True Random PRO

Leave everything to chance

Documentation

Date: 28.03.2022

Version: 2022.1.0

© 2016-2022 **cross**tales LLC

https://www.crosstales.com

Table of Contents

1. Overview....4
1.1. Why use True Random?....4
1.2. How does this differ from Unity Random?....4
2. Features....5
2.1. Generate true random....5
2.2. Documentation & control....5
2.3. Compatibility....5
3. Demonstration....6
3.1. GenerateFloat....6
3.2. GenerateInteger....6
3.3. GenerateSequence....7
3.4. GenerateString....7
3.5. GenerateVector2....8
3.6. GenerateVector3....8
3.7. GenerateVector4....9
3.8. DiceRoll....9
3.9. Magic 8-Ball....10
4. Setup....11
4.1. Add True Random....11
5. API....13
5.1. TRManager....13
5.1.1. GenerateInteger....13
5.1.2. GenerateFloat....13
5.1.3. GenerateSequence....14
5.1.4. GenerateString....14
5.1.5. GenerateVector2....14
5.1.6. GenerateVector3....14
5.1.7. GenerateVector4....14
5.1.8. Calculate Quota....15
5.2. Callbacks....16
5.2.1. Integers....16
5.2.2. Floats....16
5.2.3. Sequence....16
5.2.4. Strings....16
5.2.5. Vector2....17
5.2.6. Vector3....17
5.2.7. Vector4....17
5.2.8. Errors....17
5.2.9. Quota....17
5.2.10. Example....18
5.3. Use PRNG....18
5.4. Complete API....18
6. Quota....19
7. Third-party support (PlayMaker etc.)....19
8. Verify installation....19
9. Upgrade to new version....19
10. Important notes....20
11. Problems, improvements etc....20
12. Release notes....20
13. Credits....21

14. Contact and further information.....21
15. Our other assets.....22

Thank you for buying our asset "True Random"!

If you have questions about this asset, send us an email at truerandom@crosstales.com. Please don't forget to rate it or write a little review – it's very much appreciated.

1. Overview

1.1. Why use True Random?

"True Random" can generate random numbers for you or your game. They are "truly random", because they are generated with **atmospheric noise**, which supersedes the pseudo-random number algorithms typically use in computer programs.
People use True Random for holding drawings, lotteries and sweepstakes, to drive online games, for scientific applications and for art and music.
Here some more information regarding "true" vs. "pseudo" random:

> There are two principal methods used to generate random numbers. The first method measures some physical phenomenon that is expected to be random and then compensates for possible biases in the measurement process. Example sources include measuring atmospheric noise, thermal noise, and other external electromagnetic and quantum phenomena. For example, cosmic background radiation or radioactive decay as measured over short timescales represent sources of natural entropy.
>
> The second method uses computational algorithms that can produce long sequences of apparently random results, which are in fact completely determined by a shorter initial value, known as a seed value or key. As a result, the entire seemingly random sequence can be reproduced if the seed value is known. This type of random number generator is often called a pseudorandom number generator. This type of generator typically does not rely on sources of naturally occurring entropy, though it may be periodically seeded by natural sources. This generator type is non-blocking, so they are not rate-limited by an external event, making large bulk reads a possibility.[1]

1.2. How does this differ from Unity Random?

Perhaps you have wondered how Unity generates randomness. In reality, random numbers used in Unity are pseudo-random, which means they are generated in a predictable fashion using a mathematical formula.
This is fine for many purposes, but it may not be random in the way you expect it to be when you think of dice rolls and lottery drawings.

1 https://en.wikipedia.org/wiki/Random_number_generation#.22True.22_vs._pseudo-random_numbers

2. Features

2.1. Generate true random

- Following **data types** can be generated randomly:
  - **Integers**, **Floats** & **Sequences**
  - **Strings**
  - **Vector2**, **Vector3** & **Vector4**
- Extension methods for vectors to generate **Colors** and **Quaternions**
- **Export** the generated **results** as text file

2.2. Documentation & control

- **Test** all **random generators** in the **editor**
- Powerful **API** for **maximum control**
- Detailed **demo scenes**
- Comprehensive **documentation** and **support**
- Full **C# source code**

