TamiasMini2D: System Verification and Validation Plan

Oluwaseun Owojaiye

November 8, 2018

1 Revision History

Date	Version	Notes
2018-11-01	1.1	Updates based on document review and issue tracker
2018-10-15	1.0	Initial draft

2 Symbols, Abbreviations and Acronyms

2.1 Table of Symbols

Symbol	Unit	Description
$\mathbf{a}$	$\mathrm{m\,s^{-2}}$	Acceleration
α	$\mathrm{rad\,s^{-2}}$	Angular acceleration
C_R	unitless	Coefficient of restitution
$\mathbf{F}$	N	Force
g	$\mathrm{m\,s^{-2}}$	Gravitational acceleration (9.81 $\mathrm{m\,s^{-2}}$)
G	$\mathrm{m^3\,kg^{-1}\,s^{-2}}$	Gravitational constant (6.673×10^{-11} $\mathrm{m^3\,kg^{-1}\,s^{-2}}$)
$\mathbf{I}$	$\mathrm{kg\,m^2}$	Moment of inertia
$\hat{\mathbf{i}}$	m	Horizontal unit vector
$\hat{\mathbf{j}}$	m	Vertical unit vector
j	N s	Impulse (scalar)
$\mathbf{J}$	N s	Impulse (vector)
L	m	Length
m	kg	Mass
n	unitless	Number of particles in a rigid body
$\mathbf{n}$	m	Collision normal vector
$\boldsymbol{\omega}$	$\mathrm{rad\,s^{-1}}$	Angular velocity
$\mathbf{p}$	m	Position
$\boldsymbol{\phi}$	rad	Orientation
r	m	Distance
$\mathbf{r}$	m	Displacement
t	s	Time
τ	N m	Torque
$\boldsymbol{\theta}$	rad	Angular displacement
$\mathbf{v}$	$\mathrm{m\,s^{-1}}$	Velocity

2.2 Abbreviations and Acronyms

Symbol	Description
IM	Instance Model
R	Requirement
T	Test
TBD	To be determined
2D	Two-dimensional
SRS	System Requirement Specification

Contents

List of Tables

List of Figures

This document provides a high-level verification and validation plan for TamiasMini2D - a 2D rigid body physics library. This document is based on the System Requirement Specification(SRS) document located in the following project repository link: https://github.com/smiths/caseStudies/tree/master/CaseStudies/gamephys. It discusses the verification and validation requirements for TamiasMini2D, and describes the test strategy and methods that will be used to evaluate the software. The verification and validation of the software utilizes review, analysis, and testing method to determine whether a software product complies with the specifed requirements. These requirements include both functional and non-functional.

[The text is better for version control, and for reading in other editors, if you use a hard-wrap at 80 characters —SS]

3 General Information

3.1 Summary

The software being tested is TamiasMini2D. It is a 2D rigid body physics library designed to simulate the interaction between rigid bodies. Since physics libraries are an important part of video game development, game developers will be able to make use of this library in their products.

3.2 Objectives

The purpose of verification and validation activities are to find bugs and defects in the TamiasMini2D physics library software and also to determine if it has met all the required functionality. It is also to verify that software meets the required standard and that the end product conforms with the software requirements based on the SRS. The objectives of System VnV activities for TamiasMini2D are to:

- Build confidence in software correctness and performance.
- Verify the maintainability of the software, based on the product's ability to be easily enhanced, modified and reused.
- Verify and demonstrate the ease of use and learning of the software.

3.3 References

[You should introduce the references, not just include a link. —SS]

1. `https://github.com/smiths/caseStudies/blob/GamePhy_Olu/CaseStudies/gamephys/docs/SRS/GamePhysicsSRS.pdf`

4 Plan

4.1 Verification and Validation Team

The verification and validation team consists of a one member team: Olu Owojaiye

4.2 SRS Verification Plan

The SRS for the project will be reviewed by Dr. Smith [LaTeX has a rule that it inserts two spaces at the end of a sentence. It detects a sentence as a period followed by a capital letter. This comes up, for instance, with Dr. Smith. Since the period after Dr. isn't actually the end of a sentence, you need to tell LaTeX to insert one space. You do this either by Dr. Smith (if you don't mind a line-break between Dr. and Smith), or Dr. Spencer Smith (to force LaTeX to not insert a line break). —SS] and coursemates and feedback will be provided. Some SRS feedback for this project have been provided and addressed using github issue tracker. Also once the software has been implemented, the SRS will be reviewed to ensure that software has met all the specified requirements and more feedback will be provided via github issue tracker.

