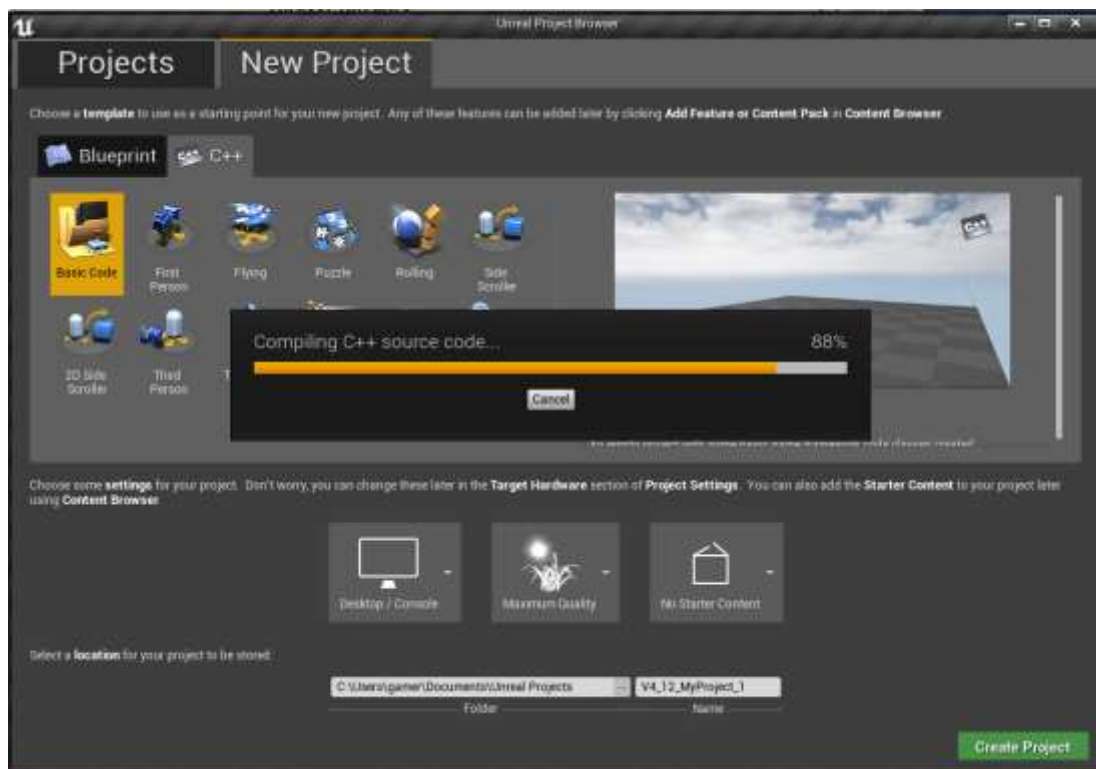
Annexe 4.2 : Implémentations des opérations du générateur combiné

Partie 1 : Générer les fichiers d'un Plugin

- ➔ Le code des fichiers originaux générés par UE4 se trouve en [Annexe 1](#)
- ➔ Le Plugin est créé avec les outils fournis par UE4, à l'intérieur d'un projet UE4 préalablement créé ; ainsi, les répertoires et fichiers créés par UE4 pour ce Plugin se trouvent dans le répertoire « Plugins » du projet UE4

Partie 1.1 : Créer un projet C++ basique

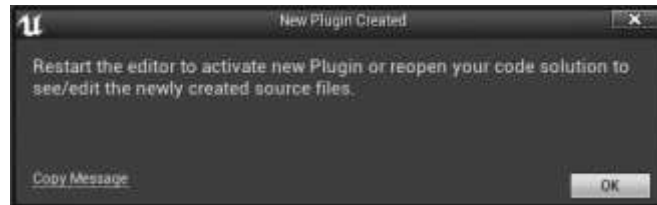
1. S'assurer que la version d'Unreal Engine 4.12.5 soit installée avec Visual Studio 2015 Update 3 (14.0.25425.01) et Microsoft .NET (4.6.01055)
2. S'assurer que les numéros de lignes soient affichés dans l'éditeur de code de Visual Studio 2015 en cochant la case « Numéros de lignes » dans l'onglet : Outils > Options... > Éditeur de texte > C/C++ > Général
3. Lancer UE 4.12.5 > New Project > C++ > Basic Code
4. Choisir un nom pour ce projet, par exemple « V4_12_MyProject_1 », puis cliquer sur « Create Project » et attendre que la fenêtre de l'éditeur soit ouverte (UE4 génère ainsi des fichiers C++ puis utilise Visual Studio pour compiler ce projet C++ basique) :



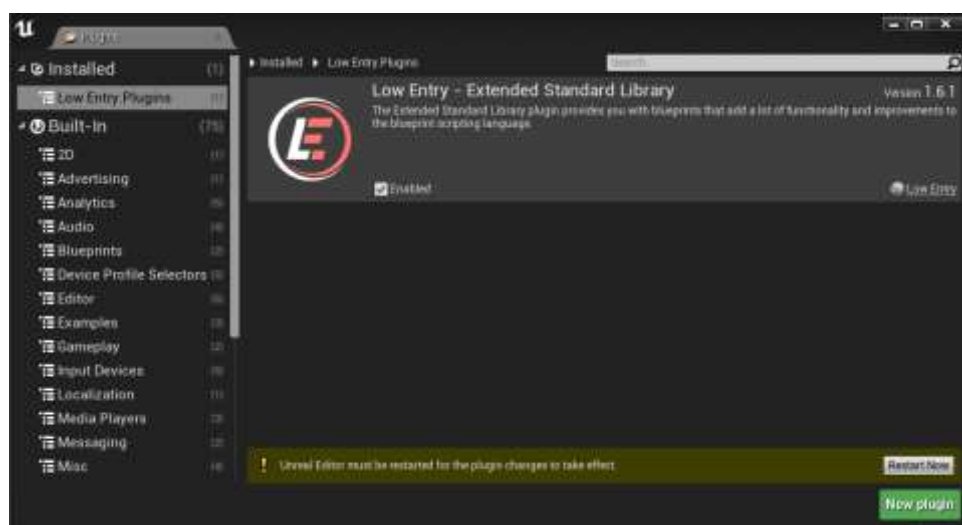
Partie 1.2 : Créer un nouveau Plugin

1. Dans l'éditeur, Settings > Plugins puis cliquer sur « New Plugin » et faire défiler l'ascenseur si nécessaire pour sélectionner « Blueprint Library »