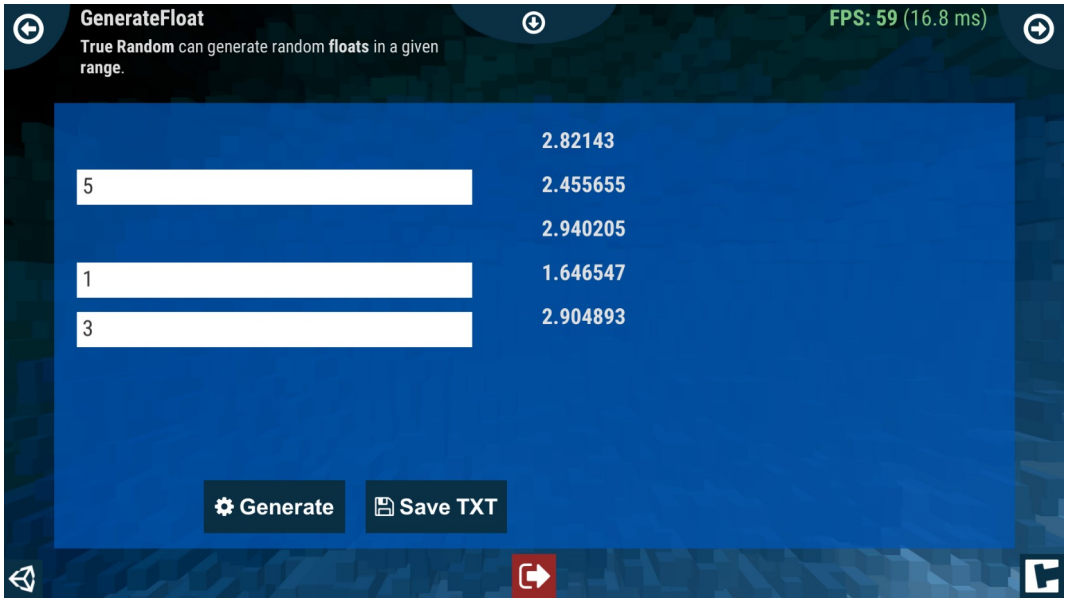
2.3. Compatibility

- Supports **all build platforms**
- Works with **Windows**, **Mac** and **Linux** editors
- Compatible with **Unity 2018.4 – 2022**
- **C# delegates** and **Unity events**
- Works with **Online Check**
- **PlayMaker** **actions**

3. Demonstration

The asset comes with many demo scenes to show the main usage.

3.1. GenerateFloat



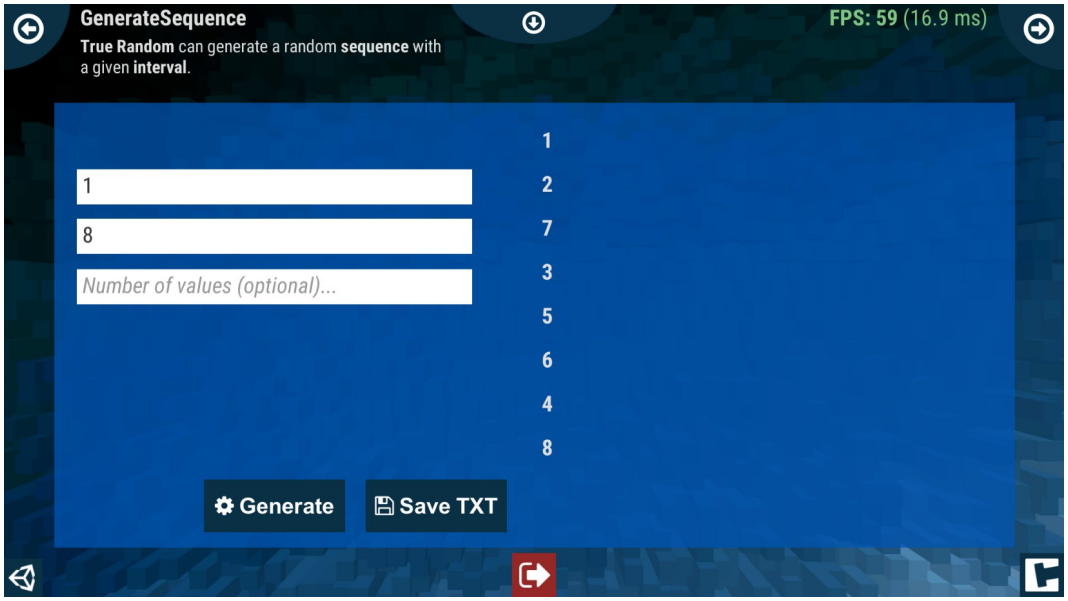
This demo scene shows how to generate floats.

3.2. GenerateInteger

This demo scene shows how to generate integers.

