Simple Man: Async operations

Start your coroutines and async operations by single line of code! You don't need to write IEnumerators and custom callbacks anymore. This pack already have default templates like 'Delay', 'Wait until', 'Repeat while' and other.

Author: Igor-Valerii Chebotar
Email: igor.valerii.chebotar@gmail.com

MonoBehavior methods

Function name	Description
Delay	Calls target method after delay
DelayRealtime	Calls target method after delay in real time
SkipFrames	Calls target method after certain number of frames
WaitUntill	Calls target method after condition completed
WaitWhile	Calls target method after condition failed
RepeatUntill	Calls target method every tick while condition is not complete
RepeatWhile	Calls target method every tick while condition is complete
RepeatForever	Calls target method every tick forever (can be stopeed manually)

C# Examples

```
//Call 'DoAction' method after 3 seconds
this.Delay(3, DoAction);
```

```
//Call 'DoAction' method after 3 seconds with parameter
this.Delay(3, ( ) => DoAction("Hello!"));
```

```
//Call 'Tick' method while 'isRunning' flag is true
this.RepeatWhile(() => isRunning, Tick);
```

```
//Call 'Tick' method every 3 seconds while 'isRunning' flag is true
this.RepeatWhile(() => isRunning, Tick, 3);
```

Safe async methods (for advanced users)

Use the "SafeAsync" static class to make asynchronous operations safe at runtime in non-mono behavior classes. It works just like normal C# asynchronous operations, but with a check that Unity is in run mode and the project or build is not paused. This operation will automatically stop when Unity switches to editor mode, so you don't have to check it manually. It also contains standard templates for asynchronous operations with callbacks and provides the ability to cancel an asynchronous operation without cancellation tokens and the Try-catch constructions.

Use async operations carefully! Try to use coroutines instead of async if it possible.

C# Examples

```
//Call 'DoAction' method after 3 seconds in async method, of cource
await SafeAsync.Delay(3);
DoAction();
```

```
//You can check result of async operation. It can be useful for checking that is
//not interruped by system (switching to editor mode). So if it was interrupted by system,
//you should finish your async operation as soon as posible, because if you ignore this rule,
//the code below will be executed in editor mode.
EAsyncOperationResult result = await SafeAsync.Delay(3);
if(result == EAsyncOperationResult.CanceledBySystem)
    return;

DoAction();
```