

# Test Plan for Breaking Effect (BE)

Marshall Xiaoye Ma

November 15, 2017

# 1 Revision History

Date	Version	Notes
2017-10-21	1.0	New Document

## 2 Symbols, Abbreviations and Acronyms

symbol	unit	description
$S$	unit of length	displacement
$t$	$s$	the time in seconds since the start of the game
$v_0$	$m/s$	initial speed
$a$	$m/s^2$	acceleration
$g$	$m/s^2$	gravity acceleration
$\Delta E_k$	J	variation of kinetic energy
$\Delta E_p$	J	variation of potential energy
$W_f$	$J$	work done by kinetic friction
$x_n$	$m$	x coordinates of gravity center of piece $n$ , $n \in N$
$y_n$	$m$	y coordinates of gravity center of piece $n$
$z_n$	$m$	z coordinates of gravity center of piece $n$
$\theta_1$	degree	angle between initial speed and horizontal
$\theta_2$	degree	angle between x axiom and projection on horizontal of initial speed
$S_x$	m	displacement on direction of x axiom
$S_y$	m	displacement on direction of y axiom
$S_z$	m	displacement on direction of z axiom
$\mu_k$		coefficient of friction

<b>symbol</b>	<b>description</b>
A	Assumption
DD	Data Definition
GD	General Definition
GS	Goal Statement
IM	Instance Model
LC	Likely Change
PS	Physical System Description
R	Requirement
SRS	Software Requirements Specification
BE	Breaking Effect
T	Theoretical Model

# Contents

<b>1</b>	<b>Revision History</b>	<b>i</b>
<b>2</b>	<b>Symbols, Abbreviations and Acronyms</b>	<b>ii</b>
<b>3</b>	<b>General Information</b>	<b>1</b>
3.1	Purpose . . . . .	1
3.2	Scope . . . . .	1
3.3	Overview of Document . . . . .	1
<b>4</b>	<b>Plan</b>	<b>2</b>
4.1	Software Description . . . . .	2
4.2	Test Team . . . . .	2
4.3	Automated Testing Approach . . . . .	2
4.4	Verification Tools . . . . .	2
4.5	Non-Testing Based Verification . . . . .	3
<b>5</b>	<b>System Test Description</b>	<b>3</b>
5.1	Tests for Functional Requirements . . . . .	4
5.1.1	Getting input from user . . . . .	4
5.1.2	Obtaining gravity center of each piece . . . . .	8
5.1.3	Initial speed calculation of each piece . . . . .	9
5.1.4	Calculation of angle between initial speed and horizontal( $\theta_1$ ) as well as the angle between x axiom and projection on horizontal of initial speed( $\theta_2$ ) . . . . .	9
5.1.5	Calculation of displacement for each piece in the air . . . . .	11
5.1.6	Calculation of displacement for each piece on the ground . . . . .	12
5.1.7	Comparison with professional plug-in . . . . .	12
5.2	Tests for Nonfunctional Requirements . . . . .	13
5.2.1	Performance test . . . . .	13
5.2.2	Comparison with professional plug-in . . . . .	13
5.3	Traceability Between Test Cases and Requirements . . . . .	13
<b>6</b>	<b>Unit Testing Plan</b>	<b>14</b>
6.1	Unit test for initial speed calculation . . . . .	15
6.2	Unit test for gravity center calculation . . . . .	15
6.3	Unit test for angles calculation . . . . .	15
6.4	Unit test for displacement calculation . . . . .	16

<b>7</b>	<b>Appendix</b>	<b>18</b>
7.1	Symbolic Parameters . . . . .	18
7.2	Usability Survey Questions? . . . . .	18

## List of Tables

1	Traceability Matrix Showing the Connections Between Items of Different Sections . . . . .	14
---	--	----

## List of Figures

[If you don't have a list of figures, you can remove this heading. —SS]

## 3 General Information

### 3.1 Purpose

The purpose for this document is to build test plan for project Breaking Effect. Test cases in the document are designed based on SRS of the project and aim to cover both functional and non-functional requirements described in SRS.

[You don't need to end paragraphs with an explicit character. This practice has the unfortunate consequence of sometimes leaving an extra blank line in the generated document. —SS]

This document is used as a guide that need to be followed exactly in test stage before release of this project.

### 3.2 Scope

This test plan covers verification and validation for the project Breaking Effect to make sure the program is implemented as requirement specification, including automated testing, system test and unit test. The project relies on Unity3D as platform and uses function provided by Unity3D for object cutting. So test for object cutting is not included in this test plan. This test plan is designed based on SRS so that it doesn't cover test cases for requirements not in SRS.