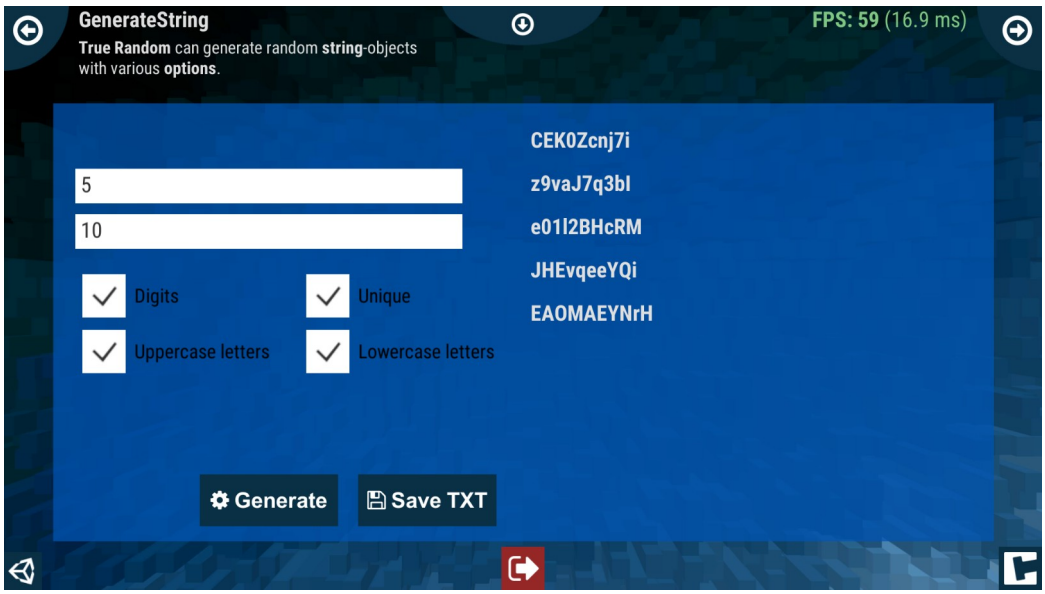
3.3. GenerateSequence

In this demo scene shows how generate sequences with a given interval.



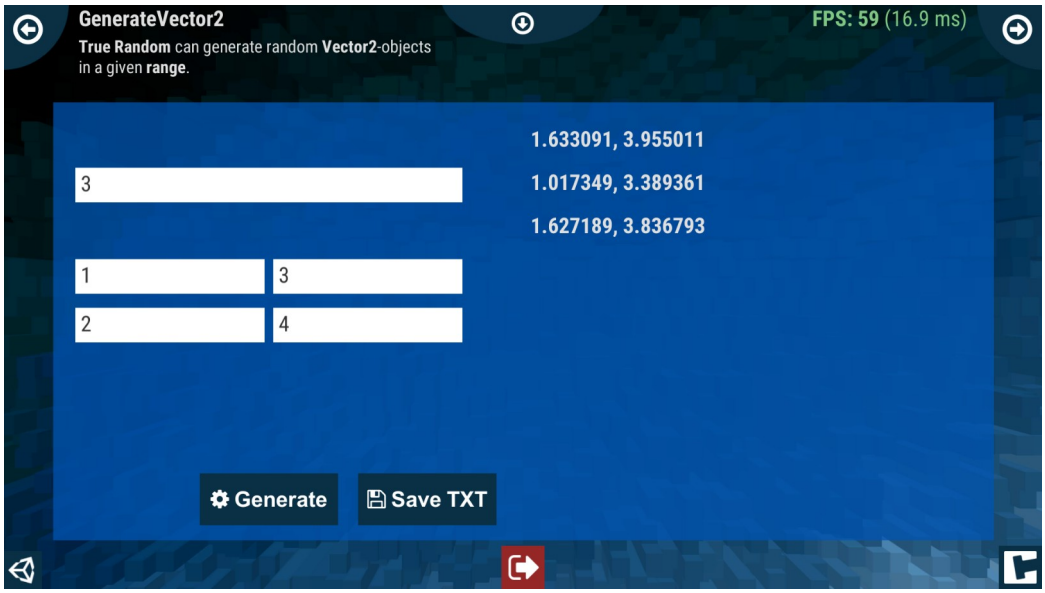
3.4. GenerateString

This scene shows how to generate strings.