[You can be specific about which classmates are going to review your documents; the specific assignments are in Repos.xlsx. —SS]

4.3 Design Verification Plan

To ensure that the Design Specification has been properly specified and meets software requirements, Dr. Smith and my coursemate(s) will be verifying the software design. The Module Guide and Module Interface Specification will contain information about the software design. Feedback is expected to be provided by reviewers via github issue tracker. [You should have links to MG and MIS. —SS]

4.4 Implementation Verification Plan

The implementation of TamiasMini2D will involve inspection of the software to ensure that all the required features have been implemented successfully and are functional. Once the development activities are completed, Dr. Smith and some of CAS741 coursemates will perform the implementation verification activities. The software will be installed by the testers and system test cases specified in Section 5 will be run. Reviewers are expected to verify both functional and non-functional requirements specified below. Exploratory testing can also be performed by testers. Any implementation verification issues will be reported and tracked via github issue tracker and these issues will be resolved in order of severity by myself. After the issues raised have been fixed, they will be sent back to the reviewer(s) for re-verification.

4.5 Software Validation Plan

There is currently no software validation plan for TamiasMini2D.

[You should the reason why there is no software validation plan. —SS]

5 System Test Description

5.1 Tests for Functional Requirements

5.1.1 Translational Motion Testing

1. TC1: Static rigid body velocity-position calculation

   Description: Calculate the position and velocity of a 2D rigid body that is static after t secs. Gravity is not applied. The object does not move, hence velocity and position does not change based on IM1. Varying positive input parameters can be used in TC1 for each of the parameters

   Control: Automatic

   Initial State: NA

Input: $\mathbf{p_i(t_0)} = (0, 0)$;This is the initial position of body - (x,y) coordinate position

$\mathbf{v_i(t_0)} = 0$
$\mathbf{F_i(t_0)} = 0$
$\mathbf{m_i} = 10$
$\mathbf{g} = 0$ (acceleration due to gravity does not apply on a static body)

Output: $\mathbf{p_i(t)} = (0, 0)$; $\mathbf{v_i(t)} = (0, 0)$; after t secs where t= 3

How test will be performed: Unit testing with PyUnit

2. TC2: Dynamic 2D rigid body falling from a height velocity-position calculation

   Description: Calculation of the new position and velocity of a dynamic body. Force of gravity is in effect and we apply a horizontal force F. The new position and velocity is calculated. This test can be applied to a set of rigid bodies falling from a height at the same different time t.

   Control: Automatic

   Initial State: NA

   Input: $\mathbf{p_i(t_0)} = (20, 20)$ this is the (x,y) coordinate position
   $\mathbf{v_i(t_0)} = 0$
   $\mathbf{F_i(t_0)} = 10$
   $\mathbf{m_i} = 100$
   $\mathbf{g} = 9.8$ (acceleration due to gravity)

   Output: $\mathbf{p_i(t)} = (20.9, 20.882)$; $\mathbf{v_i(t)} = (0.3, 0.294)$; after t secs, where t= 3

   How test will be performed: Unit testing with PyUnit

   [Where did the answers come from? The reader won't be able to verify what you are saying. You are verifying for one point in time. You can accomplish more by verifying the full history of the change in position. You should provide the closed form solutions for position change under constant acceleration. You then get can run the simulation until the position is zero and verify that the positions are correct. To get one number for your test you can use the Euclidean (or other) norm of your vector and then divide by the norm of the expected result. —SS]

3. TC3: Projectile motion of dynamic 2D rigid velocity-position calculation

   Description:"Projectile motion is a form of motion experienced by an object or particle (a projectile) that is thrown near the Earth's surface and moves along a curved path under the action of gravity only". The rigid object is falling from a height specified in input. In horizontal direction, velocity is constant.