[An explicit web-link to your GitHub repo would be nice. —SS]

### 3.3 Overview of Document

This test plan firstly describes environment and platform for Breaking Effect. Purpose and scope of outlined the document generally in previous sections. Detailed test plan and test cases are covered in following sections including test team, automated testing approach, verification tools and test cases for both functional requirements and non-functional requirements.

## 4 Plan

### 4.1 Software Description

Breaking effect presents how the pieces of an object move after it separates into parts with suddenness or violence.

This project implements running time breaking effect in codes for 3-D models in unity3D without help from any similar plug-in. Including different shapes 3-D objects breaking based on physics and pieces interacting with the momentum provided by the breaking force. The breaking effect program simulates 3-D objects destruction process in vision by implementing scientific computing functions.

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

This project concentrates on calculation while HCI or GUI are not important parts. Applied force is decided in codes in advance as input and trace of motion is the output after calculation.

### 4.2 Test Team

The team that is responsible for all tests is Xiaoye Ma.

### 4.3 Automated Testing Approach

Most testing work for Breaking Effect does not rely on automated testing. Because the final output is visualization and experience from users. Verification of inputs will be done automatically through help from Unit Test Generator provided by IDE Visual Studio 2017. Test cases are covered in following sections. Verification of correctness and output from intermediate steps will be tested manually.

Automated testing also helps check correctness of C# codes including bugs and syntax errors detecting.

### 4.4 Verification Tools

The project is implemented by C# programming language that is supported by platform Unity3D. The program is written through IDE Visual Studio 2017 which supports following tools:



- Unit Test Generator

Unit Test Generator will be used to decrease workload involved in creating new unit tests. It helps the routine test creation tasks. It also provides the ability to generate and configure test project and test class.

- Automatic code checking

Visual Studio automatically help check code error when developer is programming to pick out errors including syntax error, grammar error.

## 4.5 Non-Testing Based Verification

Code walkthrough will be done by developer and a peer reviewer to identify any defects. They both review codes and do logic analysis to see if the program satisfies all requirements in SRS. The walkthrough should also find and fix possible bugs and peer reviewer is expected to provide any comment to help developer improve the program.

[Great to see a code walkthrough! However, could you please give some more detail on how the walkthrough will be done. How many participants? How will they be recruited? What documentation will they be given? etc. What specific guidelines will there be for the meeting and the process? If you google "code walkthrough" you will find guidance on how to prepare for and structure the meeting. How will the results of the code walkthrough be summarized? —SS]

## 5 System Test Description

Section 5.1.1 focuses on verifying correctness of users inputs that covers R1 and R3. Testing cases in this section ensure Breaking Effect can handle both valid inputs and invalid inputs. Corresponding error messages for different situations of invalid inputs including incomplete inputs, inputs in incorrect data types and inputs do not satisfy data constraints.

Sections from 5.1.2 focus on verifying outputs from intermediate steps and make sure all modules in program work correctly that cover other functional requirements including R2, R4, R5, R6, R7.

[Very nice to have explicit cross-references between documents. To keep the information up to date, consider using make to build your documentation. —SS]

## 5.1 Tests for Functional Requirements

### 5.1.1 Getting input from user

This test suite is designed to ensure the program can handle both valid and invalid inputs from user.

Inputs provided by user include target object, explosion level (also known as initial momentum after explosion) and coefficient of friction on the ground.

#### Correct input

1. testCorrectInput

Type: Functional, Dynamic, Automated

Initial State: New Session.

Input:  $E = 5$ ,  $(X, Y, Z) = (0, 0, 0)$ ,  $\mu_k = 0.05$

Output: Receive inputs successfully and display all inputs in console.

How test will be performed: Automated unit test

#### Missing necessary input

1. testNoObject

Type: Functional, Dynamic, Automated

Initial State: New Session.

Input:  $E = 5$ ,  $\mu_k = 0.05$

Output: Error message - No object is found.

How test will be performed: Automated unit test

2. testNoMu

Type: Functional, Dynamic, Automated

Initial State: New Session.

Input:  $E = 5$ ,  $(X, Y, Z) = (0, 0, 0)$

Output: Error message - Please input coefficient of friction on the ground.

How test will be performed: Automated unit test

### 3. testNoMomentum

Type: Functional, Dynamic, Automated

Initial State: New Session.

Input:  $(X, Y, Z) = (0, 0, 0)$ ,  $\mu_k = 0.05$

Output: Error message - Please input explosion level.

How test will be performed: Automated unit test

## Non-numerical value

### 1. testNonNumMomentum

Type: Functional, Dynamic, Automated

Initial State: New Session.

Input:  $E = a$ ,  $(X, Y, Z) = (0, 0, 0)$ ,  $\mu_k = 0.05$

Output: Error message - Explosion level can only be a number from 1 to 10. [You have a test where  $E$  is not a number, but I don't see a test where  $E$  is a number, but not in the correct range. Okay, I found the test later in the document. These tests are so simple and repetitive, it would be better to summarize them in a table, where each row corresponds to a different test. —SS]

How test will be performed: Automated unit test

### 2. testNonNumCoordinate

Type: Functional, Dynamic, Automated

Initial State: New Session.

Input:  $E = 5$ ,  $(X, Y, Z) = (0, 0, a)$ ,  $\mu_k = 0.05$

Output: Error message - Coordinate can only be a number from -1000 to 1000.

How test will be performed: Automated unit test

### 3. testNonNumMu

Type: Functional, Dynamic, Automated

Initial State: New Session.

Input:  $E = 5$ ,  $(X, Y, Z) = (0, 0, 0)$ ,  $\mu_k = a$

Output: Error message - coefficient of friction can only be a number from 0 to 1.

How test will be performed: Automated unit test

## Incomplete inputs

### 1. testMissingAxiom1

Type: Functional, Dynamic, Automated

Initial State: New Session.

Input:  $E = 5$ ,  $(X, Y, Z) = (0, 0, )$ ,  $\mu_k = 0.05$

Output: Error message - Please complete input.

How test will be performed: Automated unit test

### 2. testMissingAxiom2

Type: Functional, Dynamic, Automated

Initial State: New Session.

Input:  $E = 5$ ,  $(X, Y, Z) = (0, , 0)$ ,  $\mu_k = 0.05$

Output: Error message - Please complete input.

How test will be performed: Automated unit test

3. testMissingAxiom3

Type: Functional, Dynamic, Automated

Initial State: New Session.

Input:  $E = 5$ ,  $(X, Y, Z) = (, 0, 0)$ ,  $\mu_k = 0.05$

Output: Error message - Please complete input.

How test will be performed: Automated unit test

**Inputs out of scope**

1. testWrongMomentum

Type: Functional, Dynamic, Automated

Initial State: New Session.

Input:  $E = -1$  or  $E = 11$ ,  $(X, Y, Z) = (0, 0, 0)$ ,  $\mu_k = 0.05$

Output: Error message - Explosion level can only be a number from 1 to 10.

How test will be performed: Automated unit test

2. testWrongXY1

Type: Functional, Dynamic, Automated

Initial State: New Session.

Input:  $E = 5$ ,  $(X, Y, Z) = (0, 99999, 0)$ ,  $\mu_k = 0.05$

Output: Error message - Coordinate X and Y can only be a number from -1000 to 1000.

How test will be performed: Automated unit test

3. testWrongXY2

Type: Functional, Dynamic, Automated

Initial State: New Session.

Input:  $E = 5$ ,  $(X, Y, Z) = (99999, 0, 0)$ ,  $\mu_k = 0.05$

Output: Error message - Coordinate X and Y can only be a number from -1000 to 1000.

How test will be performed: Automated unit test

#### 4. testWrongZ

Type: Functional, Dynamic, Automated

Initial State: New Session.

Input:  $E = 5$ ,  $(X, Y, Z) = (0, 0, 5)$ ,  $\mu_k = 0.05$

Output: Error message - Coordinate Z can only be 0.

How test will be performed: Automated unit test

#### 5. testWrongMu

Type: Functional, Dynamic, Automated

Initial State: New Session.

Input:  $E = 5$ ,  $(X, Y, Z) = (0, 0, 0)$ ,  $\mu_k = 99$

Output: Error message - coefficient of friction can only be a number from 0 to 1.

How test will be performed: Automated unit test

### 5.1.2 Obtaining gravity center of each piece

This section covers R4 in SRS that program needs to obtain gravity center of each piece correctly. Testing for edge cases and error handling will be covered in Section 6.

There is an assumption for this section that all tests in 5.1.1 are passed.

#### 1. testGravityCenter

Type: Functional, Dynamic, Manual

Initial State: New Session.

Input:  $E = 5$ ,  $(X, Y, Z) = (0, 0, 0)$ ,  $\mu_k = 0.05$

Output: A list of coordinates of all pieces in format  $(x_n, y_n, z_n)$  in the console.