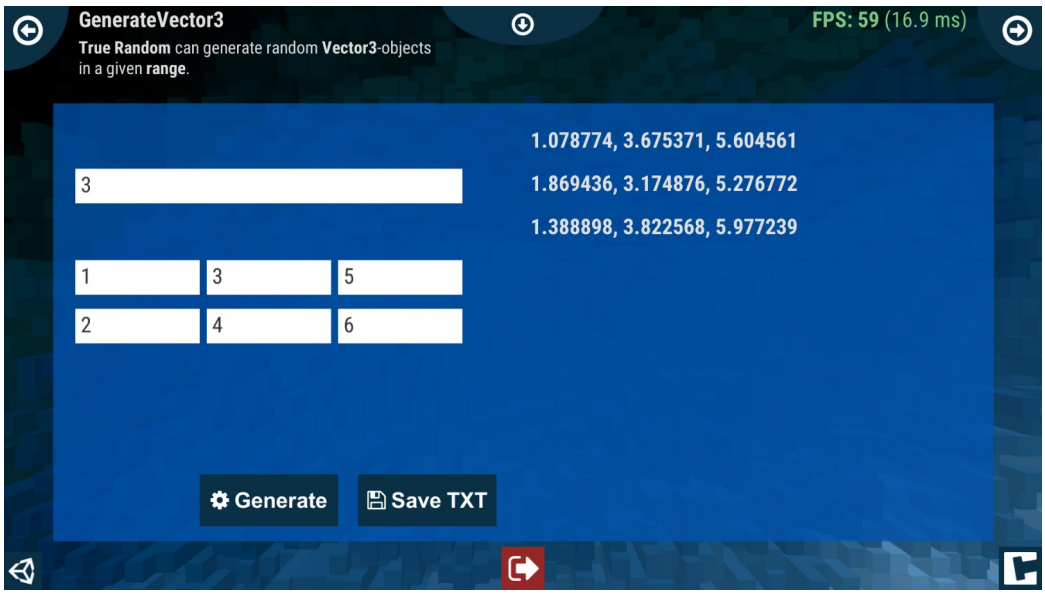
3.5. GenerateVector2

This scene shows how to generate Vector2.



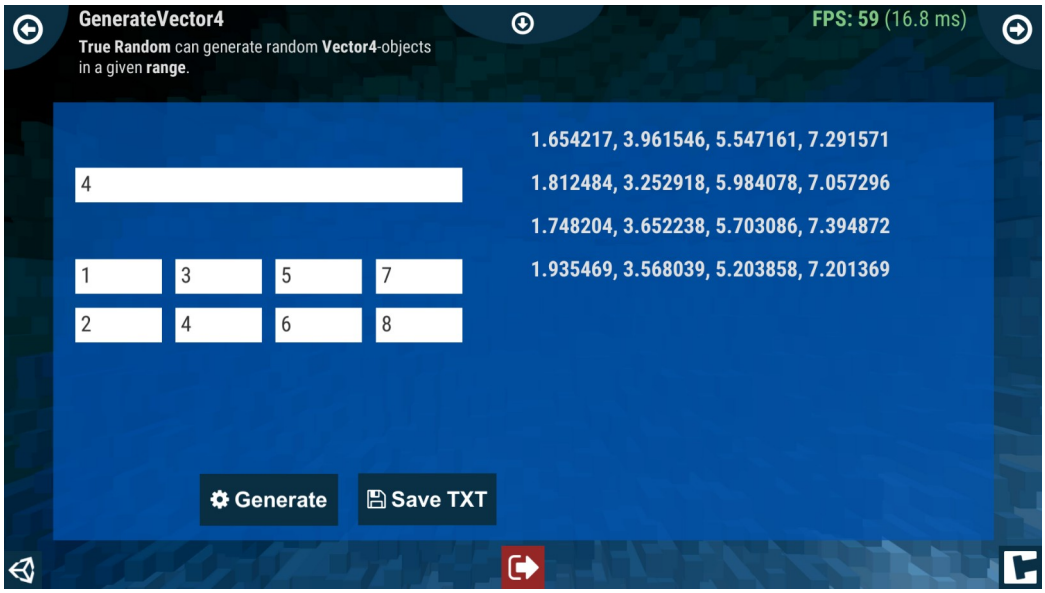
3.6. GenerateVector3

This scene shows how to generate Vector3.



