Module Interface Specification for Breaking Effect

Marshall Xiaoye Ma

November 19, 2017

1 Revision History

Date	**Version**	**Notes**
Date 2017-11-17	1.0	New doc

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at https://github.com/MaXiaoye/cas741/blob/master/Doc/SRS/SRS.pdf

[Also add any additional symbols, abbreviations or acronyms —SS]

Contents

3 Introduction

The following document details the Module Interface Specifications for Breaking Effect.

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.

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.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://github.com/MaXiaoye/cas741.

4 Notation

[You should describe your notation. You can use what is below as a starting point. —SS]

The structure of the MIS for modules comes from Hoffman and Strooper (1995), with the addition that template modules have been adapted from Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | ... | c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by Program Name.

Data Type	**Notation**	**Description**
character	char	a single symbol or digit
integer	$\mathbb{Z}$	a number without a fractional component in (-∞, ∞)
natural number	$\mathbb{N}$	a number without a fractional component in [1, ∞)
real	$\mathbb{R}$	any number in (-∞, ∞)
instance of class TargetObject	TargetObj	An object instance of class TargetObject that extended from Object class provided by Unity3D
instance of class PieceObject	PieceObj	An object instance of class PieceObject that extended from Object class provided by Unity3D

The specification of Program Name uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, Program Name uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	**Level 2**
Hardware-Hiding Module	
Behaviour-Hiding Module	Input Module Target Object Module Piece Object Module Obtaining gravity center module Angle calculation module Displacement in the air calculation module Displacement on the ground calculation module
Software Decision Module	Object cutting module Output Module

Table 1: Module Hierarchy

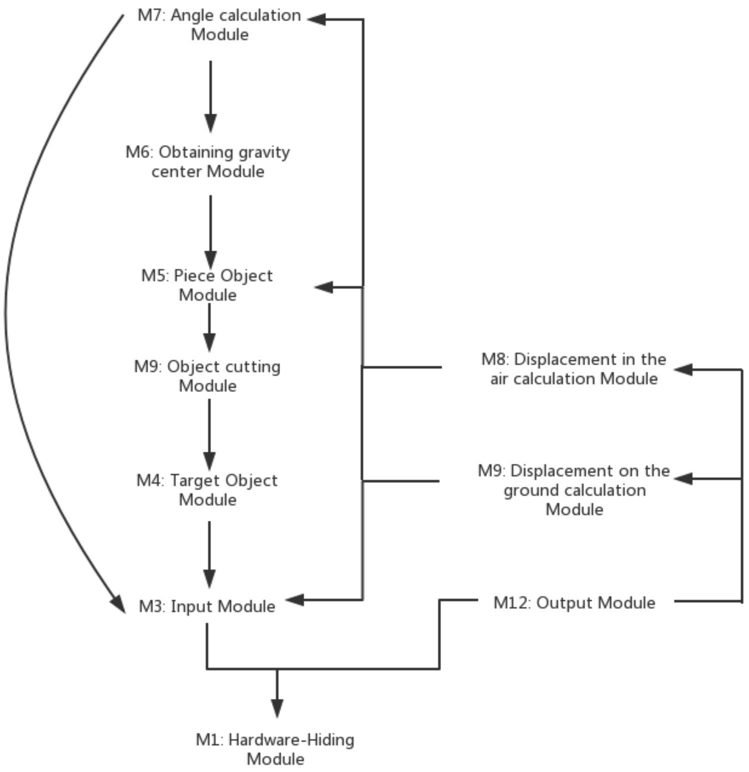
M7: Angle calculation Module
M6: Obtaining gravity center Module
M5: Piece Object Module
M9: Object cutting Module
M4: Target Object Module
M3: Input Module
M8: Displacement in the air calculation Module
M9: Displacement on the ground calculation Module
M12: Output Module
M1: Hardware-Hiding Module

Figure 1: Use hierarchy among modules

6 MIS of Input Module(M3)

This module collect input data to do verification and store in corresponding variables. Include position of target object, explosion level, coefficient of ground friction.

6.1 Module

InputModule

6.2 Uses

Hardware-Hiding Module (M1)

6.3 Syntax

6.3.1 Exported Access Programs

Name	In	Out	Exceptions
GetInitPos()	TargetObject	$\mathbb{R}^3$	InvalidObject
μ_k	$\mathbb{R}$	-	TypeError
E	$\mathbb{Z}$	-	TypeError
InputVerify()	$\mathbb{R}^5$	void	OutOfScope

6.4 Semantics

6.4.1 State Variables

$(X, Y, Z) : \mathbb{R}^3$

6.4.2 Access Routine Semantics

GetInitPos():

- transition: $X = Object.X; Y = Object.Y; Z = Object.Z$. [Values in other module (target object) are changed —Author]

- output: None

- exception: exc := (TargetObject = null ⇒ NoObjectException)

InputVerifiy():

- transition: N/A

- output: Exceptions or None.

- exception:[Different kinds of exceptions for different invalid inputs. —Author]
  exc := $(\mu_k = \text{null} \Rightarrow \text{NoMuException})$
  exc := $(\text{ExplosionLv} = \text{null} \Rightarrow \text{NoELvException})$
  exc := $(X, Y, Z \notin \mathbb{R} \vee (X, Y, Z \leq -1000) \vee (X, Y, Z \geq 1000) \Rightarrow \text{InvalidCoorException})$
  exc := $(E \notin \mathbb{R} \vee (E \leq 0) \vee (E \geq 10) \Rightarrow \text{InvalidELvException})$
  exc := $(\mu_k \notin \mathbb{R} \vee (\mu_k \leq 0) \vee (\mu_k \geq 1) \Rightarrow \text{InvalidMuException})$

7 MIS of target object module(M4)

External function provided by platform

7.1 Module

TarObjModule

7.2 Uses

Input Module(M3)

7.3 Syntax

7.3.1 Exported Access Programs

Name	In	Out	Exceptions
X	$\mathbb{R}$	-	-
Y	$\mathbb{R}$	-	-
Z	$\mathbb{R}$	-	-

7.4 Semantics

7.4.1 State Variables

None

7.4.2 Access Routine Semantics

None

8 MIS of piece object module(M5)

External function provided by platform

8.1 Module

ObjCutModule

8.2 Uses

Input Module(M3)

8.3 Syntax

8.3.1 Exported Access Programs

Name	In	Out	Exceptions
x	$\mathbb{R}$	-	-
y	$\mathbb{R}$	-	-
z	$\mathbb{R}$	-	-
onGround	Boolean	-	-
θ_1	$\mathbb{R}$	-	-
θ_2	$\mathbb{R}$	-	-

8.4 Semantics

8.4.1 State Variables

onGround: Boolean [It is a variable in each PieceObj. That means program checks this variable each frame to call different IM as well as using equations —Author]

8.4.2 Access Routine Semantics

None

9 MIS of acquiring gravity center module (M6)

Call cutting function provided by Unity3D to split target object into pieces then do traversal to obtain gravity centers of all pieces. [Each piece is an object. Gravity center is a value in PieceObj —Author]

9.1 Module

ObtainGCModule

9.2 Uses

Input Module(M3)
Object cutting module(M13)

9.3 Syntax

9.3.1 Exported Access Programs

Name	In	Out	Exceptions
Traverse()	PieceObj;$\mathbb{R}$	List of PieceObj	-

9.4 Semantics

9.4.1 State Variables

None

9.4.2 Access Routine Semantics

Traverse():
This function do a tranersal to assign gravity center coordinates to all PieceObj, return a list of PieceObj.

- transition: None
- output: List of PieceObj
- exception: None

10 MIS of Angle calculation module(M7)

Calculate the angle between initial speed v_0 and horizontal θ_1. Calculate the angle between x axiom and projection on horizontal of initial speed θ_2. [θ_1 and θ_2 are values of each PieceObj —Author]

10.1 Module

AngleCalModule

10.2 Uses

Input Module(M3)
Obtaining gravity center module (M6)

10.3 Syntax

10.3.1 Exported Access Programs

Name	In	Out	Exceptions
AngleSH	$\mathbb{R}$;PieceObj	$\mathbb{R}$	-
AngleXV	$\mathbb{R}$;PieceObj	$\mathbb{R}$	-

10.4 Semantics

10.4.1 State Variables

None

10.4.2 Access Routine Semantics

AngleSH():
Equation: $\theta_1 = arctan\frac{|z_n|}{\sqrt{(x_n-X)^2+(y_n-Y)^2}}$

- transition: None
- output: $\theta_1 : \mathbb{R}$
- exception: None

AngleXV():
Equation: $\theta_2 = arctan\frac{|y_n-Y|}{|x_n-X|}$

- transition: None
- output: $\theta_2 : \mathbb{R}$
- exception: None

11 MIS of Displacement in the air calculation module(M8)

Calculate and output trace of motion for each piece in the air by using follow equations.

11.1 Module

DisAirCalModule

11.2 Uses

Input Module(M3)
Angle calculation module(M7)

11.3 Syntax

11.3.1 Exported Access Programs

Name	In	Out	Exceptions
DisAirCalX	$\mathbb{R}$; PieceObj; TargetObject	$\mathbb{R}$	-
DisAirCalY	$\mathbb{R}$; PieceObj; TargetObject	$\mathbb{R}$	-
DisAirCalZ	$\mathbb{R}$; PieceObj; TargetObject	$\mathbb{R}$	-

11.4 Semantics

11.4.1 State Variables

None

11.4.2 Access Routine Semantics

DisAirCalX():
Equation: $S_x = v_0 \cdot cos\theta_1 \cdot cos\theta_2 \cdot t$

- transition: None
- output: $S_x : \mathbb{R}$
- exception: None

DisAirCalY():
Equation: $S_y = v_0 \cdot cos\theta_1 \cdot sin\theta_2 \cdot t$

- transition: None
- output: $S_y : \mathbb{R}$
- exception: None

DisAirCalX():
Equation: $S_z = v_0 \cdot sin\theta_1 \cdot t - \frac{1}{2}gt^2$

- transition: None

- output: $S_z : \mathbb{R}$
- exception: None

12 MIS of Displacement on the ground calculation module(M9)

Calculate and output trace of motion for each piece on the ground by using follow equations.

12.1 Module

DisGroCalModule

12.2 Uses

Input Module(M3) Angle calculation module(M7)

12.3 Syntax

12.3.1 Exported Access Programs

Name	In	Out	Exceptions
DisGroCalX	$\mathbb{R}$; PieceObj; TargetObject	$\mathbb{R}$	-
DisGroCalY	$\mathbb{R}$; PieceObj; TargetObject	$\mathbb{R}$	-

12.4 Semantics

12.4.1 State Variables

None

12.4.2 Access Routine Semantics

DisGroCalX():
Euqation: $a = \mu_k g$; $S_x = v_0 \cdot cos\theta_1 \cdot cos\theta_2 \cdot t - \frac{1}{2}at^2$

- transition: None
- output: $S_x, S_y, S_z : \mathbb{R}$
- exception: None

DisGroCalY():
Euqation: $a = \mu_k g$; $S_y = v_0 \cdot cos\theta_1 \cdot sin\theta_2 \cdot t - \frac{1}{2}at^2$

- transition: None
- output: $S_x, S_y, S_z : \mathbb{R}$
- exception: None

13 MIS of Object cutting Module(M13)

External function provided by platform. Split target object to pieces.

13.1 Module

ObjCutModule

13.2 Uses

Input Module(M3)

13.3 Syntax

13.3.1 Exported Access Programs

Name	In	Out	Exceptions
cut()	TargetObj	PieceObj	-

13.4 Semantics

13.4.1 State Variables

None

13.4.2 Access Routine Semantics

cut():

- transition: None
- output: PieceObj
- exception: None

14 MIS of Output Module(M12)

Unity3D interface with codes by calling function update() each frame. Unity3D convert data into visualization.

14.1 Module

OutputModule

14.2 Uses

Displacement in the air calculation module(M8)
Displacement on the ground calculation module(M9)

14.3 Syntax

14.3.1 Exported Access Programs

Name	In	Out	Exceptions
update()	codes to be run each frame	Visualization	-
traverseDis()	List of PieceObj	-	-
translate()	$S_x, S_y, S_z : \mathbb{R}^3$	PieceObj.position	-

14.4 Semantics

14.4.1 State Variables

Screen;
PieceObj

14.4.2 Access Routine Semantics

update():

- transition: $\mathbb{R}^3 \rightarrow$ Visualization
- output: None
- exception: None

translate():

- transition: PieceObj.position
- output: None

- exception: None

References

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL http://citeseer.ist.psu.edu/428727.html.

15 Appendix

[Extra information if required —SS]