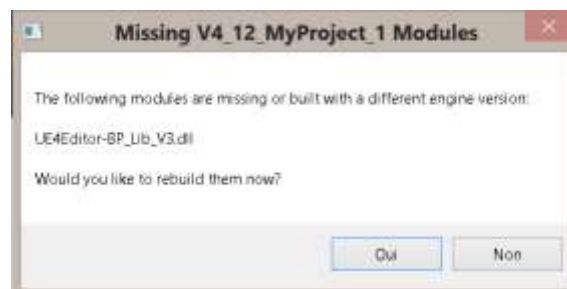
- Choisir un nom, par exemple « BP_Lib_V3 » et cliquer sur « Create Plugin » ; les répertoires et les fichiers du Plugin sont générés (le projet Visual Studio 2015 se trouve dans le répertoire du projet UE4 « V4_12_MyProject_1 ») puis la boîte de dialogue suivante s'affiche :



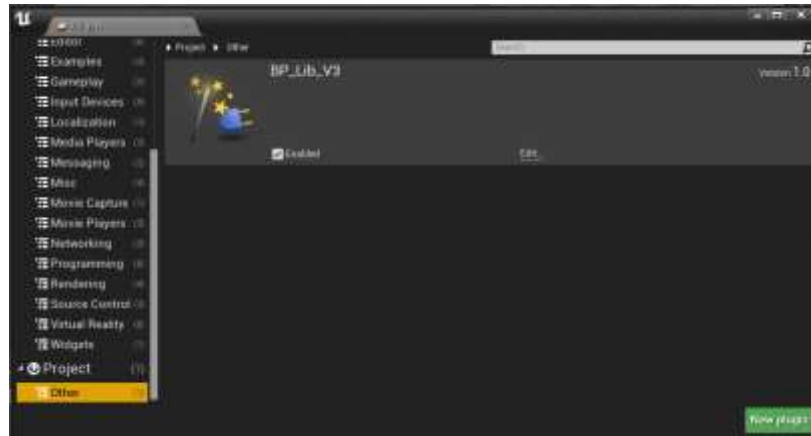
- Cliquer sur « OK » puis sur « Restart Now » dans l'onglet des Plugins :



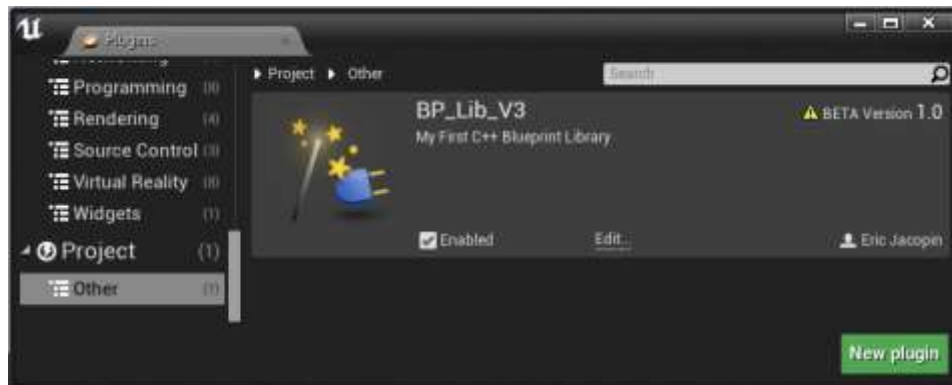
- Valider la reconstruction des modules en cliquant sur « Oui » :



- L'éditeur se ferme et se relance après une compilation C++ pour s'ouvrir sur l'onglet des Plugins ; faire défiler l'ascenseur sur la gauche de l'onglet « Plugins » pour rendre visible le Plugin « BP_Lib_V3 » en cliquant sur « Other » :



6. Cliquer sur « Edit... » pour ajouter et modifier les champs à votre convenance ; l'auteur, le numéro et le type de version seront visibles à partir de l'onglet des Plugins :

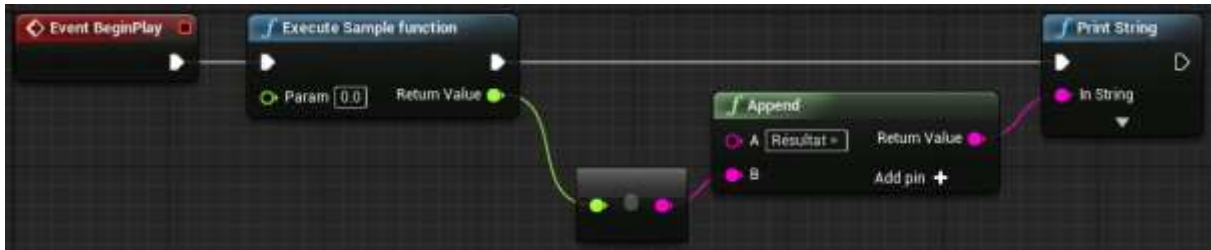


7. Fermer l'onglet des Plugins puis, dans la barre des tâches de Windows, cliquer sur Visual Studio 2015 ; puis cliquer sur « Abandonner » pour charger les mises à jours à partir du disque :



Partie 1.3 : Tester le Plugin

1. Dans l'éditeur d'Unreal > Blueprints > Open Level Blueprint
2. Dans le Level Blueprint > Right-click pour faire apparaître un menu contextuel permettant d'ajouter des nœuds et dérouler « BP_Lib_V3_Testing » pour ajouter un « Execute Sample function » et imprimer la valeur qu'elle retourne à partir de « Event BeginPlay » :



3. Compiler, sauvegarder et tester (left-click sur « Play ») ; « Résultat = -1.0 » doit s'afficher en haut et à gauche de la fenêtre de jeu, puis disparaître...
4. Dans l'explorateur de solutions de Visual Studio 2015, dérouler Plugins > BP_Lib_V3 > Source > BP_Lib_V3 > Public et ouvrir le fichier « BP_Lib_V3BPLibrary.h » ; l'entête de la fonction doit se trouver en ligne 30 et 31
5. Dans Visual Studio 2015, ouvrir le fichier « BP_Lib_V3BPLibrary.cpp » (left-click sur le fichier dans l'explorateur de solutions) pour vérifier, en ligne 14, que la fonction n'utilise pas son paramètre « Param » de type « float » et ne fait que retourner « -1 »