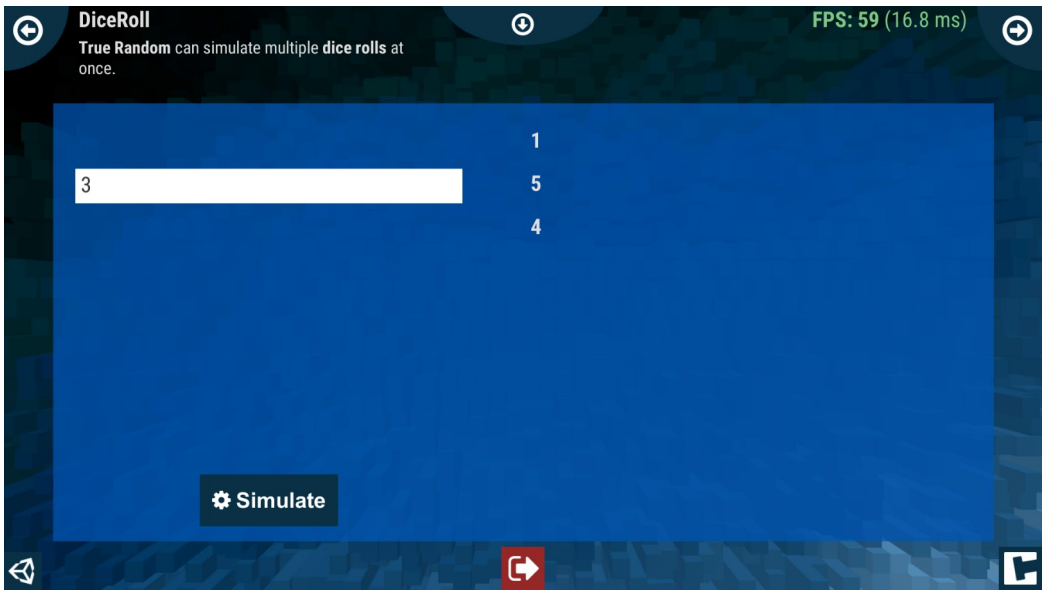
3.7. GenerateVector4

This scene shows how to generate Vector4.



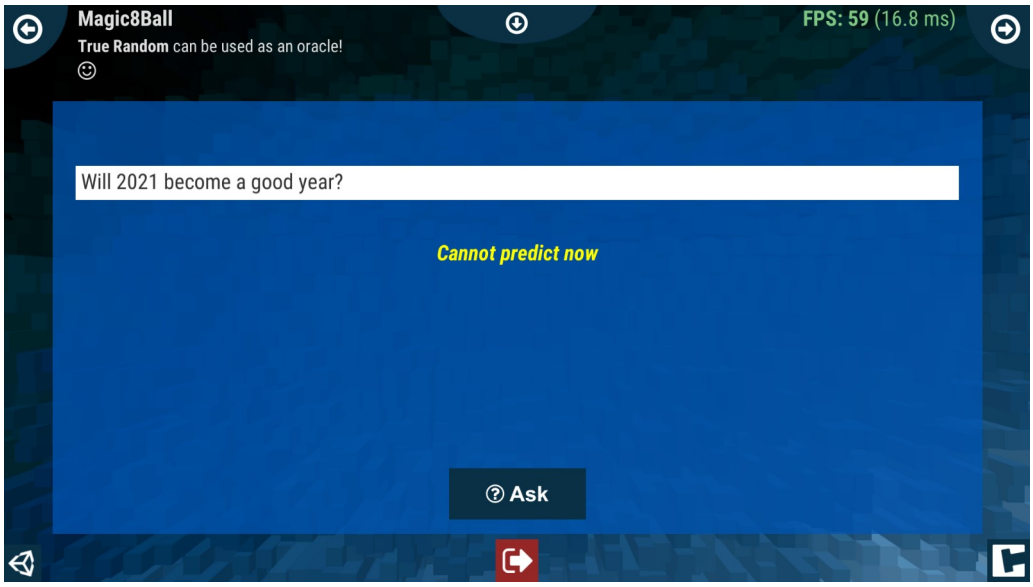
3.8. DiceRoll

This scene shows how to simulate dice rolls.



3.9. Magic 8-Ball

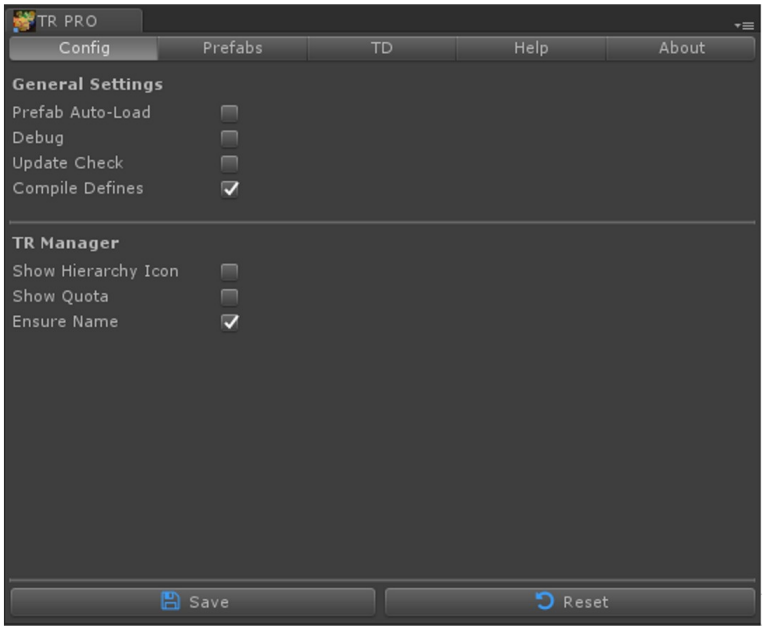
This scene shows how to use it in "Magic 8-Ball".



https://en.wikipedia.org/wiki/Magic_8-Ball

4. Setup

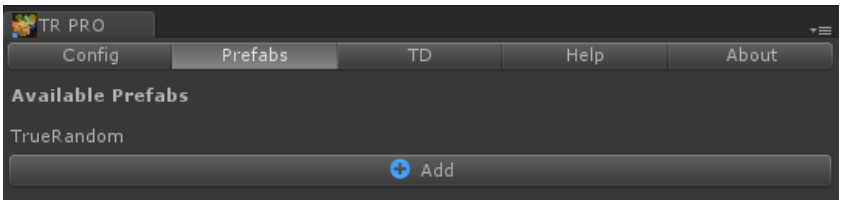
"True Random" has global settings under "Edit\Preferences..." and under "Tools\TR PRO\Configuration...":



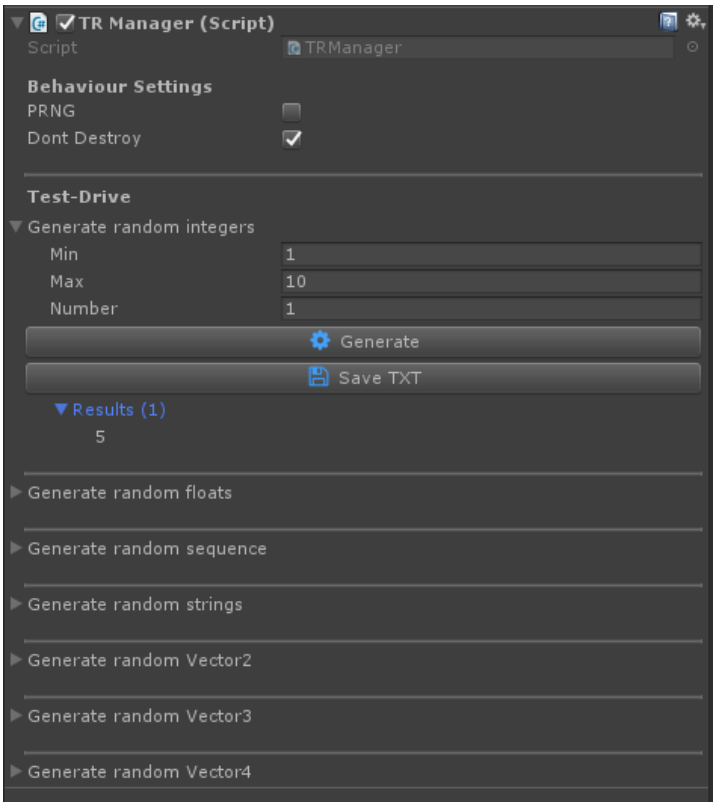
4.1. Add True Random

To add True Random manually to the project, there are two ways:

1. Add the prefab **TrueRandom** from *Assets/Plugins/crosstales/TrueRandom/Prefabs* to the scene
2. Or go to *Tools => TR PRO => Add => **TrueRandom***
3. Right-click in the *hierarchy-window => TR PRO => **TrueRandom***
4. Add it from the Prefabs-tab:



The TRManager looks like this:



5. API

The asset contains various classes and methods. The most important ones are explained here.

Make sure to **include** the **name space** in the relevant source files:

```
using Crosstales.TrueRandom;
```

5.1. TRManager

The "TRManager.cs" is a singleton and contains the following important methods.

Important: use the callbacks to get the result of the call!

Alternatively, use the properties to get the last results:

TRManager.Instance.CurrentIntegers, **TRManager.Instance.CurrentFloats**, **TRManager.Instance.CurrentSequence**, **TRManager.Instance.CurrentStrings**, **TRManager.Instance.CurrentVector2**, **TRManager.Instance.CurrentVector3** and **TRManager.Instance.CurrentVector4**

5.1.1. GenerateInteger

Generate random integers.

For example:

```
//Generates 6 random numbers between 1 and 100
TRManager.Instance.GenerateInteger(1, 100, 6);
```

5.1.2. GenerateFloat

Generate random floats.

For example:

```
//Generates 8 random numbers between 0.2 and 0.8
TRManager.Instance.GenerateFloat(0.2f, 0.8f, 8);
```

5.1.3. GenerateSequence

Generate a random sequence with a given interval.

For example:

```
//Generates a random sequence with 42 numbers (interval 1 - 42)
TRManager.Instance.GenerateSequence(1, 42);
```

5.1.4. GenerateString

Generate random string.

For example:

```
//Generates 5 random strings with a length of 10 characters
TRManager.Instance.GenerateRandomString(10, 5);
```

5.1.5. GenerateVector2

Generate random Vector2.

For example:

```
//Generates 5 random vectors between 0 and 1
TRManager.Instance.GenerateVector2(Vector2.zero, Vector2.one, 5);
```

5.1.6. GenerateVector3

Generate random Vector3.

For example:

```
//Generates 7 random vectors between 0 and 1
TRManager.Instance.GenerateVector3(Vector3.zero, Vector3.one, 7);
```

5.1.7. GenerateVector4

Generate random Vector4.

For example:

```
//Generates 3 random vectors between 0 and 1
TRManager.Instance.GenerateVector4(Vector4.zero, Vector4.one, 3);
```

5.1.8. Calculate Quota

There are various methods to calculate the needed size in bits:

```
//Calculates needed bits (from the quota) for generating 6 random integers with a maximal value of 100
int bitsForInt = TRManager.Instance.CalculateInteger(100, 6);


//Calculates needed bits (from the quota) for generating 8 random floats
int bitsForInt = TRManager.Instance.CalculateFloat(8);


//Calculates needed bits (from the quota) for generating a random sequence with an interval between 1 and 42
int bitsForSeq = TRManager.Instance.CalculateSequence(1, 42);


//Calculates needed bits (from the quota) for generating 5 random strings with a length of 10
int bitsForString = TRManager.Instance.CalculateString(10, 5);

//Calculates needed bits (from the quota) for generating 5 random Vector2
int bitsForVector2 = TRManager.Instance.CalculateVector2(5);

//Calculates needed bits (from the quota) for generating 7 random Vector3
int bitsForVector3 = TRManager.Instance.CalculateVector3(7);

//Calculates needed bits (from the quota) for generating 3 random Vector4
int bitsForVector4 = TRManager.Instance.CalculateVector4(3);

//This returns the current quota
int quota = TRManager.Instance.CurrentQuota;
```

5.2. Callbacks

There are various callbacks available. Subscribe them in the "Start"-method and unsubscribe in "OnDestroy".

5.2.1. Integers

```
GenerateIntegerStart(string id);
GenerateIntegerStart OnGenerateIntegerStart;

GenerateIntegerFinished(List<int> result, string id);
GenerateIntegerFinished OnGenerateIntegerFinished;
```

5.2.2. Floats

```
GenerateFloatStart(string id);
GenerateFloatStart OnGenerateFloatStart;

GenerateFloatFinished(List<float> result, string id);
GenerateFloatFinished OnGenerateFloatFinished;
```

5.2.3. Sequence

```
GenerateSequenceStart(string id);
GenerateSequenceStart OnGenerateSequenceStart;

GenerateSequenceFinished(List<int> result, string id);
GenerateSequenceFinished OnGenerateSequenceFinished;
```

5.2.4. Strings

```
GenerateStringStart(string id);
GenerateStringStart OnGenerateStringStart;

GenerateStringFinished(List<string> result, string id);
GenerateStringFinished OnGenerateStringFinished;
```

5.2.5. Vector2

```
GenerateVector2Start(string id);
GenerateVector2Start OnGenerateVector2Start;

GenerateVector2Finished(List<Vector2> result, string id);
GenerateVector2Finished OnGenerateVector2Finished;
```

5.2.6. Vector3

```
GenerateVector3Start(string id);
GenerateVector3Start OnGenerateVector3Start;

GenerateVector3Finished(List<Vector3> result, string id);
GenerateVector3Finished OnGenerateVector3Finished;
```

5.2.7. Vector4

```
GenerateVector4Start(string id);
GenerateVector4Start OnGenerateVector2Start;

GenerateVector4Finished(List<Vector4> result, string id);
GenerateVector4Finished OnGenerateVector4Finished;
```

5.2.8. Errors

```
ErrorInfo(string info, string id);
ErrorInfo OnErorInfo;
```

5.2.9. Quota

```
UpdateQuota(int quota);
UpdateQuota OnUpdateQuota;
```

5.2.10. Example

Get informed when the generation of a sequence starts and finishes:

```
void OnEnable() {
 // Subscribe event listeners
 TRManager.Instance.OnGenerateSequenceStart += seqStartMethod;
 TRManager.Instance.OnGenerateSequenceFinished += seqFinishedMethod;

 TRManager.Instance.GenerateSequence(1, 10);
}

void OnDisable() {
 // Unsubscribe event listeners
 TRManager.Instance.OnGenerateSequenceStart -= seqStartMethod;
 TRManager.Instance.OnGenerateSequenceFinished -= seqFinishedMethod;
}

private void seqStartMethod(string id) {
 Debug.Log("seqStartMethod: " + id);
}

private void seqFinishedMethod(List<int> result, string id) {
 Debug.LogWarning("seqFinishedMethod: " + result.Count);

 // do something with the result!
}
```

5.3. Use PRNG

There are also methods for every generator with standard PRNG which needs no quota. There are two possibilities:

1. Enable PRNG globally: `TRManager.Instance.PRNG = true;`
2. Call the methods which ends with "PRNG", like *GenerateVector2PRNG*

5.4. Complete API

For more details, please see the TrueRandom-api.pdf

6. Quota

True Random has a quota limitation of 1'000'000 random bits per IP-address in 24 hours; this allows to generate:

- min. 32'000 integers (depends on the integer size)
- min. 3'200 sequences (interval of 10 elements)
- min. 3'200 strings (length of 10 chars, depends on the settings)

If the quota expires, Unity's pseudo-random will be used automatically.

7. Third-party support (PlayMaker etc.)

"True Random PRO" supports various assets from other publishers. Please import the desired packages from "Assets/Plugins/crosstales/TrueRandom/3rd party".

8. Verify installation

Check if True Random is installed:

```
#if CT_TR
    Debug.Log("True Random installed: " + Util.Constants.ASSET_VERSION);
#else
    Debug.LogWarning("True Random NOT installed!");
#endif
```

9. Upgrade to new version

Follow this steps to upgrade the version of "True Random PRO":

1. Update "True Random PRO" to the latest version from the "Unity AssetStore"
2. Inside the project in Unity, go to menu "File" => "New Scene"
3. Delete the "Assets/Plugins/crosstales/TrueRandom" folder from the Project-view
4. Import the latest version downloaded from the "Unity AssetStore"

10. Important notes

After this setup, the "True Random" is ready to use. It is important to know that it uses the **singleton**-pattern, which means that **once instantiated**, "True Random" will **live until** the application is **terminated**.

- Be patient, "True Random" is using a service from Random.org. They deliver huge amounts of randomly generated data to a lot of people. So generating your random stuff needs a little bit time.
- Do not send more than one generating request at once. Otherwise you risk that none of them return numbers.
- "True Random" is not meant to replace Unity's pseudo-random. It's just for different needs.
- You start with a quota of 1'000'00 bits every 24 hours. Every bit you use will reduce your quota. If your quota expires, Unity's pseudo-random will be used automatically.

11. Problems, improvements etc.

If you encounter any problems with this asset, just send us an email with a problem description and the invoice number and we will try to solve it.

12. Release notes

See "VERSIONS.txt" under "Assets/Plugins/crosstales/TrueRandom/Documentation" or online:

https://crosstales.com/media/data/assets/truerandom/VERSIONS.txt

13. Credits

"True Random" uses the API from https://www.random.org/.

We aren't affiliated to this company.

The icons are based on Font Awesome.

14. Contact and further information

crosstales LLC

Schanzeneggstrasse 1

CH-8002 Zürich

Homepage:	https://www.crosstales.com/en/portfolio/truerandom/
Email:	truerandom@crosstales.com
AssetStore:	https://assetstore.unity.com/lists/crosstales-42213
Forum:	https://forum.unity.com/threads/true-random-pro-real-randomness-for-unity.457277/
Documentation:	https://www.crosstales.com/media/data/assets/TrueRandom/TrueRandom-doc.pdf
API:	https://www.crosstales.com/media/data/assets/truerandom/api/
WebGL-Demo:	https://www.crosstales.com/media/data/assets/truerandom/webgl/

15. Our other assets

 3D Skybox	Those beautiful packages contain professional 8k, HDR, stereoscopic 360° real-world skyboxes for your projects.
 Bad Word Filter	The "Bad Word Filter" (aka profanity or obscenity filter) is exactly what the title suggests: a tool to filter swearwords and other "bad sentences".
 DJ	DJ is a player for external music-files. It allows a user to play his own sound inside any Unity-app. It can also read ID3-tags.
 File Browser	File Browser is a wrapper for native file dialogs on Windows, macOS, Linux and UWP (WSA).
 Online Check	You need a reliable solution to check for **Internet availability**? Here it is!
 Radio	Radio allows implementing free music from Internet radio stations into your project..
 RT-Voice	RT-Voice uses the computer's (already implemented) TTS (text-to-speech) voices to turn the written lines into speech and dialogue at run-time! Therefore, all text in your game/app can be spoken out loud to the player.
 Turbo Backup	Turbo Backup is the fastest and safest way to backup your Unity project. It only stores the difference between the last backup, this makes it incredible fast.
 Turbo Builder	Turbo Builder creates builds for multiple platforms in one click. It works together with Turbo Switch to offer an incredible fast build pipeline.

 Turbo Switch	Turbo Switch is a Unity editor extension to reduce the time for assets to import during platform switches. We measured speed improvements up to 100x faster than the built-in switch in Unity.