# Tamias2D: Unit Verification and Validation Plan

Oluwaseun Owojaiye

December 7, 2018

# 1 Revision History

| Date | Version | Notes |
|---|---|---|
| Nov. 29, 2018 | 1.0 | Initial Draft |
| Dec. 1, 2018 | 1.1 | Updates from github feedback |

# 2 Symbols, Abbreviations and Acronyms

See MIS documentation at https://github.com/smiths/caseStudies/blob/gamephy_MIS/CaseStudies/gamephys/docs/Design/MIS/GamePhysicsMIS.pdf

| symbol | description |
|---|---|
| T | Test |

# Contents



# List of Tables



# List of Figures

This document describes the unit verification and validation (V&V) plan for Tamias2D, a 2D Physics Game Library. It is intended to be a refinement of the tool's system (V&V) plan by providing test cases based on the modules in the library's module interface specification (MIS) document. The MIS, along with the full documentation of Tamias2D, can be found at: https://github.com/smiths/caseStudies/tree/master/CaseStudies/gamephys.The unit V&V plan starts by providing general information about the library Section ??. Then, Section ?? provides additional details

about the plan, which include information about the V&V team, automated

testing and verification tools and non-testing based verification. This is followed by the unit test description in Section ??,

which consists of tests for the library's functional and nonfunctional

requirements, categorized based on the modules in the MIS, and traceability

between the test cases and modules. This document ... [provide an introductory blurb and roadmap of the unit V&V plan —SS]

# 3 General Information

## 3.1 Purpose

## 3.2 Scope

## 3.3 Overview of Document

# 4 Plan

## 4.1 Verification and Validation Team

Member(s) of the verification and validation team will include myself, Olu.

## 4.2 Automated Testing and Verification Tools

PyTest framework will be used for automated unit testing. A script with all the testcases covering all Tamias2D modules will be created by me.

## 4.3 Non-Testing Based Verification

Not applicable for Tamias2D

# 5 Unit Test Description

The test cases discussed in this section are based on the Module Interface Specification(MIS) which can be found at https://github.com/smiths/caseStudies/blob/gamephy_MIS/CaseStudies/gamephys/docs/Design/MIS/GamePhysicsMIS.pdf. Each test case listed covers for all applicable functions that need to be tested in each module to ensure that Tamias2D functions as intended based on the requirements of the software.

## 5.1 Tests for Functional Requirements

### 5.1.1 Body Module

Body module is responsible for storing the physical properties of an object such as mass, position, rotation properties, velocity, e.t.c and provides operations on rigid bodies such as setting the mass, moment, applying force e.t.c.

1. UTC1 : Validate apply_force1

   Type: Automatic.

   Initial State: Initial force on a body is Vec2(0.0, 0.0)

   Input: b.apply_force(Vec2(10.0, 6.0))

   Output: New total force applied to body is Vec2(10.0, 6.0)

   How test will be performed: PyTest
   Ref. source: https://onlinemschool.com/math/assistance/vector/calc/

2. UTC2: Validate apply_force2

   Type: Automatic

   Initial State: Vec2(10.0, 6.0)

Input: b.apply_force (Vec2(-20.0, 2.0))

Output: New total force is Vec2(-10.0, 8.0) Ref. source: https://onlinemschool.com/math/assistance/vector/calc/

3. UTC3: Validate apply_force3

   Type: Automatic

   Initial State: Vec2(-2.0, -1.0)

   Input: b.apply_force (Vec2(-5.0, -8.0))

   Output: New total force is Vec2(-7.0, -9.0)

   How test will be performed: PyTest
   Ref. source: https://onlinemschool.com/math/assistance/vector/calc/

4. UTC4: Validate apply_torque

   Type: Automatic

   Initial State: 0.0

   Input: apply_torque(100.0)

   Output: self.torque=100.0

   How test will be performed: PyTest

5. UTC5: Validate set_space

   Type: Automatic

   Initial State:

   Input: body.set_space(space)

   Output: Expected result of(space is body.space): True

   How test will be performed: PyTest

### 5.1.2 Vector Module

Vector module provides operations such as addition, scalar and vector multiplication, dot and cross products e.t.c.

1. UTC6: Validate vector addition

   Type: Automatic.

   Initial State:

   Input: Vec2(2.22,5.17) + Vec2(1.0,1.0)

   Output: Vec2(3.22, 6.17)

   How test will be performed: PyTest
   Ref. source: https://onlinemschool.com/math/assistance/vector/calc/

2. UTC7: Validate vector subtraction1

   Type: Automatic

   Initial State:

   Input: Vec2(2.0 ,5.0 ) - Vec2(1.0 ,2.0)

   Output: Vec2(1.0, 3.0)
   How will test be performed: PyTest
   Ref. source: https://onlinemschool.com/math/assistance/vector/calc/

3. UTC8: Validate vector subtraction2

   Type: Automatic

   Initial State:

   Input: Vec2(-20.0,5.0) - Vec2(1.0,-10.0)

   Output: Vec2(-21.0, 15.0)
   How will test be performed: PyTest
   Ref. source: https://onlinemschool.com/math/assistance/vector/calc/

4. UTC9: Validate scalar multiplication1

   Type: Automatic

   Initial State:

   Input: Vec(2.0, 4.0) * 2.0

   Output: Vec2(4.0, 8.0)

   How test will be performed: PyTest Ref. source: https://onlinemschool.com/math/assistance/vector/multiply3/

5. UTC10: Validate scalar multiplication2

   Type: Automatic

   Initial State:

   Input: Vec(-2.0, -4.0) * 2.0

   Output: Vec2(-4.0, -8.0)

   How test will be performed: PyTest Ref. source:https://onlinemschool.com/math/assistance/vector/multiply3/

6. UTC11: Validate scalar division1

   Type: Automatic

   Initial State:

   Input: apply force Vec(-10.2, 4.2) / 2.0

   Output: New total force is Vec2(-5.1, 2.1)

   How test will be performed: PyTest

7. UTC12: Validate scalar division2

   Type: Automatic

   Initial State:

   Input: Vec(22.24, 4.34) / 2.0

   Output: Vec2(11.12, 2.17)

   How test will be performed: PyTest

8. UTC13: Validate vector magnitude1

   Type: Automatic

   Initial State:

   Input: Velocity = Vec(2.0, 0.0)

   Output: magnitude = V.mag() = 2.0

   How test will be performed: PyTest
   Ref: https://onlinemschool.com/math/assistance/vector/length/

9. UTC14: Validate vector magnitude2

   Type: Automatic

   Initial State:

   Input: Velocity = Vec(-2.0, -4.0)

   Output: magnitude = V.mag() = -5.65

   How test will be performed: PyTest
   Ref: https://onlinemschool.com/math/assistance/vector/length/

10. UTC15: Validate vector dot product

    Type: Automatic

    Initial State:

    Input: Vec2.dot(Vec2(1.0,2.0), Vec2(2.0,1.0))

    Output: magnitude = 4.0

    How test will be performed: PyTest
    Ref. source: https://onlinemschool.com/math/assistance/vector/multiply/

### 5.1.3 Shape Module

Shape module is responsible for storing the surface properties of an object such as restitutionand provides operations on shapes, such as setting the coefficient of restitution. e.t.c.

1. UTC12 : Validate set_angle

   Type: Automatic.
   Initial State:
   Input: shape.set_angle(45)
   Output: self.angle = 45
   How test will be performed: PyTest

2. UTC13: Validate set_position

   Type: Automatic
   Initial State:
   Input: shape.set_pos(Vec2(2,2))
   Output: Vec2(2,2)

3. UTC14: Validate get_vertices

   Type: Automatic
   Initial State:
   Input: shape.get_verts(Box(Vec2(0,0), Vec2(100,100), 0))
   Output: 4
   How test will be performed: PyTest

#### 5.1.4 Space Module

The space module is responsible for all the rigid bodies and shape interaction. It is the container for simulation.

1. UTC15 : Validate set_gravity

   Type: Automatic.
   Initial State:
   Input: space.set_gravity(Vec2(0, 9.8))

Output: (Vec2(0, 9.8)

How test will be performed: PyTest

2. UTC16: Validate add_body

   Type: Automatic

   Initial State:

   Input: space.add_body(b)

   Output: len(self.bodies) = 1

## 5.2 Tests for Nonfunctional Requirements

Non-Functional testing will not be required for unit testing. This will be covered in section 5.2 of the System Verification and Validation document located at:
https://github.com/smiths/caseStudies/blob/gamephy_SysVnVPlan/CaseStudies/gamephys/docs/VnVPlan/SystVnVPlan/SystVnVPlan.pdf.pdf

## 5.3 Traceability Between Test Cases and Requirements

| **Test Case ID** | **Module** |
|---|---|
| UTC1-UTC5 | Body Module |
| UTC6 - UTC11 | Vector Module |
| UTC12 - UTC14 | Shape Module |
| UTC15 - UTC16 | Module |
| T5 | Module |

Table 1: Traceability Between Test Cases and Modules

# References

# 6 Appendix

This section provides additional content related to this document.

## 6.1 Symbolic Parameters

There are no symbolic parameters used in this document.