Partie 2 : Modifier le Plugin

- ➔ **Rappel 1 :** Le projet Visual Studio est directement accessible à partir du répertoire du projet UE4 ; dans une fenêtre Windows, un double-click-gauche sur le (fichier du) projet ouvrira le projet dans Visual Studio 2015.
- ➔ **Rappel 2 :** Une opération d'une classe C++ à destination d'un Blueprint qui est précédée du mot-clef « `static` » n'a pas de case « self » dans le nœud Blueprint associé et peut donc être utilisé dans n'importe quel contexte.
- ➔ **Rappel 3 :** Les types entiers non signés, c'est-à-dire `uint8`, `uint16`, `uint32` et `uint64`, et le type entier signé `int64` ne peuvent pas être utilisés dans l'entête d'une opération implémentant un nœud Blueprint.
- ➔ Le code final de cette partie se trouve en [Annexe 2](#)

Partie 2.1 : Changer la valeur retournée

1. Dans l'éditeur d'UE4, left-click sur le bouton « Save » pour créer « NewMap »
2. Settings > Project Settings... > Maps & Modes > ; donnez la valeur « NewMap » à « Editor Startup Map » et « Game Default Map » puis, dans l'éditeur UE4 : File > Exit pour fermer le projet
3. Dans Visual Studio 2015, modifier la valeur « -1 » en « -2 » à la ligne 14 du fichier « BP_Lib_V3BPLibrary.cpp » puis click-droit sur « V4_12_MyProject_1 » pour régénérer le projet
4. Lorsque la régénération C++ est terminée, relancer « V4_12_MyProject_1 » qui doit maintenant s'ouvrir sur « NewMap » ; click sur « Play » et vérifier que « Résultat = -2.0 » s'affiche en haut et à gauche de la fenêtre de jeu puis disparaît après quelques secondes...

Partie 2.2 : Utiliser le paramètre

1. Dans l'éditeur UE4 : File > Exit pour fermer le projet ; dans Visual Studio 2015, modifier la valeur « -2 » en l'expression « -2 * Param » à la ligne 14 du fichier « BP_Lib_V3BPLibrary.cpp » puis click-droit sur « V4_12_MyProject_1 » pour régénérer le projet
2. Lorsque la régénération C++ est terminée, relancer « V4_12_MyProject_1 » et ouvrir le Level Blueprint pour modifier la valeur de « Param » en 20, par exemple ; click sur « Play » et vérifier que « Résultat = -20.0 » s'affiche en haut et à gauche de la fenêtre de jeu

Partie 2.3 : Ajouter une opération

1. Dans l'éditeur UE4 : File > Exit pour fermer le projet ; dans Visual Studio 2015, ajouter, juste après « `GENERATED_UCLASS_BODY()` » (qui doit se trouver en ligne 28) le code suivant dans le fichier « BP_Lib_V3BPLibrary.h » :

29.

```

30. private:
31.     static unsigned long next;
32.
33. public:
34.     UFUNCTION(BlueprintCallable, meta = (DisplayName = "Execute c88rand",
        Keywords = "BP_Lib_V3 sample test testing"), Category = "BP_Lib_V3Testing")
35.     static int32 BP_Lib_V3c88rand();    /* RAND_MAX assumed to be 32767 */
36.

```

2. Dans Visual Studio 2015, ajouter, le code suivant en ligne 6 du fichier « BP_Lib_V3BPLibrary.cpp » :

```

6. unsigned long UBP_Lib_V3BPLibrary::next = 0;

```

3. Dans Visual Studio 2015, ajouter, le code suivant à partir de la ligne 14 du fichier « BP_Lib_V3BPLibrary.cpp » :

```

14. int32 UBP_Lib_V3BPLibrary::BP_Lib_V3c88rand()
15. {
16.     UBP_Lib_V3BPLibrary::next = UBP_Lib_V3BPLibrary::next * 1103515245 + 12345;
17.     return (int32)((UBP_Lib_V3BPLibrary::next / 65536) % 32768);
18. }
19.

```

4. Dans Visual Studio 2015, régénérer le projet « V4_12_MyProject_1 » (click-droit sur le nom du projet pour faire apparaître un menu contextuel qui donne accès la régénération)
5. Lorsque la régénération C++ est terminée, relancer « V4_12_MyProject_1 » et cliquer sur « Play » pour voir s’afficher « 0 » en haut et à gauche ; appuyer sur la touche « Esc » pour arrêter le jeu puis à nouveau sur « Play » pour voir s’afficher la valeur « 21468 » et ainsi de suite ; les 11 premiers nombres qui apparaîtront sont les suivants : 0, 21468, 9988, 22117, 3498, 16927, 16045, 19741, 12122, 8410, 12261, ...
6. Dans le fichier « BP_Lib_V3BPLibrary.h », modifier l’initialisation de `UBP_Lib_V3BPLibrary::next` (ligne 6 avant la modification, cf. le point 2. ci-dessus) de façon à ce que la précédente suite des nombres diffère entre deux appels successifs :

```

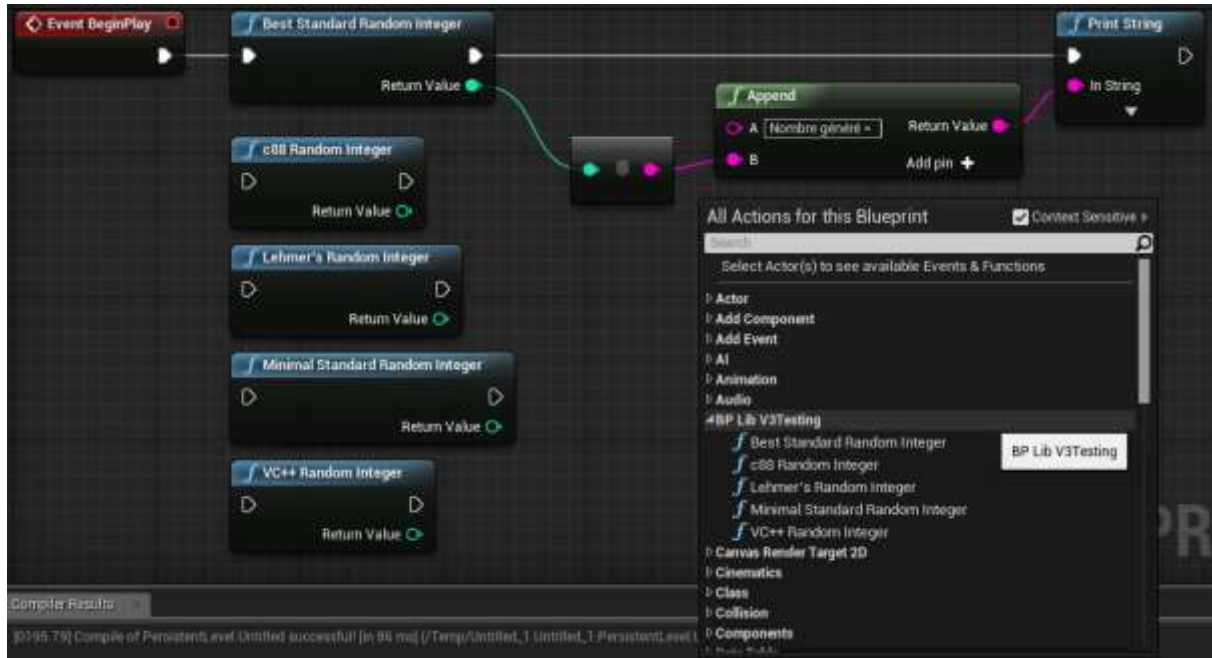
6. #include <time.h>          // Visibility for time/1
7.
8. unsigned long UBP_Lib_V3BPLibrary::next = (unsigned long) time(NULL);

