Documentation

Required Packages

- TextMeshPro

How To Use

1. Add the package to your Unity project.
2. Open the example scene named Name Generator in the scenes folder.
3. Click on play, and when you enter a number and press Generate it will generate a list of names.
4. To add this to your game, all you need to do is add the NameGenerator script into your game, and call the StartGeneration from somewhere, and set the variables as desired.

Main Scripts

NameGenerator

This script is in chage of actually generating all the names given the set parameters in the inspector.

FinishedNameGeneration

This is a Unity Event that is called once it is done generating all the different names. It takes in a List of strings to pass through the event.

_consonants

An array of consonants.

_vowels

An array of vowels.

nameLengthMin

The minimum length of a name.

nameLengthMax

The maximum length of a name.

numberOfNames

How many names you want it to create.

logNames

If you want it to log all the names generated to the console or not.

removeObscenities

If you want it to remove the obscenities or not.

_names

A list of all the names that are generated.

SetAmountToGenerate(int numberToSpawn)

This sets numberOfNames equal to the numberToSpawn that is passed in.

SetAmountToGenerate(string numberToSpawn)

This sets numberOfNames equal to numberToSpawn if it isn't equal to nothing. If it is equal to number it defaults to 100.

StartGeneration

This clears the _names list then invokes FinishedNameGeneration. It then starts a coroutine called GenerateNames.

GenerateNames

While the count of _names is less than numberOfNames it will keep generating names using the other GenerateNames function, passing in the nameLengthMin and Max and sets name equal to the result. It then Tests the name and then starts again after waiting a short amount of time.

TestNames(string name)

This tests the name to see if it meets the quiteria, I know it passes in the min and max length of the name, but we can still get names longer than this. It also tests to see if the name consists of any characters that aren't letters. It then looks to see if the name generated matches any of the words in the Obscenes list and returns if its true, so it isn't added to the list. The last test it does is to see if the name is already in the list or not. If its not already in the list it adds it.

GenerateNames(int lengthMin, int lengthMax)

This function is named the same as the other function, but it also takes 2 parameters which are the min and max length of the word. It first creates a new list of char which is just a list of characters. It then generates a random number between the min and max plus 1 as the max is exclusive but min is inclusive and sets it to length. Then creates a variable called nextChar which is of type char.

It then loops through a for loop for the amount of times specified by length. There is a new variable called isVowel which just stores if the next char should be a vowel or consonant. If its a vowel it will randomly selects a vowel if the name is longer than 1 char it then checks to see if the previous char is also the same vowel, if it is then it keeps looping until it isn't true anymore, then sets nextChar to that character.

If isVowel is false it just selects a random consonant from the array and sets nextChar to that character.

It then goes through a series of checks to see if it follows some basic rules of the English language, such as i before e except after c. This isn't perfect as some English words don't follow this but it's to get understanding of rules. If the rules are all fine. it then checks if the last letter is v or j. If it is, it then adds a e to the end to follow more English rules. (The tests we spoke of earlier will catch this, as if it adds an e, the word might not be the desired length anymore and should be removed.

Once its done it then adds each char to a string and returns it.

UpdateUI

This just updates the UI by clearing any previous names and then updating it with the new list of names.

content

This is just a reference to the content gameObject in the scroll view.

textFieldPrefab

This is a reference to the text prefab that will be used when it creates the list of names it generates.

contentList

This is a list of all the gameObjects it creates to display the names.

AddNameListToUI(List<string> nameList)

It first gets passed a list of strings called nameList which it checks to see if the count is greater then 0. Then for each name it creates a gameObject sets its name, and then gets the TMP_InputField sets it to read only and then sets the text to the name that was passed in by the list. It then adds it to the contentList.

If there are no names in the list, it just clears the UI. Destroying all the gameObjects.

Obscenities

This is just is a static class which holds a list of bad words.

Obscenes

This is just a really long list of all the bad words that shouldn't be allowed to pass through.