Procedural World Generation

Design Requirement Specification Document

DIBRIS – Università di Genova. Scuola Politecnica, Corso di Ingegneria del Software 80154

**DATA – 07/07/2018**

**VERSION: 3.0**

**Autori**

Stefano Sbarbaro

**REVISION HISTORY**

|  |  |  |  |
| --- | --- | --- | --- |
| Versione | Data | Autori | Note |
| 1.0 | 15/04/2017 | SS | Inizio del documento |
| 2.0 | 27/06/2018 | SS | Fine del documento |
| 3.0 | 07/07/2018 | SS | Fine documento  aggiornato |
|  |  |  |  |

Indice dei Contenuti

1[1Introduzione 3](#__RefHeading___Toc106_1667604106)

[1.1Scopo del Documento 3](#__RefHeading___Toc108_1667604106)

[1.2Descrizione Generale del Sistema 3](#__RefHeading___Toc110_1667604106)

[1.3Riferimenti Bibliografici 3](#__RefHeading___Toc1851_1623032702)

[1.4Definizioni e Acronimi 3](#__RefHeading___Toc112_1667604106)

[1.5Overview del documento 4](#__RefHeading___Toc1853_1623032702)

2[2Descrizione del Progetto 4](#__RefHeading___Toc1855_1623032702)

[2.1Introduzione 4](#__RefHeading___Toc1857_1623032702)

[2.2Architettura del Sistema 4](#__RefHeading___Toc1859_1623032702)

3[3Interfacce 5](#__RefHeading___Toc1863_1623032702)

[3.1Blueprint Interface 5](#__RefHeading___Toc226_1667604106)

[3.2Class Defaults Interfaces 6](#__RefHeading___Toc1010_1110369364)

3.3 Blueprint Assign Mesh 6

5[4Use Cases 7](#__RefHeading___Toc1014_1110369364)

6[5Structural Design 7](#__RefHeading___Toc1867_1623032702)

[5.1Class Diagram 7](#__RefHeading___Toc1869_1623032702)

[5.1.1RoadGeneration](#__RefHeading___Toc1871_1623032702) 8

[5.1.1.1Attributi](#__RefHeading___Toc1016_1110369364) 8

[5.1.1.2 Metodi](#__RefHeading___Toc1018_1110369364) 8

[5.1.2Noise](#__RefHeading___Toc1020_1110369364) 9

[5.1.2.1Attributi](#__RefHeading___Toc1022_1110369364) 9

[5.1.2.2 Metodi](#__RefHeading___Toc1024_1110369364) 9

[5.1.3AssignMesh](#__RefHeading___Toc1020_1110369364) 9

[5.1.3.1Attributi](#__RefHeading___Toc1022_1110369364) 10

[5.1.3.2 Metodi](#__RefHeading___Toc1024_1110369364) 10

7[6Dynamic Model](#__RefHeading___Toc1875_1623032702) 10

[6.1 Sequence diagram ... 10](#__RefHeading___Toc1032_1110369364)

# Introduzione

## Scopo del Documento

Questo documento specifica l'architettura del sistema software chiamato Procedural World Generation ed è stato redatto per gli sviluppatori, i progettisti e i tester del sistema.

## Descrizione Generale del Sistema

Procedural World Generation è un tool dedicato allo sviluppo e alla generazione in modo procedurale della piantina di una mappa, in particolare di una città.

Il software si propone di essere in grado di creare contenuti in modo indipendente o più precisamente con un limitato supporto dell’intervento dell’umano; tutte le mappe prodotte dal tool dovranno essere sempre diverse tra loro se sarà differente il seed che le contraddistingue.

Partendo da questo prototipo solo in futuro verrà implementata la terza dimensione sfruttando proprio la piantina della mappa come la base dalla quale potranno essere modellati gli edifici e/o altre diverse strutture.

Il software dovrà integrarsi con l’engine grafico Unreal Engine 4 e rispettare alcuni parametri di gameplay, generandosi di conseguenza.

## Riferimenti Bibliografici

User requirements.

“A survey of Procedural Techniques for City Generation”, articolo scritto da George Kelly & Hugh McCabe.

“Procedural Modeling of Cities”, articolo scritto da Yoav I.H. Parish & Pascal Müller.

“Procedural City Generator”, MSc Master’s Project di Praveen Kumar Ilangovan.

## Definizioni e Acronimi

|  |  |
| --- | --- |
| **Acronimo-Nome** | **Definizione** |
| PWG | Procedural World Generation |
| UE4 | Unreal Engine 4 |

## Overview del documento

Nella sezione due viene presentata una descrizione ad alto livello dell'Applicazione. La sezione cinque descrive la struttura del sistema, mentre nella sezione sei vengono descritti in dettaglio come i requisiti sono implementati.

# Descrizione del Progetto

## Introduzione

In questo progetto verrà adottato un processo object oriented con un architettura desktop. Per disegnare i diagrammi UML verrà utilizzato il tool Dia.

## Architettura del Sistema

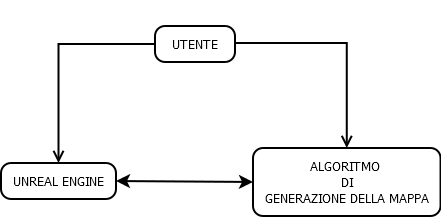


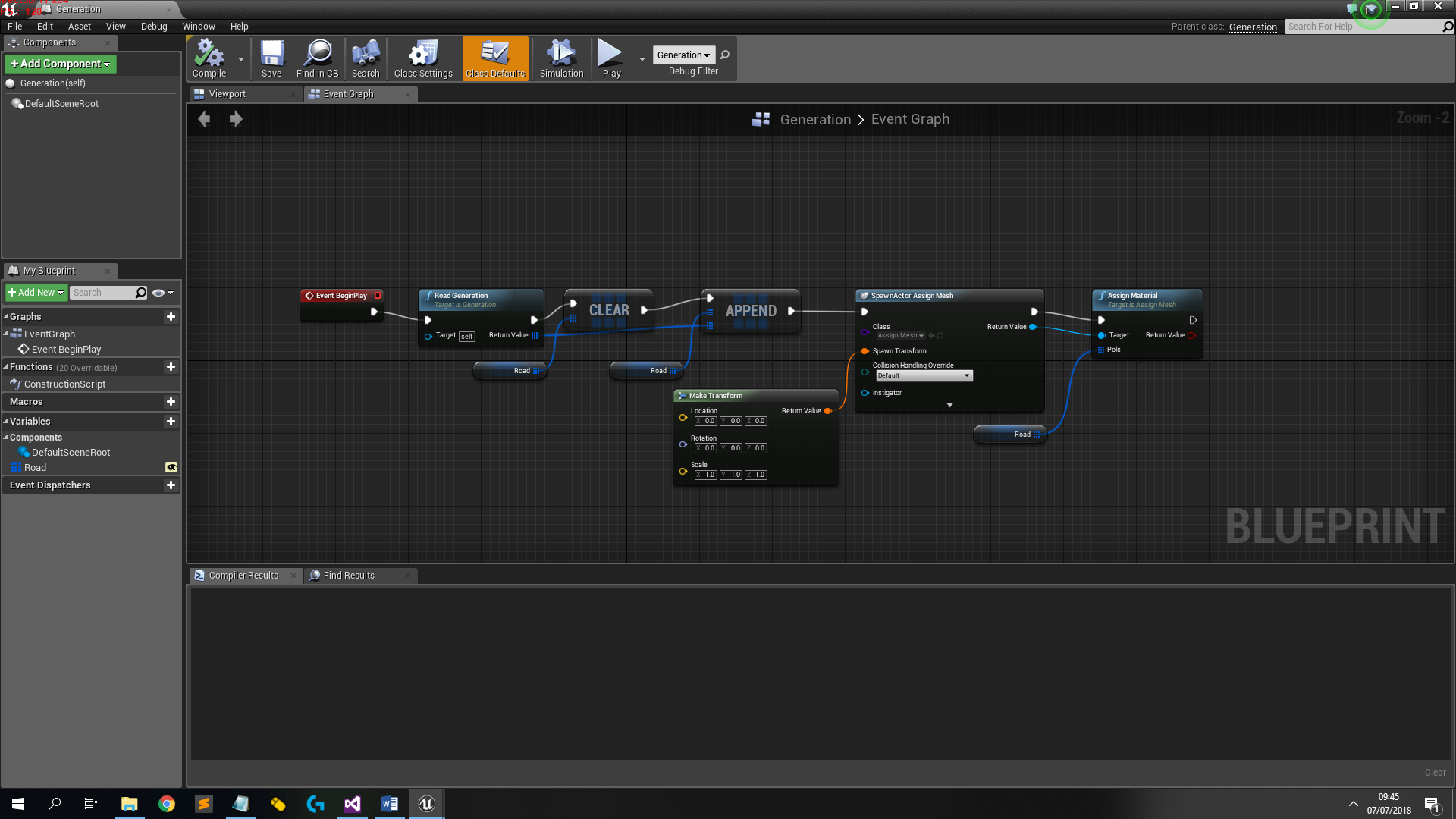
Figure 1: Architettura del sistema

In Figura 1 viene presentata un'architettura del sistema. L’algoritmo di generazione è il punto centrale del progetto, l’utente potrà visualizzare il risultato di esso attraverso il motore grafico di UE4 ma anche modificare attraverso quest’ultimo alcuni aspetti dell’algoritmo stesso.

# Blueprint Interface

Blueprints interfaces in Unreal Engine è un sistema basato sul concetto di utilizzare un'interfaccia costituita da nodi per creare elementi di gioco dall'interno di Unreal Editor e viene usato per definire classi o oggetti orientati agli oggetti (OO).

## Blueprint Generation



In Figura 2 è mostrato il Blueprint “Generation” che contiene le funzioni principali attraverso le quali si sviluppa il progetto:

* RoadGeneration contiene l’algoritmo di generazione della mappa;

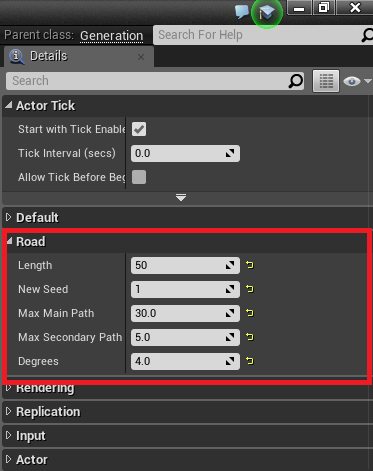
restituisce un array contenente tutti i segmenti stradali generati.

* AssignMesh e AssignMaterial sono rispettivamente una classe e un metodo appartenente ad essa utilizzati per la visualizzazione del risultato dell’algoritmo, per l’assegnazione di un materiale ai poligoni.

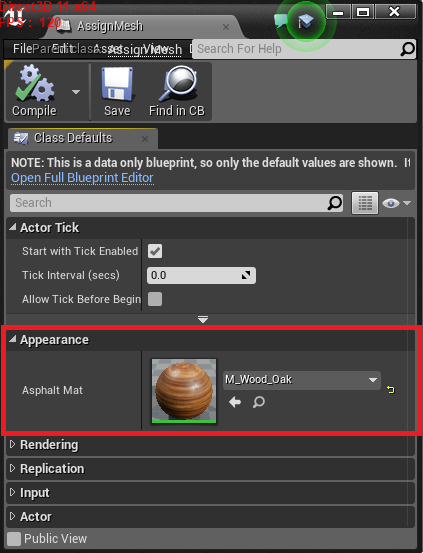
## Class Defaults(Generation) Interface

In Class Defaults, accedibile direttamente dal Blueprint “Generation” si possono modificare parametri in modo veloce:

* “Lenght” rappresenta il numero di segmenti che spawneranno sulla mappa;
* “Seed” è un valore che funziona come indicazione per la randomizzazione dell’algoritmo: al variare di esso la mappa sarà sempre diversa;
* “MaxMainPath” e “MaxSecondaryPath” influenzano molto la generazione della mappa: pongono un limite sulla generazione consecutiva dello stesso tipo di strade. Aumentando i valori si possono ottenere così una prevalenza di strade principali o secondarie e viceversa.
* “Degrees” influenza la generazione della mappa aggiungendo una rotazione alle strade in modo che le intersezioni non siano solo perpendicolari tra loro. Il valore indicato assegna una rotazione casuale compresa tra (-4, 4) gradi.



## Blueprint AssignMesh

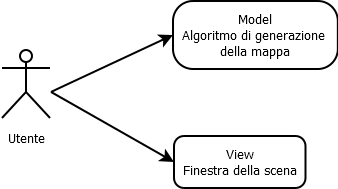


Tramite questo blueprint posso andare a modificare il materiale utilizzato per la colorazione dei poligoni.

# Use Cases

Figure 2: Use Cases

Per una definizione delle diverse funzionalità fare riferimento all'URD.



# Structural Design

## Class Diagram

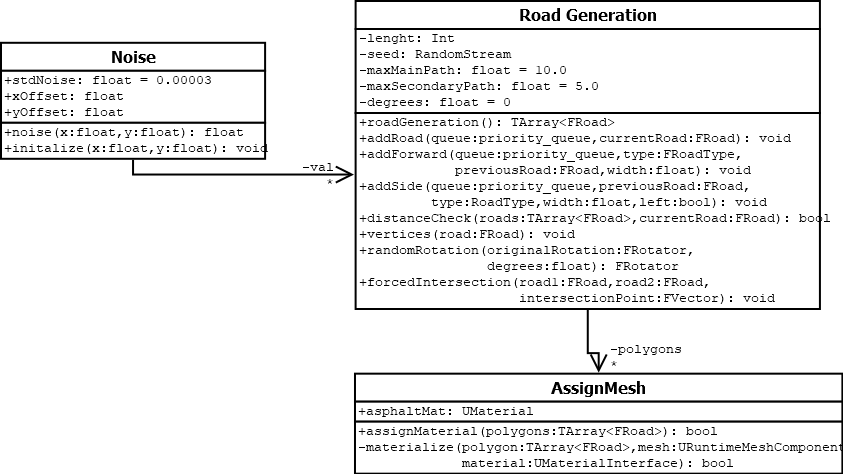


Figure 3: Class Diagram

### Road Generation

L’idea base dell’algoritmo è collocare un segmento di strada alla volta usando una coda di priorità tenendo conto che ogni volta che viene posizionato potrebbe essere anche necessario collegarlo con altre strade esistenti. Usando questa tecnica le strade principali avranno una priorità più alta di quelle secondarie cercando di costringere queste ultime ad adattarsi a quelle più grandi.

#### Attributi

totalLenght: numero totale di segmenti di strada da posizionare.

stdNoise: coefficiente del rumore per la funzione di Perlin.

seed: permette di ottenere una mappa diversa cambiandone il valore.

maxMainPath: massimo numero di segmenti di strada principale consecutivi.

maxSecondaryPath: massimo numero di segmenti di strada secondaria consecutivi.

degrees: assegna un limite massimo e minimo consentendo la rotazione di una strada in modo da non generare sempre mappe con incroci perpendicolari tra loro.

#### Metodi

roadGeneration(RoadSegment): metodo principale per la generazione della rete di strade corretta: deve ritornare l’array contenente tutte le strade che sono state posizionate.

addRoad(queue, current): viene deciso se aggiungere alla coda di priorità una strada principale o una secondaria valutando una aggiunta davanti o una a destra o una a sinistra o entrambe.

addForward(queue, previous, width, newType): aggiunta del segmento in avanti.

addSide(queue, newType, previous, width, left(bool)): deviazione a destra o a sinistra rispetto al segmento corrente(previous).

vertices(road): metodo utilizzato per individuare le estremità del segmento di strada (road) passatogli.

distanceCheck(roads, currentRoad): esclusione del segmento corrente se troppo vicino ad altri.

randomRotation(originalRotation, degrees): aggiunta di tot gradi di rotazione.

### Noise

Il rumore dovrebbe essere pensato come un’approssimazione predeterminata della densità di popolazione su diverse aree e viene utilizzato nella generazione di strade per consentire il raggiungimento di obbiettivi globali come i posizionamenti dei segmenti in modo da seguire un pattern.

L’autore della classe è Sebastien Rombauts, maggiori info nei commenti della classe Noise.cpp.

#### Attributi

stdNoise è il coefficiente del rumore di Perlin.

xOffset e yOffset sono valori che vengono passati dall’algoritmo di generazione delle strade e possono essere o le coordinate di un punto di un vettore o due numeri generati casualmente.

#### Metodi

Initialize: sfruttando il seed prescelto viene creata un’istanza del rumore passando al metodo due numeri casuali (xOffset e yOffset) in un intervallo a piacere.

Noise: attraverso addForward e addSide viene chiamato tale metodo passando come parametri le coordinate di un punto del segmento. Il valore ritornato è di tipo float compreso tra (0,1) che va ad essere sommato alla priorità del segmento considerato influenzandone la posizione all’interno della coda.

#### AssignMesh

Questa classe riceve in input l’array contenente i rettangoli stradali che sono stati scelti dall’algoritmo di generazione ed effettuando una triangolazione di essi viene assegnato un meteriale che può essere selezionabile direttamente da un blueprint apposito in Unreal Engine.

La classe si basa su strutture e metodi della classe “polypartition” il cui autore è Ivan Fratric; la stessa AssignMesh è presa da un’implementazione di polypartition, per maggiori info guardare polypartition.h.

#### Attributi

asphaltMat è di tipo UMaterial tramite esso posso gestire il materiale direttamente dall’editor UE.

#### Metodi

assignMaterial: associa al poligono il material.

materialize: effettua la triangolazione del poligono e crea il mesh.

# Dynamic Model

## Sequence Diagram, startGeneration

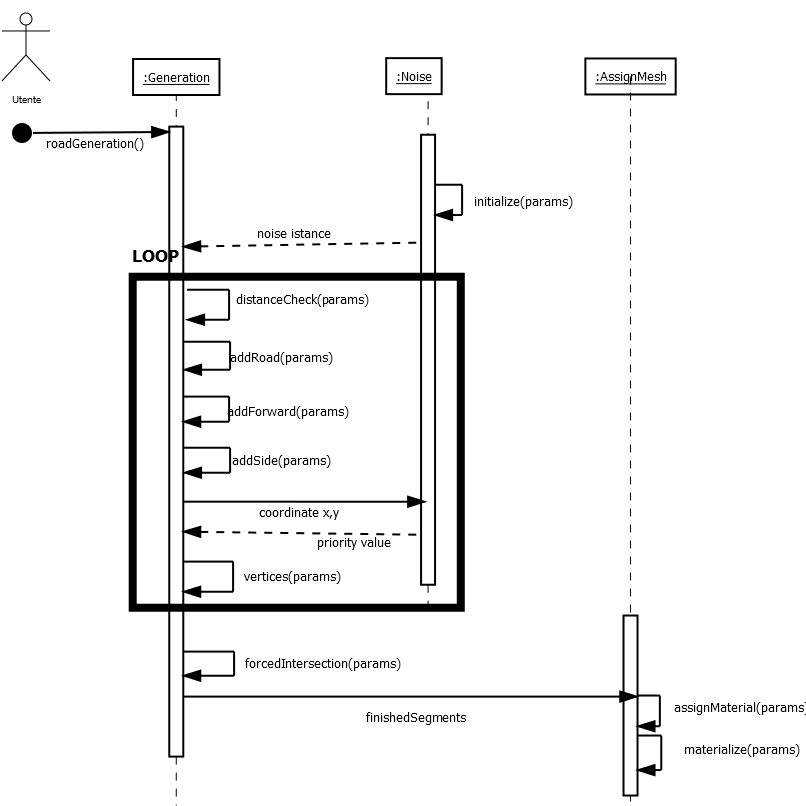


Figure 4: Sequence Diagram