   Control: Automatic

   Initial State: NA

   Input: $\mathbf{p_i}(\mathbf{t_0})$ = (0, 125) this is the (x,y) coordinate position
   $\mathbf{v_i}(\mathbf{t_0})$ = (10,0)
   $\mathbf{F_i}(\mathbf{t_0}) = 10$
   $\mathbf{m_i} = 10$
   $\mathbf{g} = 9.8$ (acceleration due to gravity)
   **Need to doublecheck this calculation
   Output: $\mathbf{p_i}(\mathbf{t})$ = (50, 0); $\mathbf{v_i}(\mathbf{t})$ = (10, 50); after t secs, where t= 5

   How test will be performed: Unit testing with PyUnit

[Same comment as for previous test. I believe all of the equations you will need for your closed form solutions are at: `https://en.wikipedia.org/wiki/Projectile_motion` —SS]

[You are using x and y in the conventional orientation, but you should also have tests where your projectile is thrown in the opposite direction and when your projectile has a non-zero velocity in the y direction. You want to make sure that there isn't an error in any of the coordinate directions. If everything is down and to the left you won't notice errors with motion up or down. —SS]

5.1.2 Rotation of 2D Rigid Body Simulation

This test is to simulate the rotation of a 2D rigid body about its axis.

1. TC4: 2D Rigid body rotation about its axis

   Description: In rotational motion of 2D rigid bodies, Torque τ is the force which produces rotation. It has magnitude and direction.This test can also be used for multiple set of rigid bodies.(IM2)

   Control: Automatic

   Initial State: NA

   Input: $\mathbf{m_i} = 100$
   $\mathbf{g} = 0$
   $\phi_i(\mathbf{t_0}) = 50$
   $\omega_i(\mathbf{t_0}) = 0.3$
   $\tau = 1000$
   $\mathbf{I}(\mathbf{i}) = 10000$

   Output: $\phi(\mathbf{t}) = 50.9$; $\omega(\mathbf{t}) = 0.3$; after t secs where t= 3

   How test will be performed: Unit testing with PyUnit

[Nice to see rotation tests not being forgotten. As before though, I would like to know how you come up with your output answers. —SS]

5.1.3 Rotation of 2D Rigid Body Simulation

This test is to simulate the collision of 2D rigid bodies.

1. TC5: Dynamic rigid body collision with static body test

   Description: This is to test a set of rigid bodies that collide. This test case will test for collision of a dynamic object falling from a height with a static object. At collision the static object does not move, the dynamic object's velocity, position, angular velocity, orientation is calculated.Momentum is conserved.

   Control: Automatic

   Initial State: NA

Input: $\mathbf{m_k}$ = TBD (to be determined)
$\mathbf{p_k(t_0)}$ = TBD
$\mathbf{v_k(t_0)}$ = TBD
$\phi_k(\mathbf{t_0})$ = TBD
$\omega_k(\mathbf{t_0})$ = TBD
$\mathbf{C_R}$ = TBD

Output: $\mathbf{v_k(t)}$ = TBD
$\mathbf{p_k(t)}$ = TBD
$\phi_k(\mathbf{t})$ = TBD
$\omega_k(\mathbf{t})$ = TBD
after t secs(t, TBD)

How test will be performed: Unit testing with PyUnit

2. TC6: Dynamic rigid Body collision test(set of bodies)

Description: This is to test a set of rigid bodies that collide. This test case will test for collision of a set of dynamic object falling from a height with a static object. At collision bodies' velocity, position, angular velocity, orientation is calculated. Momentum is conserved. The input and output set generated is determined by the number of bodies added in space.Multiple objects will be simulated to fall at different times fro a height so we can simulate collision.

