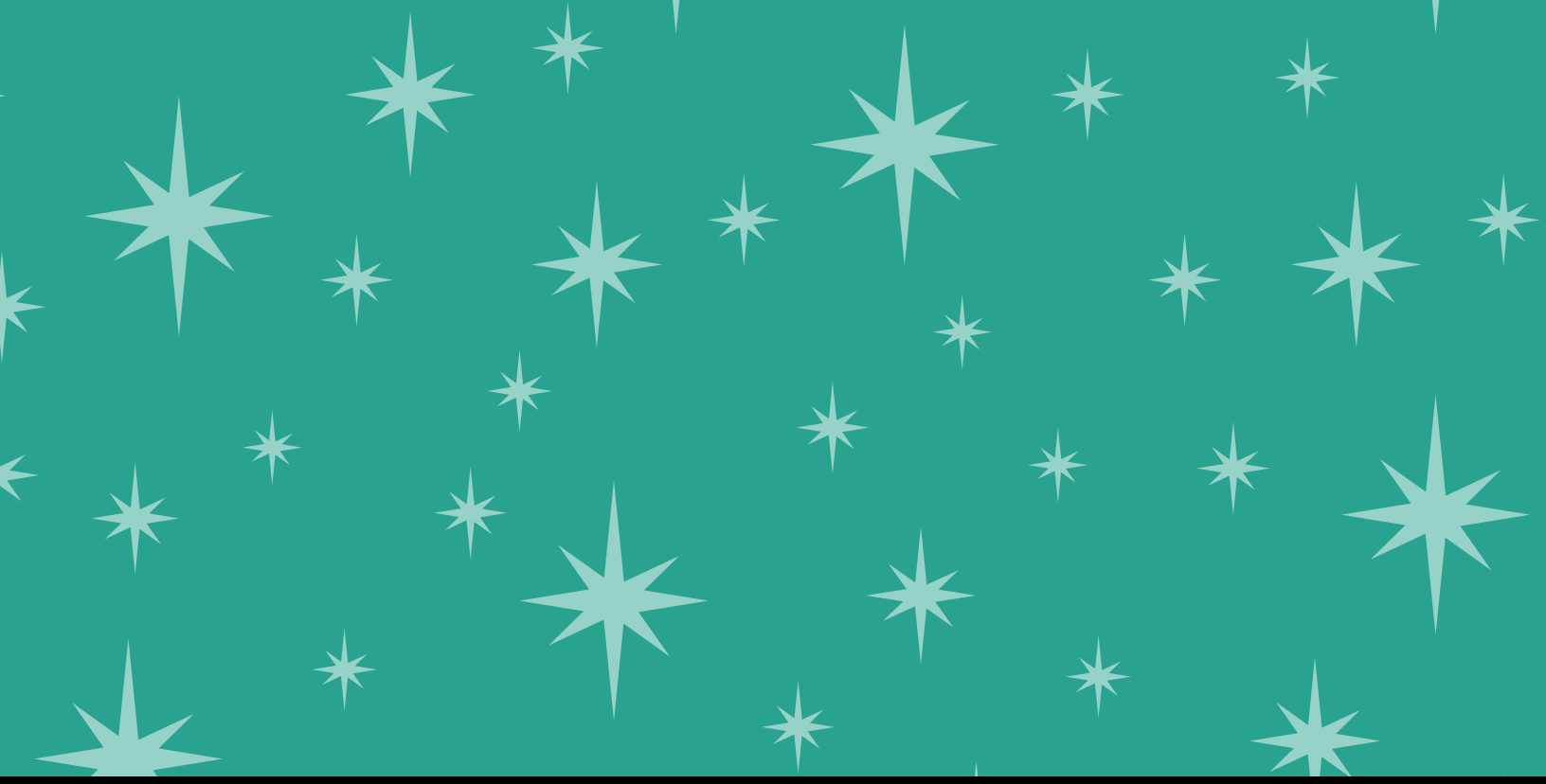
GameMaker Unit Testing Library

USER MANUAL

Atomic City, LLC
1.0.0



Introduction

1 Introduction

The purpose of this unit testing library is to make it as easy as possible to test GameMaker scripts written in GML. The API for this library closely resembles that of unit testing libraries for other programming languages.

Unit testing and this library are most helpful when a project has a number of GML scripts and care must be made that they perform as expected with any input. It is also useful for regression testing when making changes to the scripts.

The goal of this library is to provide valuable testing features while causing the least amount of disruption to a GameMaker project.

This Manual

The latest version of this manual can be found online at http://atomic.city/products/gamemaker-unit-testing-library/.

1.1 What's New

Version 1.0.1:

- Initial full release
- Configurable output to either the debug console or overlaid on a room
- A simple API with just a few functions that can fulfill all unit testing needs
- A fully documented API

1.2 System Requirements

This package requires GameMaker Studio 2. It has not been tested with earlier versions.

1.3 Installation

The Unit Testing Library can be found in the GameMaker marketplace here. The library is installed via the GameMaker marketplace. From the marketplace tab in GameMaker Studio, the library can be searched for, added to the user's library, and then downloaded and installed.

After the package has been downloaded and imported, the resource tree should contain the following items:

- Scripts

- ac_assert_true
- ac_assert_false
- ac_assert_equal
- ac_unit_tests
- _ac_add_test_result

- Objects
  - oUnitTests
- Rooms
  - rUnitTests

1.4 Library Conventions

Atomic City libraries for GameMaker Studio have some conventions to make their usage as simple as possible while working within the abilities of GML.

Atomic City Functions

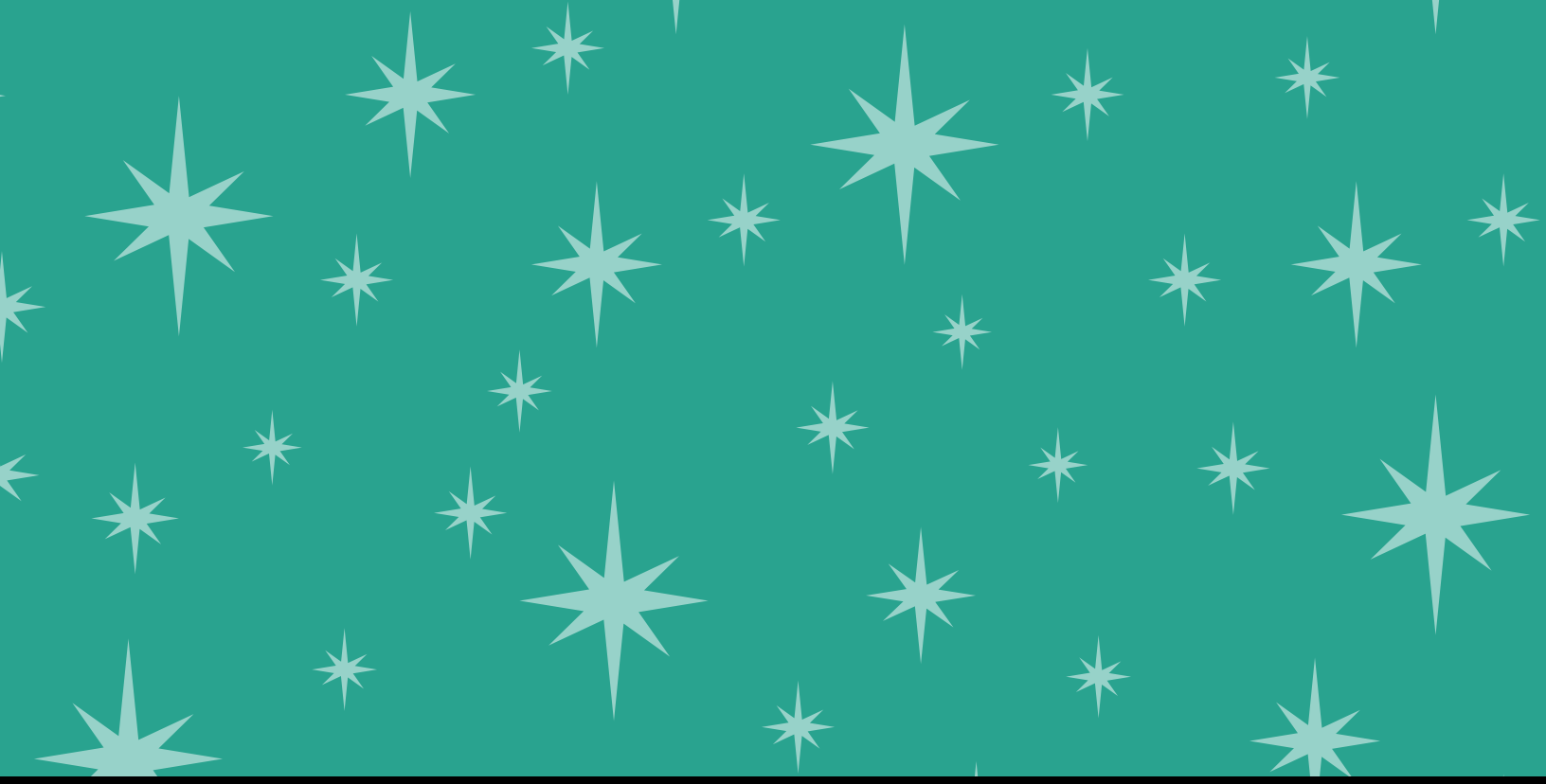
All Atomic City functions begin with the `ac_` prefix. This is because GML doesn't have true name-spacing. This prefix allows the library's functions to coexist with other similarly-named functions.

Non-Public Functions

Any script that starts with an underscore (_) is not meant to be used by end users of the libraries. These functions should be considered private to the library. GML doesn't offer a way to hide private scripts, so Atomic City differentiates them with the prefixed underscore. The behavior of these non-public scripts is not guaranteed to work the same from one version to the next. Calling these scripts is not supported and may result in undefined behavior.

IDE Documentation

All Public functions in Atomic City's libraries are documented so that parameter help should show in the script editor in GameMaker Studio if the user has that option enabled.



Usage

2 Usage

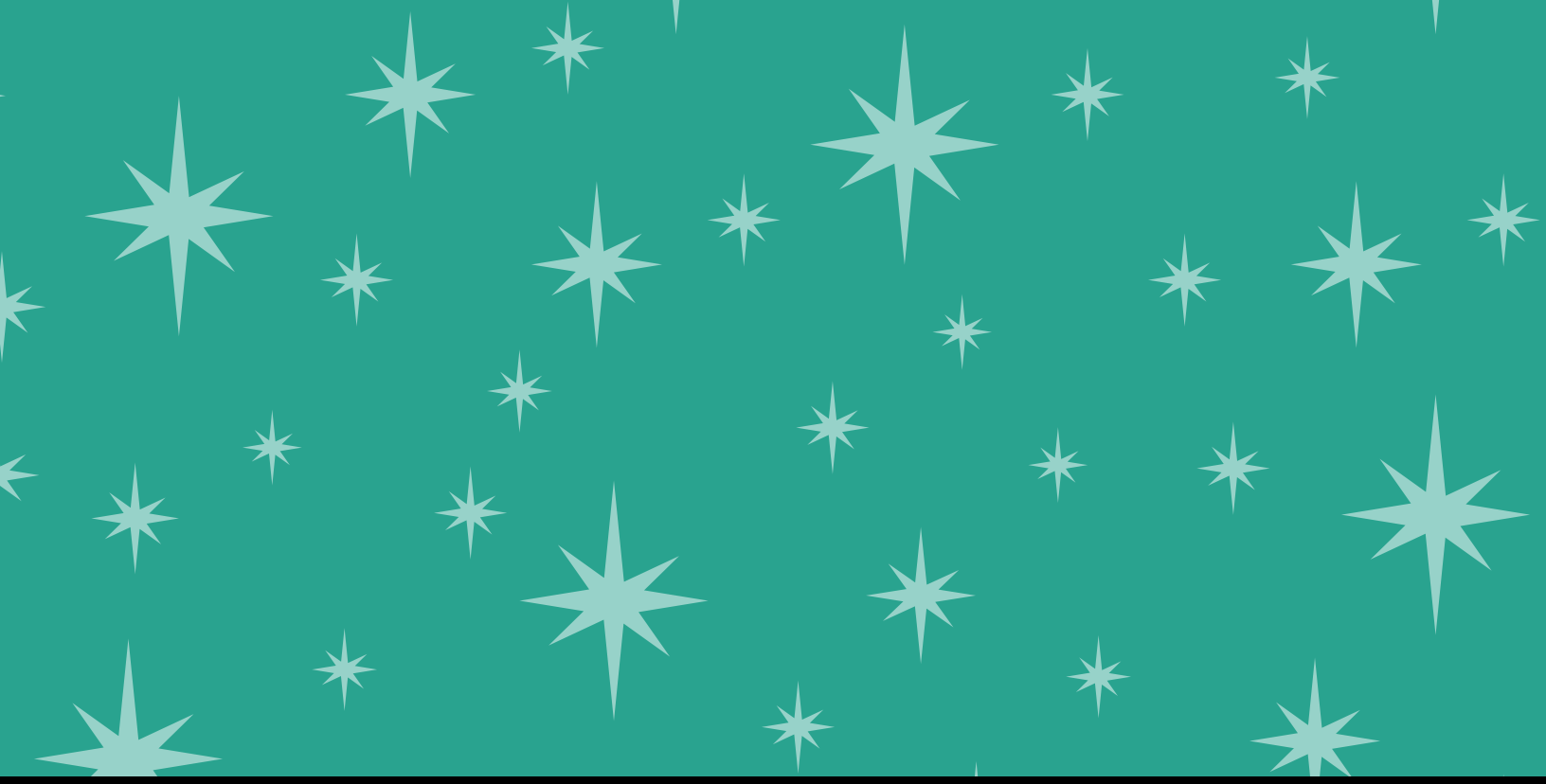
Because the goal of this library is to be simple to use, there are only a few functions. Overall usage is described in this section, while the API is gone over in detail in the API Section[9] .

2.1 Test Output Location

There are two different locations that this Unit Testing Library can display its output. The output consists of the results of all the unit tests that have run.

The output is configured by the existence of an instance of the object named `oUnitTests` in the room that is executing the calls to the unit testing functions. If the instance exists, the output will be overlaid on the current room view. The output will be color coded so that green signifies a passed test while red signifies a failed test. If the instance doesn't exist, the output will be displayed in the **Output** console. In this case the output is not color coded.

When the library is first installed, there will be a room called `rUnitTests` that will contain an instance of the `oUnitTests` object. The room is configured to execute the `ac_unit_tests` script on creation. Unit tests can go in this script and they will be executed when the room is shown. However, this setup isn't required. The unit tests can go anywhere. Neither the room or the object are required and the test output will still display in the console.



API

3 API

API Documentation Conventions

The API reference in this manual follows these conventions.

Attributes

Some parameters can have attributes. They are:

- **optional:** This parameter is not needed. If the parameter has a default value, it will be described in the remarks section.
- **any type:** This parameter can be any data type. This usually means that the function doesn't need to know any details of the parameter or that the parameter will be used in conjunction with another parameter. Requirements for the parameter will be discussed in the remarks section.

Data Types

The data types used in this reference match those of GML. They are:

- string
- real
- array
- boolean
- pointer
- enum
- undefined

3.1 ac_assert_true

Description

This function checks the `condition` parameter, and if it evaluates to true, the unit test info will display as **passed** and the function will return `true`. If `condition` evaluates to anything other than true, the test will display as **failed** and the function will return `false`.

Parameters

Attributes	Type	Name	Description
	bool	condition	the condition to test
optional	string	description	the description of the test; useful when reading test output

Return Value

Type	Description
bool	`true` if `condition` evaluates to true; otherwise `false`

Remarks

The location of the output of this function depends on whether the `oUnitTests` object exists in the current room. See the Test Output Location[7] section for more details.

Examples

```
var test = ac_assert_true(true); // test equals true
```

3.2 ac_assert_false

Description

This function checks the `condition` parameter, and if it evaluates to false, the unit test info will display as **passed** and the function will return `true`. If `condition` evaluates to anything other than false, the test will display as **failed** and the function will return `false`.

Parameters

Attributes	Type	Name	Description
	bool	condition	the condition to test
optional	string	description	the description of the test; useful when reading test output

Return Value

Type	Description
bool	`true` if `condition` evaluates to false; otherwise `false`

Remarks

The location of the output of this function depends on whether the `oUnitTests` object exists in the current room. See the Test Output Location[7] for more details.

Examples

```
var test = ac_assert_false(true); // test equals false
```

3.3 ac_assert_equal

Description

This function checks the parameter `actual` and compares it to the parameter `expected`. If the parameters are equal, the unit test info will display as **passed** and the function will return `true`. If the parameters `expected` and `actual` are different, the test will display as **failed** and the function will return `false`.

Parameters

Attributes	Type	Name	Description
any type		expected	the value that is correct for the test to pass
any type		actual	the value that is actually submitted for the test
optional	string	description	the description of the test; useful when reading test output

Return Value

Type	Description
bool	`true` if `expected` equals `actual`; otherwise `false`

Remarks

If expected and actual are different data types, the test may not pass. This is by design.

The location of the output of this function depends on whether the `oUnitTests` object exists in the current room. See the Test Output Location[7] for more details.

Examples

```
var test1 = ac_assert_equal(1, 1); // test1 equals true
var test2 = ac_assert_equal(1, 2); // test2 equals false
```

- A -

- B -

- E -

- O -

- P -

- R -

- S -

- U -