How test will be performed: Manually input and check output by developer.

[You can be more specific here. What is the shape. What is the centre of gravity? Why isn't this test automated? —SS]

### 5.1.3 Initial speed calculation of each piece

Test case in this section covers R2 in SRS that program calculates initial speed for each piece. Testing for edge cases and error handling will be covered in Section 6.

There is an assumption for this section that all tests in 5.1.1 and 5.1.2 are passed.

#### 1. testInitialSpeed

Type: Functional, Dynamic, Manual

Initial State: New Session.

Input:  $E = [1, 10]$ ,  $(X, Y, Z) = (0, 0, 0)$ ,  $\mu_k = 0.05$

Output: initial speed  $v_0 = 10 * E$  [Why? How was this derived? The reader should be able to verify the test cases and there isn't enough information here to do the verification. —SS]

How test will be performed: Manually input and check output by developer.

### 5.1.4 Calculation of angle between initial speed and horizontal( $\theta_1$ ) as well as the angle between x axiom and projection on horizontal of initial speed( $\theta_2$ )

This section is designed to determine angle calculation module works correctly to calculate two angles, which covers R5 in SRS. Testing for edge cases and error handling will be covered in Section 6.

There is an assumption for this section that all tests in 5.1.1, 5.1.2 and 5.1.3 are passed.

### 1. testAngles1

Type: Functional, Dynamic, Manual

Initial State: New Session.

Input:  $E = 5$ ,  $(X, Y, Z) = (0, 0, 0)$ ,  $\mu_k = 0.05$ ,  $(x_n, y_n, z_n)$  and  $x_n, y_n, z_n$  can't be zero, which is the output list from [5.1.2](#)

Output: Pairs of  $\theta_1$  and  $\theta_2$ . Where  $\theta_1 = \arctan \frac{|z_n|}{\sqrt{x_n^2 + y_n^2}}$  and  $\theta_2 = \arctan \frac{y_n}{x_n}$

How test will be performed: Manually input and check output by developer. [Here we take  $(X, Y, Z) = (0, 0, 0)$  as input case for System test. Test cases for different values will be covered in unit test part [section 6](#) —Author]

[Rather than be general, you should give specific input values and specific output values. This comment applies elsewhere in this document as well. Again, it isn't clear why this has to be a manual test. —SS]

### 2. testAngles2

Type: Functional, Dynamic, Manual

Initial State: New Session.

Input:  $E = 5$ ,  $(X, Y, Z) = (0, 0, 0)$ ,  $\mu_k = 0.05$ ,  $(0, 0, z_n)$  and  $z_n$  can't be zero, which is output from [5.1.2](#)

Output:  $\theta_1 = 90^\circ$  and  $\theta_2 = \arctan \frac{y_n}{x_n}$

How test will be performed: Manually input and check output by developer.

### 3. testAngles3

Type: Functional, Dynamic, Manual

Initial State: New Session.

Input:  $E = 5$ ,  $(X, Y, Z) = (0, 0, 0)$ ,  $\mu_k = 0.05$ ,  $(x_n, y_n, 0)$  and  $x_n, y_n$  can't be zero, which is output from [5.1.2](#)

Output:  $\theta_1 = \arctan \frac{|z_n|}{\sqrt{x_n^2 + y_n^2}}$  and  $\theta_2 = 90^\circ$



How test will be performed: Manually input and check output by developer.

#### 4. testAngles4

Type: Functional, Dynamic, Manual

Initial State: New Session.

Input:  $E = 5$ ,  $(X, Y, Z) = (0, 0, 0)$ ,  $\mu_k = 0.05$ ,  $(x_n, y_n, z_n) = (0, 0, 0)$  which is output from 5.1.2

Output:  $\theta_1 = 90^\circ$  and  $\theta_2 = 90^\circ$

How test will be performed: Manually input and check output by developer.

### 5.1.5 Calculation of displacement for each piece in the air

This section covers R6 that displacement of each piece in the air should be calculated correctly. Testing for edge cases and error handling will be covered in Section 6.

There is an assumption for this section that all tests in 5.1.1, 5.1.2, 5.1.3 and 5.1.4 are passed.

#### 1. testMotionAir

Type: Functional, Dynamic, Manual

Initial State: New Session.

Input:  $E = 5$ ,  $(X, Y, Z) = (0, 0, 0)$ ,  $\mu_k = 0.05$ , time  $t$ , and  $\theta_1$ ,  $\theta_2$ ,  $v_0$  that are result from 5.1.4, 5.1.3

Output:  $S_x = v_0 \cdot \cos\theta_1 \cdot \cos\theta_2 \cdot t$ ,  $S_y = v_0 \cdot \cos\theta_1 \cdot \sin\theta_2 \cdot t$  and  $S_z = v_0 \cdot \sin\theta_1 \cdot t - \frac{1}{2}gt^2$ . Above output should be shown both in text in console and in vision on the screen.

How test will be performed: Manually input and check output by developer. [Here we take  $E = 5$  and  $(X, Y, Z) = (0, 0, 0)$  as input case for System test. Test cases for different values will be covered in unit test part section 6 —Author]

### 5.1.6 Calculation of displacement for each piece on the ground

This section tests the final functional requirement R7 is satisfied correctly that displacement of each piece on the ground should be calculated correctly. Testing for edge cases and error handling will be covered in Section 6.

There is an assumption for this section that all tests in 5.1.1, 5.1.2, 5.1.3, 5.1.4 and 5.1.5 passed.

#### 1. testMotionGround

Type: Functional, Dynamic, Manual

Initial State: New Session.

Input:  $E = 5$ ,  $(X, Y, Z) = (0, 0, 0)$ ,  $\mu_k = 0.05$ , time  $t$  and  $\theta_1$ ,  $\theta_2$ ,  $v_0$  that are result from 5.1.4, 5.1.3

Output:  $S_x = v_0 \cdot \cos\theta_1 \cdot \cos\theta_2 \cdot t - \frac{1}{2}at^2$  and  $S_y = v_0 \cdot \cos\theta_1 \cdot \sin\theta_2 \cdot t - \frac{1}{2}at^2$ . Above output should be shown both in text in console and in vision on the screen.

How test will be performed: Manually input and check output by developer. [Here we take  $E = 5$  and  $(X, Y, Z) = (0, 0, 0)$  as input case for System test. Test cases for different values will be covered in unit test part section 6 —Author]

### 5.1.7 Comparison with professional plug-in

This test case is designed to compare Breaking Effect with existing plug-in in Unity3D. Visualization is the most important point of the project so this section aims to test if Breaking Effect simulates the explosion vividly.

#### 1. testCompare

Type: Functional, Dynamic, Manual

Initial State: New session.

Input:  $E = 5$ ,  $(X, Y, Z) = (0, 0, 0)$ ,  $\mu_k = 0.05$

Output: Motion of each piece.

How test will be performed: Compare output with professional plug-ins.

## 5.2 Tests for Nonfunctional Requirements

### 5.2.1 Performance test

1. testPerformance

Type: Nonfunctional, Dynamic, Manual

Initial State: New session

Input/Condition:  $E = 5$ ,  $(X, Y, Z) = (0, 0, 0)$ ,  $\mu_k = 0.05$

Output/Result: Motion of each piece.

How test will be performed: Adjust amount of pieces to see the performance that if program can run smoothly in huge amount of pieces.

[Be more specific. What will the range of the number of pieces be? How will you judge smoothness? You might need to also summarize the target hardware. Can you summarize the total time of calculation and plot it? —SS]

### 5.2.2 Comparison with professional plug-in

This test case is designed to compare Breaking Effect with existing plug-in in Unity3D. Visualization is the most important point of the project so this section aims to test if Breaking Effect simulates the explosion vividly.

1. testCompare

Type: Functional, Dynamic, Manual

Initial State: New session.

Input:  $E = 5$ ,  $(X, Y, Z) = (0, 0, 0)$ ,  $\mu_k = 0.05$ . Performance of professional plug-in, example: Realistic Explosions Pack.

Output: Motion of each piece.

How test will be performed: Compare output with professional plug-ins.

[How will the comparison be assessed? quantified? —SS]

## 5.3 Traceability Between Test Cases and Requirements

	R1	R2	R3	R4	R5	R6	R7
testCorrectInput	X		X				
testNoObject	X						
testNoMu	X						
testNoMomentum	X						
testNonNumMomentum			X				
testNonNumCoordinate			X				
testNonNumMu			X				
testMissingAxiom			X				
testWrongMomentum			X				
testWrongCoordinate			X				
testWrongMu			X				
testGravityCenter				X			
testInitialSpeed		X					
testAngles					X		
testMotionAir						X	
testMotionGround							X

Table 1: Traceability Matrix Showing the Connections Between Items of Different Sections

## 6 Unit Testing Plan

Unit Test Generator provided by Visual Studio 2017 will be used to implement automated unit testing for this project.

For each function, a test class will be created through Unit Test Generator. Developer provides inputs, runs automated test to see if it is passed.

Inputs will be provided by different target objects, initial locations, initial momentum, coefficient of friction.

Unit tests for this project focuses on edge cases checking and error handling. Unit tests for input handling module are covered in 5.1.1.

Unit tests for other modules will be designed after modules designing are finished.

## 6.1 Unit test for initial speed calculation

### 1. UnitTestIS

Type: Functional, Dynamic, Automated

Initial State: New session.

Input: Edge cases for E. Example: E=1, E=10. Error cases for E. Example: E=nil, E=-1, E=99

Output: Initial speed  $v_0$  when edge cases. Error message when error cases.

How test will be performed: Automated unit testing.

## 6.2 Unit test for gravity center calculation

[It is unclear how this test differs from the previous centre of gravity test? —SS]

### 1. UnitTestGC

Type: Functional, Dynamic, Automated

Initial State: New session.

Input: location of target object  $(X, Y, 0)$  that range of X and Y is  $[-1000, 1000]$  and object cutting function provide by platform. [Object cutting function is provided by developer that is only used for doing unit testing. Users don't need to input this when using the program. Also no such input in System test, that needs input from user only. —Author]

Output: A list of coordinates of all pieces in format  $(x_n, y_n, z_n)$  in the console.

How test will be performed: Automated unit testing.

## 6.3 Unit test for angles calculation

### 1. UnitTestAC1

Type: Functional, Dynamic, Automated

Initial State: New session.

Input: Normal cases for  $(X, Y, 0)$ ,  $(x_n, y_n, z_n)$ . Range of X and Y is  $[-1000, 1000]$ . Example:  $(10, 10, 0); (31, -5, -10)$

Output:  $\theta_1 = \arctan \frac{|z_n|}{\sqrt{x_n^2 + y_n^2}}$  and  $\theta_2 = \arctan \frac{y_n}{x_n}$

How test will be performed: Automated unit testing.

## 2. UnitTestAC2

Type: Functional, Dynamic, Automated

Initial State: New session.

Input: Edge cases for  $(X, Y, 0)$ ,  $(x_n, y_n, z_n)$ . Example:  $(1000, 1000, 0); (-1000, -1000, -1000)$

Output:  $\theta_1 = \arctan \frac{|z_n|}{\sqrt{x_n^2 + y_n^2}}$  and  $\theta_2 = \arctan \frac{y_n}{x_n}$

How test will be performed: Automated unit testing.

## 6.4 Unit test for displacement calculation

### 1. UnitTestDCA

Type: Functional, Dynamic, Automated

Initial State: New session.

Input:  $E = [1 - 10]$ ,  $(X, Y, 0)$  that range of X and Y is  $[-1000, 1000]$ ,  $\mu_k = [0, 1]$ , time t, and  $\theta_1, \theta_2, v_0$  that are result from 5.1.4, 5.1.3.

Output:  $S_x = v_0 \cdot \cos\theta_1 \cdot \cos\theta_2 \cdot t$ ,  $S_y = v_0 \cdot \cos\theta_1 \cdot \sin\theta_2 \cdot t$  and  $S_z = v_0 \cdot \sin\theta_1 \cdot t - \frac{1}{2}gt^2$

How test will be performed: Automated unit testing. [Will perform finite test cases for different input in valid range. —Author]

### 2. UnitTestDCG

Type: Functional, Dynamic, Automated

Initial State: New session.

Input:  $E = [1 - 10]$ ,  $(X, Y, 0)$  that range of X and Y is  $[-1000, 1000]$ ,  $\mu_k = [0, 1]$ , time t and  $\theta_1, \theta_2, v_0$  that are result from 5.1.4, 5.1.3

Output:  $a = \mu_k g$ ,  $S_x = v_0 \cdot \cos\theta_1 \cdot \cos\theta_2 \cdot t - \frac{1}{2}at^2$  and  $S_y = v_0 \cdot \cos\theta_1 \cdot \sin\theta_2 \cdot t - \frac{1}{2}at^2$  [Acceleration  $a$  is an intermediate result that doesn't need separate module and system test. —Author]

How test will be performed: Automated unit testing.

## 7 Appendix

### 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.

symbol	value	description
$g$	$9.8m/s^2$	gravity acceleration

### 7.2 Usability Survey Questions?

Usability survey will not be done for this project.