```

7. Dans Visual Studio 2015, régénérer le projet « V4_12_MyProject_1 » puis relancer le projet UE4 « V4_12_MyProject_1 » pour vérifier l’effet de cette modification : [en quittant le projet UE4 (Dans l’éditeur > File > Exit) et en relançant le projet pour cliquer sur « Play »] plusieurs fois, on peut constater que le premier nombre affiché en haut et à gauche de la fenêtre de jeu n’est pas le même

Partie 3 : Un Plugin pour la génération de nombres aléatoires

- ➔ Le code final de cette partie se trouve en [Annexe 3](#)
- ➔ Dans UE4 le plugin propose 5 nœuds Blueprint à partir de « BP Lib V3 Testing » :



Partie 3.1 : Préparation du code actuel

1. Supprimer le code devenu maintenant superflu de l'opération
BP_Lib_V3SampleFunction incluse lors de la génération des fichiers par Unreal et renommer next en next_c88
2. Dans les fichiers « BP_Lib_V3BPLibrary.h » et « BP_Lib_V3BPLibrary.cpp », conserver le commentaire de la première ligne mais supprimer tous les autres commentaires et placer le commentaire suivant sur la ligne qui précède l'entête de la fonction BP_Lib_V3c88rand dans le fichier « BP_Lib_V3BPLibrary.h » :

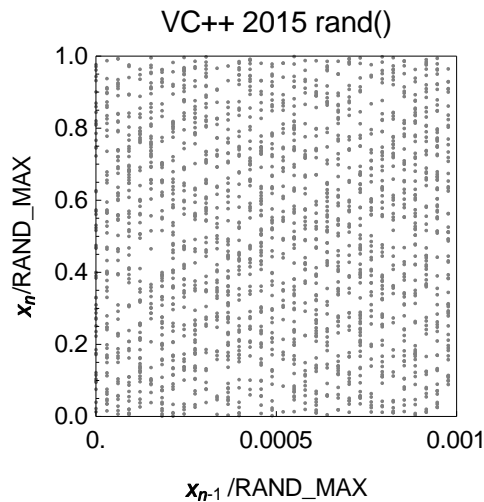

```
/* Returns a random integer between 0 and 32767 -- Linear Congruential RNG from
ANSI C88 with a=1103515245, b=12345 and m=32768 */
```
3. Ce commentaire sera visible dans l'éditeur de Blueprint lorsque la souris passera sur le nœud de l'opération

Partie 3.2 : Ajouter plusieurs opérations pour générer aléatoirement des nombres entiers

1. Le code de l'[Annexe 3](#) propose plusieurs générateurs de nombres aléatoires, tous basés sur méthode Linéaire ConGruentielle (LCG) où le prochain nombre aléatoire x_{i+1} est construit de façon linéaire à partir d'un nombre entier : $x_n = (a x_{n-1} + b) \text{ modulo } m$; lorsque b est égal à 0 on dit que le générateur est Multiplicatif (MLCG)
2. Pour un MLCG $x_i = 0$ entraîne que $\forall n > i x_n = 0$: il faut donc faire attention à ne pas utiliser la valeur 0 comme valeur initiale ; les MLCGs sont conçus de façon à ce que la valeur

0 ne puisse pas être généré : on choisit a et m de façon à ce que seul $x_i = m$ donne $x_{i+1} = 0$, ce qui n'arrivera jamais car, en raison de l'opération modulo, $\forall i, x_i \leq m - 1$

- Les LCGs/MLCGs sont cycliques : ils finissent par générer les nombres qu'ils ont déjà générés, et ce quel que soit le nombre initial x_0 qu'on leur injecte ; le graphe (x_{n-1}, x_n) donne une visualisation de cette caractéristique, par exemple ici pour la fonction rand() de Visual Studio 2015 (RAND_MAX = 32767) :



- Le code de l'opération BP_Lib_V3vcpprand() implémente exactement la fonction rand() qui se trouve dans Visual C/C++ 2015
- Le code de l'opération BP_Lib_V3lehmer() est donné à titre historique, car il est l'un des premiers LCG à avoir été publié, sinon le premier, et ce en 1949 :

Lehmer, D.H. 1949. Mathematical Methods in Large Scale Computing Units. In Proceedings of the 2nd Symposium on Large Scale Digital Calculating Machinery, 141–146. Harvard University Press.

- Le code de l'opération BP_Lib_V3mslclg() implémente un MLCG utilisé par la NASA au début des années 1980s (<http://www.sti.nasa.gov/>) :

[Generation of pseudo-random numbers](#)

Author: Howell, L. W.; Rheinfurth, M. H.

Abstract: Practical methods for generating acceptable random numbers from a variety of probability distributions which are frequently encountered in engineering applications are described. The speed, accuracy, and guarantee of statistical randomness of the various methods are discussed

Publication Year: 1982

Document Type: Technical Report

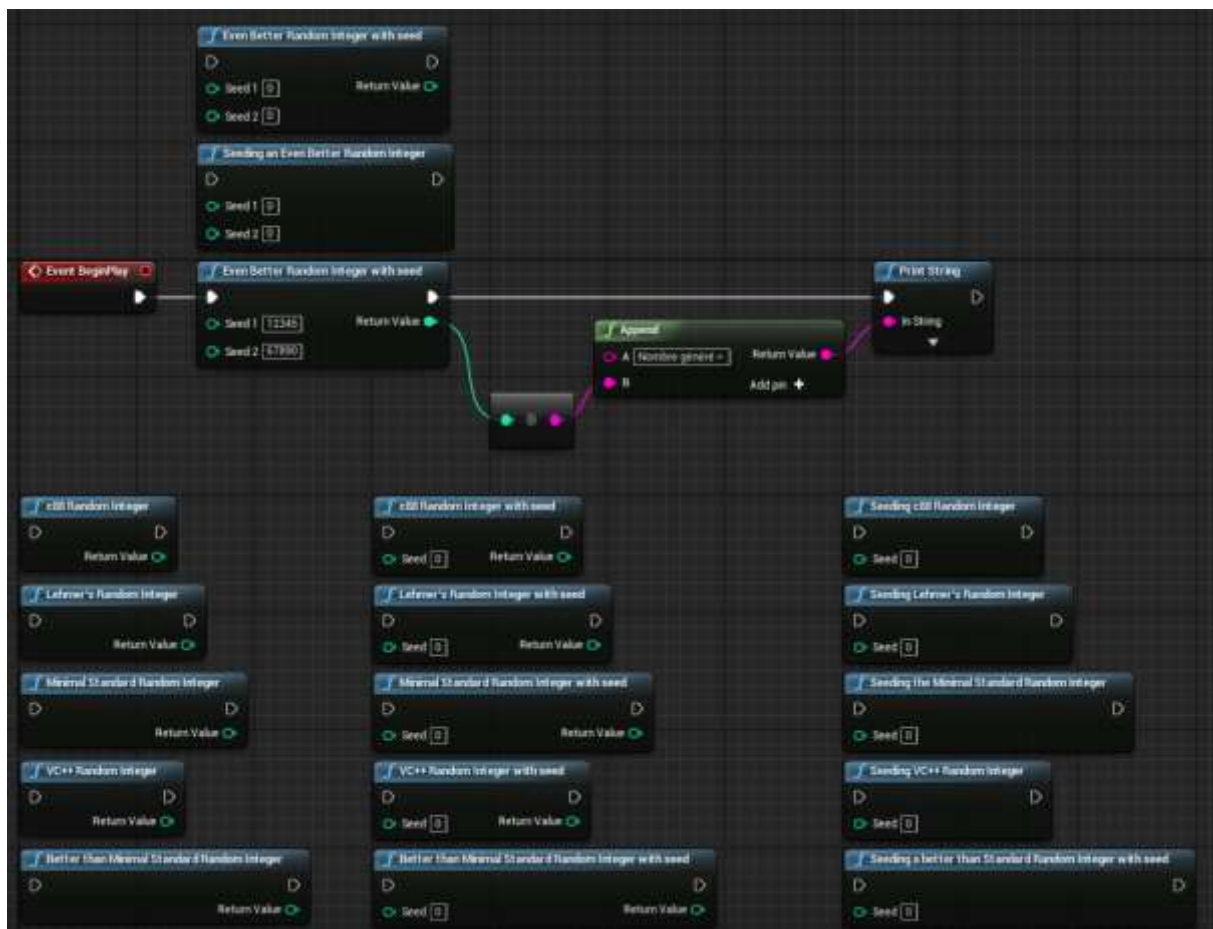
Report/Patent Number: NASA-TP-2105, NAS 1.60:2105

- Le code de l'opération BP_Lib_V3b1lclg() implémente un MLCG actuellement connu comme l'un des meilleurs (pour un générateur linéaire congruentiel) en termes de qualité (par exemple, son cycle est très grand et la distribution des nombres générés est meilleure) :

L'Écuyer, P. 1988. Efficient and Portable Combined Random Number Generator. *Communications of the ACM* 31(6):742–749, 774.

Exercices : Valeurs initiales et combinaison de deux LCGs

- ➔ Des propositions de corrections pour les questions 1. et 2. se trouvent dans l'Annexe 4
 - ➔ Le code donné dans les annexes se veut lisible et pédagogique plutôt qu'optimisé
1. Comment choisir a et m de façon à ce que seul $x_i = m$ donne $x_{i+1} = 0$?
 2. Les valeurs initiales sont générées par un appel à `time/1` ; pour chaque nœud Blueprint, ajoutez un second nœud avec en entrée une valeur qui servira de valeur initiale ; ce paramètre sera nommé « Seed » et sera de type « `int32` » ; la valeur absolue de « Seed » sera utilisée dans l'implémentation pour éviter les effets d'une valeur négative.
 3. Ajoutez le « Combined LCG » qui est décrit dans l'article de Pierre L'Écuyer (référence dans le point 7. ci-dessus) ainsi qu'un second nœud permettant d'initialiser les deux premières valeurs ; notez que si $x < 0$ alors $(x \text{ modulo } m) = m - ((-x) \text{ modulo } m)$.



