TheHangingHouse.Utlity Namespace

Contains a function that makes programmers avoid repetitive work, it’s covered several working scopes, but the majority of it is with arrays.

Clases

Util	Provides helping functions.
UtilExtensions	Provides extensions functions.

Remarks

Note that these functions are not recommended to be used in complicated math calculations in high fps needs, for example if you have calculations in real time game, and you need to calculate every vertex in mesh by looking into another arrays, this functions might affect the performance, one of good use cases is (data processing, debugging, filtering, ….)

TheHangingHouse.Utlity.Util Class

Definition

Namespace: TheHangingHouse.Utility.Util
Assembly: TheHangingHouse.Utility

Util class contains some useful functions randomly from different concepts.

Static Methods

object Parse(string txt, System.Type type)	Parsing text to a given type, Note that if the (type) doesn’t have a parse function, the Parsing function will not work, and the result will be the default instance of (type). returning object type.
T Parse<T>(string txt)	Parsing text to a given type, Note that if the (type) doesn’t have a parse function, the Parsing function will not work, and the result will be the default instance of (type). returning <T> type

TheHangingHouse.Utility.UtilExtensions

Definition

Namespace: TheHangingHouse.Utility.Util
Assembly: TheHangingHouse.Utility

The majority of functions inside this class are to make working with arrays and collections much easier and clear to eyes.

Static Methods

string Read<T>(this IEnumerable<T> ienumerable, string inbetween = ", ")	Conert given group of elements into string and make sure to include (inbetween variable) between elements. inbetween variable might be any character. for example if you want each element to be in separate lines, put inbetween variable as \n.
T Reduce<T>(this IEnumerable<T> ienumerable, System.Func<T, int, T, int, T> func)	Apply operator between each item in group and return the result as T type. The operation is based to (elements[i - 1], i - 1, elements[i], i).
T Reduce<T>(this IEnumerable<T> ienumerable, System.Func<T, T, T> func)	Apply operator between each item in group and return the result as T type. The operation is based to (elements[i - 1], elements[i]).
T[] Resize<T>(this T[] arr, int newLength, out T[] outItems)	Resize an array. Incase of newLength is less than array.CurrentLength, (outItems) will be the removed items.
bool Check<T>(this IEnumerable<T> ienumerable, System.Func<T, int, bool> func)	Check if one of element satisfies given condition. bool func(T element, int i) will be applied to all elements, The function will be true if at least one of the elements satisfies func, otherwise the result will be false.

bool Check<T>(this IEnumerable<T> ienumerable, System.Func<T, bool> func)	Check if one of element satisfies given condition. bool func(T element) will be applied to all elements, The function will be true if at least one of the elements satisfies func, otherwise the result will be false.
TOutput[] Map<TInput, TOutput>(this TInput[] arr, System.Func<TInput, int, TOutput> func)	(Map) function will go through all elements in (arr) and apply TOutput func(T element, int i) on them, After that, put the result into a new array and return it.
TOutput[] Map<TInput, TOutput>(this TInput[] arr, System.Func<TInput, TOutput> func)	(Map) function will go through all elements in (arr) and apply TOutput func(T element) on them, after that put the result into new array and return it.
List<TOutput> Map<TInput, TOutput>(this List<TInput> list, System.Func<TInput, int, TOutput> func)	(Map) function will go through all elements in (list) and apply TOutput func(T element, int i) on them, after that put the result into a new list and return it.
List<TOutput> Map<TInput, TOutput>(this List<TInput> list, System.Func<TInput, TOutput> func)	(Map) function will go through all elements in (list) and apply TOutput func(T element) on them, after that put the result into new list and return it.
T[,] Map<T>(this T[,] grid, System.Func<T, int, int, T> func)	(Map) function will go through all elements in (grid) and apply TOutput func(T element, int i, int j) on them, after that put the result into a new 2D array of type T and return it.
T[,] Map<T>(this T[,] grid, System.Func<T, T> func)	(Map) function will go through all elements in (grid) and apply TOutput func(T element) on them, after that put the result into a new 2D array of type T and return it.
T[,] Map<T>(this T[,] grid, System.Func<int, int, T> func)	(Map) function will go through all elements in (grid) and apply TOutput func(int i, int j) on them, after that put the result into a new 2D array of type T and return it.
void Apply<T>(this T[] arr, System.Action<T, int> action)	(Apply) function will go through all elements in (arr) and apply void action(T element, int i) on them.
void Apply<T>(this T[] arr, System.Action<T> action)	(Apply) function will go through all elements in (arr) and apply void action(T element) on them.

void Apply<T>(this List<T> list, System.Action<T, int> action)	(Apply) function will go through all elements in (list) and apply void action(T element, int i) on them.
void Apply<T>(this List<T> list, System.Action<T> action)	(Apply) function will go through all elements in (list) and apply void action(T element) on them.
void Apply(this Transform element, System.Action<Transform, int> action)	(Apply) function will go through all childrens of (element) and apply void action(Transform element, int i) on them.
void Apply(this Transform element, System.Action<Transform> action)	(Apply) function will go through all childrens of (element) and apply void action(Transform element) on them.
void Apply<T>(this T[,] grid, System.Action<T, int, int> action)	(Apply) function will go through all elements in (grid) and apply void action(T element, int i, int j) on them.
void Apply<T>(this T[,] grid, System.Action<int, int> action)	(Apply) function will go through all elements in (grid) and apply void action(int i, int j) on them.
T[] Filter<T>(this T[] arr, System.Func<T, int, bool> func)	The result will be elements that make bool func(T element, int i) return true.
T[] Filter<T>(this T[] arr, System.Func<T, bool> func)	The result will be elements that make bool func(T element) return true.
List<T> Filter<T>(this List<T> list, System.Func<T, int, bool> func)	The result will be elements that make bool func(T element, int i) return true.
List<T> Filter<T>(this List<T> list, System.Func<T, bool> func)	The result will be elements that make bool func(T element) return true.
string Filter(this string str, System.Func<char, int, bool> func)	The result will be characters that make bool func(string txt, int i) return true.
string Filter(this string str, System.Func<char, bool> func)	The result will be characters that make bool func(string txt) return true.
string AddSpaces(this string text)	(AddSpaces) function will add space before any big case character and return the result.

T Search<T>(this T[] arr, System.Func<T, int, bool> func)	(Search) function will apply func(T element, int i) to (arr) elements, The result will be the first element that satisfies bool func(T element, int i).
T Search<T>(this T[] arr, System.Func<T, bool> func)	(Search) function will apply func(T element) to (arr) elements, The result will be the first element that satisfies bool func(T element).
T Search<T>(this List<T> list, System.Func<T, int, bool> func)	(Search) function will apply func(T element, int i) to (list) elements, The result will be the first element that satisfies bool func(T element, int i).
T Search<T>(this List<T> list, System.Func<T, bool> func)	(Search) function will apply func(T element) to (list) elements, The result will be the first element that satisfies bool func(T element).
T[] Find<T>(this T[] arr, System.Func<T, int, bool> func)	(Find) function will go through all elements in (arr). The result will be all elements that satisfy bool func(T element, int i).
T[] Find<T>(this T[] arr, System.Func<T, bool> func)	(Find) function will go through all elements in (arr). The result will be all elements that satisfies bool func(T element).
List<T> Find<T>(this List<T> list, System.Func<T, int, bool> func)	(Find) function will go through all elements in (list). The result will be all elements that satisfies bool func(T element, int i).
List<T> Find<T>(this List<T> list, System.Func<T, bool> func)	(Find) function will go through all elements in (list). The result will be all elements that satisfies bool func(T element).
T Last<T>(this T[] arr)	Last element in (arr).
T Last<T>(this List<T> list)	Last element in (list).
byte[] Encode(this string stringToEncode)	Encodes all the characters in the specified string into a sequence of bytes.
bool ImplmentsInterface(this System.Type type, System.Type interfaceType)	Check if (type) inherits from (interfaceType).
void CopyToClipboard(this string text)	Coppy (text) to the clipboard.
void Delay(this MonoBehaviour monoBehaviour, System.Action action, float	Call action after (seconds), Note that delay is done by using coroutines, so if you call

seconds = 1f)	StopAllCoroutines() the action you passed will not work.
void DelayFrames(this MonoBehaviour monoBehaviour, System.Action action, int frames = 1)	Call action after (frames).Note that delay is done by using coroutines, so if you call StopAllCoroutines() the action you passed will not work.