Control: Automatic

Initial State: NA

Input: $\mathbf{m_k}$ = TBD (to be determined)
$\mathbf{p_k(t_0)}$ = TBD
$\mathbf{v_k(t_0)}$ = TBD
$\phi_k(\mathbf{t_0})$ = TBD
$\omega_k(\mathbf{t_0})$ = TBD
$\mathbf{C_R}$ = TBD

Output: $\mathbf{v_k(t)}$ = TBD
$\mathbf{p_k(t)}$ = TBD
$\phi_k(\mathbf{t})$ = TBD

$\omega_k(\mathbf{t})$ = TBD
after t secs(t, TBD)

How test will be performed: Unit testing with PyUnit

[The TBDs should be filled in. I get the impression that some of the physics is giving you trouble. The following resource looks pretty good `https://www.myphysicslab.com/engine2D/collision-en.html`. You could use this calculator `https://www.omnicalculator.com/physics/conservation-of-momentum` with your coefficient or restitution set to 1.0. You just need the objects to not rotate after their collision. Two spheres colliding should be fine. You can also play around with problems where one mass is so large that that object will essentially be stationary. I can also lend you a physics textbook, if that would be helpful. —SS]

5.2 Tests for Nonfunctional Requirements

5.2.1 Usability Test

Usability test

1. TC7

   Type: Usability test

   Initial State:

   Input/Condition:

   Output/Result:

   How test will be performed: Users/reviewers of TamiasMini2D will be asked to install the library and use it. They will be asked to complete the survey in the Appendix section for Usability.

[Nice to see a usability test. It is a bit simplistic, but that is fine for right now. —SS]

5.2.2 Correctness/Performance

Correctness/Performance

1. TC8

   Type: Dynamic

   Initial State:

   Input:

   Output:

   How test will be performed: Correctness/performance will be measured by comparing output results with ODEs related to each requirement/function.

[This isn't really complete. For correctness, you can probably refer to your functional tests from the previous section. —SS]

5.2.3 Reusability

Reusability test

1. TC9

   Type: Dynamic

   Initial State:

   Input:

   Output:

   How test will be performed: Users/reviewers will be asked to see if they can extend the library and use for other purposes. ...

[A survey is an interesting way to measure this. —SS]

5.2.4 Understandability/Maintainability

Understandability/Maintainability test

1. TC9

   Type: Dynamic

   Initial State:

Table 1: Requirements Traceability Matrix

Testcase Number	Instance Models	CA Requirements
TC1	IM1	R1, R2, R4, R5
TC2	IM1	R1, R2, R4, R5
TC3	IM1	R1, R2, R4, R5
TC4	IM2	R1, R2, R4, R6
TC5	IM3	R1, R3, R4, R7, R8
TC6	IM3	R1, R3, R4, R7, R8
TC7		NFR3
TC8		NFR1, NFR2
TC9		NFR5
TC10		NFR4, NFR6

Input:

Output:

How test will be performed: Users/reviewers will be asked to see check some see if they are able to find the space module and update the body parameters are desired. Users will be asked on the scale of 1 to 5 how easy it was to find the code, understand it and make changes. ...

5.3 Traceability Between Test Cases and Requirements

The purpose of the information in Table 1 below is to provide a mapping between the test cases and the requirements in the SRS for easy reference and verification.

6 Static Verification Techniques

Code review and inspection will be used as the method for implementation verification. [You can remove this section, since the details can be covered in Section 4.4. I've realized that I should remove this section from the template. —SS]

References

- http://www.physicstutorials.org/home/mechanics/1d-kinematics/projectile-motion?start=1
- https://en.wikipedia.org/wiki/Projectile-motion
- R. A. BROUCKE. "Equations of motion of a rotating rigid body", Journal of Guidance, Control, and Dynamics, Vol. 13, No. 6 (1990), pp. 1150-1152.
- https://github.com/smiths/caseStudies/blob/master/CaseStudies/gamephys/docs/SRS/Gar

7 Appendix

This is where you can place additional information.

7.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

7.2 Usability Survey Questions?

1. On the scale of 1 - 5, 1 being very difficult and 5 being very easy, How easy was it to install the program using the installation guide?

   Comment on what can be improved:

2. On a scale of 1 - 5, how easy were you able to update the parameters in a space? e.g change the velocity of a body

3. Did the program return the expected output based on the testcase and input values?

   If no, please add comments explaining issues encountered: