TamiasMini2D: Unit Verification and Validation Test Report

Oluwaseun Owojaiye

December 16, 2018

1 Revision History

Date	Version	Notes
Dec. 2, 2018	1.0	Initial Draft

2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test

For symbols, abbreviations and acronymns used in this document, please refer to SRS located below: https://github.com/smiths/caseStudies/blob/master/CaseStudies/gamephys/docs/SRS/GamePhysicsSRS.pdf

Contents

List of Tables

List of Figures

This document details the unit test activities and results for Tamias2D. For reference, the Unit test cases can be found in the Unit Verification and Validation Plan in the link below:

https://github.com/smiths/caseStudies/blob/gamephy_UnitVnV/CaseStudies/gamephys/docs/VnVPlan/UnitVnVPlan/UnitVnVPlan.pdf. Pytest framework was used for unit testing. All unit testcases are traceable to at least one module of the software, the mapping is shown in section 10 of this document.

3 Functional Requirements Evaluation

The unit test cases listed in the Unit VnV Plan document at https://github.com/smiths/caseStudies/blob/gamephy_UnitVnV/CaseStudies/gamephys/docs/VnVPlan/UnitVnVPlan/UnitVnVPlan.pdf were all executed. There are about 35 test cases that validate the functional requirements of Tamias2D. More test can be executed in a future phase of this project however the main unit test cases to check the functionality of the software were executed successfully. All tests were automated using Pytest framework. Each module has its set of unit test cases that is related to one or more functional requirement.

The testcases with the prefix test_vec2 prefixes represents testcases for vec2d module.This module is used for 2D coordinates and vector operations in Tamias2DTamias2D. All modules of the software use vec2d. The test_vec2d_add, test_vec2d_sub, test_vec2d_mul, test_vec2_mag, test_vec2_dot, test_vec2_scalarmul, test_vec2_scalardiv are used to compute vector addition, subtraction, multiplication, magnitude, dot product, scalar multiplication, and scalar division respectively. Functional Requirements 1-8 in SRS document located at https://github.com/smiths/caseStudies/tree/gamephy_SRSRevisions/CaseStudies/gamephys/docs/SRS all require at least one of the vec2d unit test cases e.g R1 states that "the software shall create a space for all rigid bodies". For a rigid body to exist in space, it needs a reference to space and the position which is the coordinate is represented as a vector i.e vec2d(x, y)in space. Unit test cases with prefix test_body represents tests in the body module. This ensures that rigid bodies can be created, initialized as well as receive inputs to be used to calculate the position and velocity as per R2. When force(s) are applied on a rigid body, the velocity, position, orientation history is computed, this relates to R5 and R6 which is also related to body module but uses vec2d, collision, shape and space mod-

ules. We also test for update_body, this is the function that calculates and updates body parameters. The collision handler which is an external module satisfies requirement R3, where the coefficient of restitution is applied to colliding bodies to determine the collision response and the impulse of collision. The requirement also uses all the modules except for body module, hence the unit test cases in the related module are important. The external collision solver module determines if rigid bodies have collided and determines their position, velocities and orientation over a period of time, the test cases with prefix test_CollisionHandler is used to check for collision as per the requirement R7. Some components of the requirements are implicitly tested in multiple modules due to the fact that the modules are interdependent.

4 Nonfunctional Requirements Evaluation

Nonfunctional unit tests were not performed for Tamias2D, nonfunctional requirement requirement evaluation will be reported in the System Verification and Validation report related to the System Verification and Validation Plan at : https://github.com/smiths/caseStudies/blob/gamephy_SysVnVPlan/CaseStudies/gamephys/docs/VnVPlan/SystVnVPlan/SystVnVPlan.pdf

5 Comparison to Existing Implementation

A comparison assessment will be performed during the System Verification and Validation test. Tamias2D will be compared with Pymunk2D for correctness test.

6 Unit Testing

During testing, when any of the test cases failed, after the whole test cycle is completed, an assertion error is displayed for failed test case. The output of the test run is displayed in the console showing how many testcases were executed, the number of passes and the number of fails. It also displays how much time it took to run all test cases. PyCharm IDE was used to run the test. To run the test, the user needs to run test_physics.py file in URL and then go to the terminal below the PyCharm Console and type python -m

pytest to run the unit test cases. When the cycle is completed the result is displayed.see the Appendix section below for the list of test run in pytest and the output results.

7 Changes Due to Testing

All of the tests cases passed hence no changes made due to testing. The only addition was a pytest.ini file was added to the project as I noticed that when I run the test file containing the unit test cases, it generates an error for some of the tests because it was calling some of the functions from the Collision module instead of the body module, so I added the pytest.ini textfile and added only the names of the functions I need to test, so if a function prefix is in the file, pytest will only call those functions from the software.

8 Automated Testing

All testing was done using pytest framework. Each testcase with output value is added into test_physics.py and each time the test is run, the terminal displays the result of the test, indication what test passed or failed with the AssertionError output and the testcaseID. The unit test case file can be found at URL

9 Trace to Requirements

Please refer to https://github.com/smiths/caseStudies/tree/gamephy_UnitVnV/CaseStudies/gamephys/docs/VnVPlan/UnitVnVPlan

10 Trace to Modules

https://github.com/smiths/caseStudies/tree/gamephy_UnitVnV/CaseStudies/gamephys/docs/VnVPlan/UnitVnVPlan

11 Code Coverage Metrics

There are no code coverage metrics for this software.

References