Partie 4 : Distribuer un Plugin

- ➔ Les fonctionnalités qui ont été développées sont accessibles dans le projet « V4_12_MyProject_1 » ; que faut-il faire pour que ces fonctionnalités soient accessibles dans un autre projet ou dans n'importe quel autre projet ?
- ➔ Le répertoire du Plugin « BP_Lib_V3 » se trouve dans le répertoire « Plugins » du répertoire du projet « V4_12_MyProject_1 »

Partie 4.1 : Choisir des méta-données appropriées

1. Dans le fichier « BP_Lib_V3BPLibrary.h » s'assurer que les méta-données conviennent et sinon, c'est le moment de choisir un contenu approprié pour les mots-clefs **DisplayName**, **Keywords** et **Category** ; par exemple pour ce dernier : `Category = "Math|Random|Vintage RNGs"` ; terminer par régénérer le projet dans Visual Studio 2015 lorsque les méta-données conviennent

Partie 4.2 : Mettre le Plugin à disposition d'un autre projet UE4

2. Copier le répertoire du Plugin « BP_Lib_V3 » dans le dossier Plugins d'un projet UE4, par exemple « C:\Users\gamer\Documents\Unreal Projects\V4_12_MyProject_2\Plugins\ », pour rendre immédiatement disponibles les fonctionnalités du Plugin « BP_Lib_V3 » pour le projet « V4_12_MyProject_2 »

Partie 4.3 : Mettre le Plugin à disposition de l'Unreal Engine 4

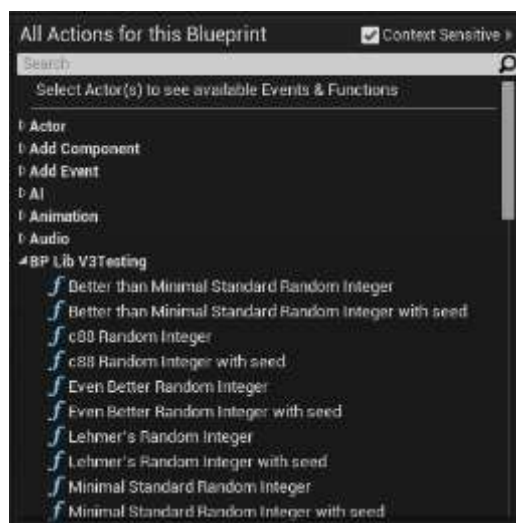
1. Copier le répertoire du Plugin « BP_Lib_V3 » dans le dossier Plugins d'UE4, par exemple « C:\Program Files\Epic Games\4.12\Engine\Plugins » pour rendre disponible les fonctionnalités du Plugin pour n'importe quel projet fabriqué avec la version 4.12 d'UE4
2. Settings > Plugins > Others pour vérifier qu'UE4 a bien détecté le Plugin « BP_Lib_V3 » :



3. Cocher la case « Enabled » puis valider le redémarrage d'UE4 en cliquant sur « Restart Now » :



4. Lorsqu'UE4 est redémarré, Blueprints > Open Level Blueprint puis click-droit dans la fenêtre de l'éditeur de Blueprints pour vérifier que les fonctionnalités de BP_Lib_V3 sont accessibles :



Partie 4.4 : Finalisation de la distribution du Plugin

1. Copier le Plugin dans un répertoire de travail
2. Concevoir un icône de 128 x 128 pixels au format PNG et remplacer, sans changer le nom, le fichier « Icon128.png » qui se trouve dans le répertoire « Ressources » du Plugin
3. Selon les objectifs, supprimer le répertoire « Source » et le répertoire « Intermediate » qui se trouvent dans le répertoire de travail du Plugin ; le fichier « Binaries\Win64\ UE4Editor-BP_Lib_V3.pdb » pourra aussi être supprimé
4. Voilà... C'est fini, le Plugin est prêt à être distribué sur la Market Place d'UE4 ! 😊



Annexe 1 : Fichiers .h et .cpp de la Partie 1 tels que générés par UE4

Annexe 1.1 : Fichier « BP_Lib_V3BPLibrary.h »

```
// Copyright 1998-2016 Epic Games, Inc. All Rights Reserved.
```

```
#pragma once
```

```
#include "Engine.h"
```

```
#include "BP_Lib_V3BPLibrary.generated.h"
```

```
/*  
 *      Function library class.  
 *      Each function in it is expected to be static and represents blueprint node that can be called in any blueprint.  
 *  
 *      When declaring function you can define metadata for the node. Key function specifiers will be BlueprintPure and BlueprintCallable.  
 *      BlueprintPure - means the function does not affect the owning object in any way and thus creates a node without Exec pins.  
 *      BlueprintCallable - makes a function which can be executed in Blueprints - Thus it has Exec pins.  
 *      DisplayName - full name of the node, shown when you mouse over the node and in the blueprint drop down menu.  
 *                      Its lets you name the node using characters not allowed in C++ function names.  
 *      CompactNodeTitle - the word(s) that appear on the node.  
 *      Keywords -      the list of keywords that helps you to find node when you search for it using Blueprint drop-down menu.  
 *                      Good example is "Print String" node which you can find also by using keyword "log".  
 *      Category -      the category your node will be under in the Blueprint drop-down menu.  
 *  
 *      For more info on custom blueprint nodes visit documentation:  
 *      https://wiki.unrealengine.com/Custom\_Blueprint\_Node\_Creation  
 */
```

```
UCLASS()  
class UBP_Lib_V3BPLibrary : public UBlueprintFunctionLibrary  
{  
    GENERATED_UCLASS_BODY()  
  
    UFUNCTION(BlueprintCallable, meta = (DisplayName = "Execute Sample function", Keywords = "BP_Lib_V3 sample test testing"), Category = "BP_Lib_V3Testing")  
    static float BP_Lib_V3SampleFunction(float Param);  
};
```



Création le 16/08/2016 – Dernière révision le 02/09/2016

Annexe 1.1 : Fichier « BP_Lib_V3BPLibrary.cpp »

```
// Copyright 1998-2016 Epic Games, Inc. All Rights Reserved.

#include "BP_Lib_V3PrivatePCH.h"
#include "BP_Lib_V3BPLibrary.h"

UBP_Lib_V3BPLibrary::UBP_Lib_V3BPLibrary(const FObjectInitializer& ObjectInitializer)
: Super(ObjectInitializer)
{

}

float UBP_Lib_V3BPLibrary::BP_Lib_V3SampleFunction(float Param)
{
    return -1;
}
```



Annexe 2 : Fichiers .h et .cpp de la Partie 2

Annexe 2.1 : Fichier « BP_Lib_V3BPLibrary.h »

```
// Copyright 1998-2016 Epic Games, Inc. All Rights Reserved.
```

```
#pragma once
```

```
#include "Engine.h"
```

```
#include "BP_Lib_V3BPLibrary.generated.h"
```

```
/*  
 *      Function library class.  
 *      Each function in it is expected to be static and represents blueprint node that can be called in any blueprint.  
 *  
 *      When declaring function you can define metadata for the node. Key function specifiers will be BlueprintPure and BlueprintCallable.  
 *      BlueprintPure - means the function does not affect the owning object in any way and thus creates a node without Exec pins.  
 *      BlueprintCallable - makes a function which can be executed in Blueprints - Thus it has Exec pins.  
 *      DisplayName - full name of the node, shown when you mouse over the node and in the blueprint drop down menu.  
 *                     Its lets you name the node using characters not allowed in C++ function names.  
 *      CompactNodeTitle - the word(s) that appear on the node.  
 *      Keywords -      the list of keywords that helps you to find node when you search for it using Blueprint drop-down menu.  
 *                     Good example is "Print String" node which you can find also by using keyword "log".  
 *      Category -      the category your node will be under in the Blueprint drop-down menu.  
 *  
 *      For more info on custom blueprint nodes visit documentation:  
 *      https://wiki.unrealengine.com/Custom\_Blueprint\_Node\_Creation  
 */
```

```
UCLASS()
```

```
class UBP_Lib_V3BPLibrary : public UBlueprintFunctionLibrary  
{
```

```
    GENERATED_UCLASS_BODY()
```

```
private:
```

```
    static unsigned long next;
```

```
public:
```

```
    UFUNCTION(BlueprintCallable, meta = (DisplayName = "Execute c88rand", Keywords = "BP_Lib_V3 sample test testing"), Category = "BP_Lib_V3Testing")  
    static int32 BP_Lib_V3c88rand(); /* RAND_MAX assumed to be 32767 */
```

```
    UFUNCTION(BlueprintCallable, meta = (DisplayName = "Execute Sample function", Keywords = "BP_Lib_V3 sample test testing"), Category = "BP_Lib_V3Testing")  
    static float BP_Lib_V3SampleFunction(float Param);
```

```
};
```



Annexe 2.2 : Fichier « BP_Lib_V3BPLibrary.cpp »

```
// Copyright 1998-2016 Epic Games, Inc. All Rights Reserved.

#include "BP_Lib_V3PrivatePCH.h"
#include "BP_Lib_V3BPLibrary.h"

#include <time.h>          // Visibility for time/1

unsigned long UBP_Lib_V3BPLibrary::next = (unsigned long) time(NULL);

UBP_Lib_V3BPLibrary::UBP_Lib_V3BPLibrary(const FObjectInitializer& ObjectInitializer)
: Super(ObjectInitializer)
{
}

int32 UBP_Lib_V3BPLibrary::BP_Lib_V3c88rand()  /* RAND_MAX assumed to be 32767 */
{
    UBP_Lib_V3BPLibrary::next = UBP_Lib_V3BPLibrary::next * 1103515245 + 12345;
    return (int32)((UBP_Lib_V3BPLibrary::next / 65536) % 32768);
}

float UBP_Lib_V3BPLibrary::BP_Lib_V3SampleFunction(float Param)
{
    return (-2 * Param);
}
```




Annexe 3 : Fichiers .h et .cpp de la Partie 3

Annexe 3.1 : Fichier « BP_Lib_V3BPLibrary.h »

// Copyright 1998-2016 Epic Games, Inc. All Rights Reserved.

#pragma once

#include "Engine.h"

#include "BP_Lib_V3BPLibrary.generated.h"

UCLASS()

class UBP_Lib_V3BPLibrary : public UBlueprintFunctionLibrary

{

GENERATED_UCLASS_BODY()

private:

static uint32 next_lehmer;

static uint32 next_c88;

static uint32 next_vcpp;

static uint64 next_mslcg;

static uint64 next_bllcg;

public:

/* Returns a random integer between 1 and 1000000000 -- Multiplicative Linear Congruential RNG from Lehmer (1949) with a=23 and m=1000000001 */

UFUNCTION(BlueprintCallable, meta = (DisplayName = "Lehmer's Random Integer", Keywords = "BP_Lib_V3 sample test testing"), Category = "BP_Lib_V3Testing")

static int32 BP_Lib_V3lehmerand();

/* Returns a random integer between 0 and 32767 -- Linear Congruential RNG from ANSI C88 with a=1103515245, b=12345 and m=2^15=32768 */

UFUNCTION(BlueprintCallable, meta = (DisplayName = "c88 Random Integer", Keywords = "BP_Lib_V3 sample test testing"), Category = "BP_Lib_V3Testing")

static int32 BP_Lib_V3c88rand();

/* Returns a random integer between 0 and 32767 -- Linear Congruential RNG from Visual C/C++ with a=214013, b=2531011 and m=2^31=2147483648 */

UFUNCTION(BlueprintCallable, meta = (DisplayName = "VC++ Random Integer", Keywords = "BP_Lib_V3 sample test testing"), Category = "BP_Lib_V3Testing")

static int32 BP_Lib_V3vcpprand();

/* Returns a random integer between 1 and 2147483646 -- Multiplicative Linear Congruential RNG with a=16807 and m=2^31-1=2147483647 */

UFUNCTION(BlueprintCallable, meta = (DisplayName = "Minimal Standard Random Integer", Keywords = "BP_Lib_V3 sample test testing"), Category = "BP_Lib_V3Testing")

static int32 BP_Lib_V3mslcgrand();

/* Returns a random integer between 1 and 2147483398 -- Multiplicative Linear Congruential RNG with a=40692 and m=2147483399 */

UFUNCTION(BlueprintCallable, meta = (DisplayName = "Best Standard Random Integer", Keywords = "BP_Lib_V3 sample test testing"), Category = "BP_Lib_V3Testing")

static int32 BP_Lib_V3bllcgrand();

};



Annexe 3.2 : Fichier « BP_Lib_V3BPLibrary.cpp »

```
// Copyright 1998-2016 Epic Games, Inc. All Rights Reserved.
```

```
#include "BP_Lib_V3PrivatePCH.h"
```

```
#include "BP_Lib_V3BPLibrary.h"
```

```
#include <time.h>          // Visibility for time/1
```

```
uint32 UBP_Lib_V3BPLibrary::next_lehmer = (uint32)time(NULL);  
uint32 UBP_Lib_V3BPLibrary::next_c88   = (uint32)time(NULL);  
uint32 UBP_Lib_V3BPLibrary::next_vcpp  = (uint32)time(NULL);  
uint64 UBP_Lib_V3BPLibrary::next_mslcg = (uint64)time(NULL);  
uint64 UBP_Lib_V3BPLibrary::next_b1lcg = (uint64)time(NULL);
```

```
UBP_Lib_V3BPLibrary::UBP_Lib_V3BPLibrary(const FObjectInitializer& ObjectInitializer)  
: Super(ObjectInitializer)
```

```
{  
}
```

```
int32 UBP_Lib_V3BPLibrary::BP_Lib_V3lehmerand()
```

```
{  
    UBP_Lib_V3BPLibrary::next_lehmer = ((UBP_Lib_V3BPLibrary::next_lehmer * 23) % 100000001);  
    return (int32)(UBP_Lib_V3BPLibrary::next_lehmer);  
}
```

```
int32 UBP_Lib_V3BPLibrary::BP_Lib_V3c88rand()
```

```
{  
    UBP_Lib_V3BPLibrary::next_c88 = UBP_Lib_V3BPLibrary::next_c88 * 1103515245 + 12345;  
    return (int32)((UBP_Lib_V3BPLibrary::next_c88 / 65536) % 32768);  
}
```

```
int32 UBP_Lib_V3BPLibrary::BP_Lib_V3vcpprand()
```

```
{  
    UBP_Lib_V3BPLibrary::next_vcpp = (UBP_Lib_V3BPLibrary::next_vcpp * 214013 + 2531011) % 2147483648;  
    return (int32)(UBP_Lib_V3BPLibrary::next_vcpp / 65536);  
}
```

```
int32 UBP_Lib_V3BPLibrary::BP_Lib_V3mslcgrand()
```

```
{  
    UBP_Lib_V3BPLibrary::next_mslcg = ((UBP_Lib_V3BPLibrary::next_mslcg * 16807) % 2147483647);  
    return (int32)(UBP_Lib_V3BPLibrary::next_mslcg);  
}
```

```
int32 UBP_Lib_V3BPLibrary::BP_Lib_V3b1lcgrand()
```

```
{  
    UBP_Lib_V3BPLibrary::next_b1lcg = ((UBP_Lib_V3BPLibrary::next_b1lcg * 40692) % 2147483399);  
    return (int32)(UBP_Lib_V3BPLibrary::next_b1lcg);  
}
```



Annexe 4 : Propositions de corrections

Annexe 4.1 : Entêtes des opérations du générateur combiné

```
/* Returns a random integer between 1 and 2147483561 -- Combined Multiplicative Linear Congruential RNG with a1=40014, a2=40692, m1=2147483563 and m=2147483399 */
UFUNCTION(BlueprintCallable, meta = (DisplayName = "Even Better Random Integer", Keywords = "BP_Lib_V3 sample test testing"), Category = "Math|Random|Vintage RNGs")
static int32 BP_Lib_V3cmlcgrand();

/* Returns a random integer between 1 and 2147483561 from a given seed -- Combined Multiplicative Linear Congruential RNG with a1=40014, a2=40692, m1=2147483563 and m=2147483399 */
UFUNCTION(BlueprintCallable, meta = (DisplayName = "Even Better Random Integer with seed", Keywords = "BP_Lib_V3 sample test testing"), Category = "Math|Random|Vintage RNGs")
static int32 BP_Lib_V3cmlcgrandwithseed(int32 Seed1, int32 Seed2);

/* Sets the seed for a combined Multiplicative Linear Congruential RNG with a1=40014, a2=40692, m1=2147483563 and m=2147483399 */
UFUNCTION(BlueprintCallable, meta = (DisplayName = "Seeding an Even Better Random Integer", Keywords = "BP_Lib_V3 sample test testing"), Category = "Math|Random|Vintage RNGs")
static void BP_Lib_V3seedcmlcgrand(int32 Seed1, int32 Seed2);
```

Annexe 4.2 : Implémentations des opérations du générateur combiné

```
int32 UBP_Lib_V3BPLibrary::BP_Lib_V3cmlcgrand()
{
    UBP_Lib_V3BPLibrary::next_s1clcg = ((UBP_Lib_V3BPLibrary::next_s1clcg * 40014) % 2147483563);
    UBP_Lib_V3BPLibrary::next_s2clcg = ((UBP_Lib_V3BPLibrary::next_s2clcg * 40692) % 2147483399);
    int64 tmp = UBP_Lib_V3BPLibrary::next_s1clcg - UBP_Lib_V3BPLibrary::next_s2clcg;
    if (0 <= tmp)
        return (int32)(tmp % 2147483562);
    else // tmp < 0
        return (int32)(2147483562 - ((-tmp) % 2147483562));
}

int32 UBP_Lib_V3BPLibrary::BP_Lib_V3cmlcgrandwithseed(int32 Seed1, int32 Seed2)
{
    UBP_Lib_V3BPLibrary::next_s1clcg = abs(Seed1);
    UBP_Lib_V3BPLibrary::next_s2clcg = abs(Seed2);
    UBP_Lib_V3BPLibrary::next_s1clcg = ((UBP_Lib_V3BPLibrary::next_s1clcg * 40014) % 2147483563);
    UBP_Lib_V3BPLibrary::next_s2clcg = ((UBP_Lib_V3BPLibrary::next_s2clcg * 40692) % 2147483399);
    int64 tmp = UBP_Lib_V3BPLibrary::next_s1clcg - UBP_Lib_V3BPLibrary::next_s2clcg;
    if (0 <= tmp)
        return (int32)(tmp % 2147483562);
    else // tmp < 0
        return (int32)(2147483562 - ((-tmp) % 2147483562));
}

void UBP_Lib_V3BPLibrary::BP_Lib_V3seedcmlcgrand(int32 Seed1, int32 Seed2)
{
    UBP_Lib_V3BPLibrary::next_s1clcg = abs(Seed1);
    UBP_Lib_V3BPLibrary::next_s2clcg = abs(Seed2);
}
```