Module Interface Specification for ...

Author Name

November 17, 2017

1 Revision History

Date	**Version**	**Notes**
Date 1	1.0	Notes
Date 2	1.1	Notes

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at [give url —SS]

[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 (-∞, ∞)

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 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

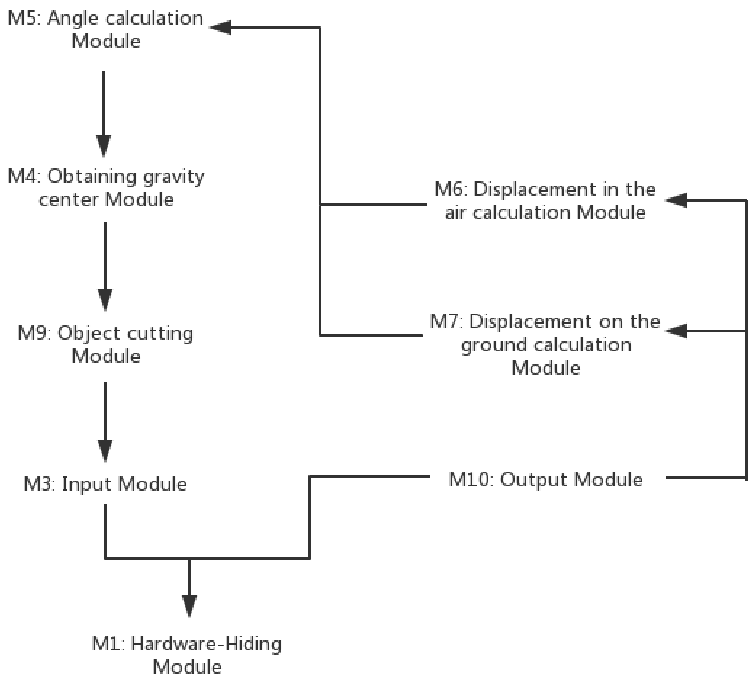
M5: Angle calculation Module
M4: Obtaining gravity center Module
M9: Object cutting Module
M3: Input Module
M1: Hardware-Hiding Module
M6: Displacement in the air calculation Module
M7: Displacement on the ground calculation Module
M10: Output Module

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. [Time is input provided by Unity3D. —Author]
[Can I put description here? —Author]

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	$\mathbb{R}$	InvalidObject
InputVerifiy	$\mathbb{R}$	void	OutOfScope

6.4 Semantics

6.4.1 State Variables

[Input module may not have state variables. But it do return different result depends on input data ? —Author]
InputVerify: Boolean

6.4.2 Access Routine Semantics

GetInitPos():

- transition: [No transition —Author]
- output: $X = Object.X; Y = Object.Y; Z = Object.Z.$ $(X, Y, Z) : \mathbb{R}$ [No output but assignment —Author]
- exception: exc := (TargetObject = null ⇒ NoObjectException)

InputVerifiy():

- transition: N/A
- output: Exceptions or None.

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

7 MIS of Obtaining gravity center module (M4)

Call cutting function provided by Unity3D to split target object then do a traversal to obtain location of gravity centers for all pieces. [Each piece is an objects, gravity center is one attribute in PiecePbj —Author]

7.1 Module

ObtainGCModule

7.2 Uses

Input Module(M3)
Object cutting module(M11)

7.3 Syntax

7.3.1 Exported Access Programs

Name	In	Out	Exceptions
CuttingFun	TargetObject	PieceObj	ExternalException
Traverse	PieceObj;$\mathbb{R}$	List of PieceObj	-

[Put all functions in the table or in Access Routine Semantics part? —Author]

7.4 Semantics

7.4.1 State Variables

[No state variables —Author]

7.4.2 Access Routine Semantics

[Do I need to put variables here? —Author]
Traverse():

- transition: PieceObj $\rightarrow$ List of PieceObj [data type transition? —Author]
- output: List of PieceObj
- exception: None

8 MIS of Angle calculation module(M5)

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]
[Can I put equations in IM here? —Author]
$\theta_1 = arctan\frac{|z_n|}{\sqrt{(x_n-X)^2+(y_n-Y)^2}}$
$\theta_2 = arctan\frac{|y_n-Y|}{|x_n-X|}$

8.1 Module

AngleCalModule

8.2 Uses

Input Module(M3)
Obtaining gravity center module (M4)

8.3 Syntax

8.3.1 Exported Access Programs

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

8.4 Semantics

8.4.1 State Variables

None

8.4.2 Access Routine Semantics

AngleSH():

- transition: [No transition —Author]
- output: $\theta_1 : \mathbb{R}$
- exception: None

AngleXV():

- transition: [No transition —Author]
- output: $\theta_2 : \mathbb{R}$
- exception: None

9 MIS of Displacement in the air calculation module(M6)

Calculate and output trace of motion for each piece in the air by using follow equations.
$S_x = v_0 \cdot cos\theta_1 \cdot cos\theta_2 \cdot t$
$S_y = v_0 \cdot cos\theta_1 \cdot sin\theta_2 \cdot t,$
$S_z = v_0 \cdot sin\theta_1 \cdot t - \frac{1}{2}gt^2$ [S is displacement in one frame, t is the gap between each frame(δt). —Author]

9.1 Module

DisAirCalModule

9.2 Uses

Input Module(M3)
Angle calculation module(M5)

9.3 Syntax

9.3.1 Exported Access Programs

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

9.4 Semantics

9.4.1 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]

9.4.2 Access Routine Semantics

DisAirCal():

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

10 MIS of Displacement on the ground calculation module(M7)

Calculate and output trace of motion for each piece on the ground by using follow equations.
$S_x = v_0 \cdot cos\theta_1 \cdot cos\theta_2 \cdot t - \frac{1}{2}at^2$
$S_y = v_0 \cdot cos\theta_1 \cdot sin\theta_2 \cdot t - \frac{1}{2}at^2$

10.1 Module

DisGroCalModule

10.2 Uses

Input Module(M3) Angle calculation module(M5)

10.3 Syntax

10.3.1 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
DisGroCal	$\mathbb{R}$; PieceObj; TargetObject	$\mathbb{R}$	-

10.4 Semantics

10.4.1 State Variables

onGround: Boolean

10.4.2 Access Routine Semantics

DisGroCal():

- transition: $a = \mu_k g$ [a is intermediate result. —Author]
- output: $S_x, S_y, S_z : \mathbb{R}$
- exception: None

11 MIS of Object cutting Module(M11)

External function provided by platform

11.1 Module

ObjCutModule

11.2 Uses

Input Module(M3)

11.3 Syntax

11.3.1 Exported Access Programs

Name	In	Out	Exceptions
cut()	TargetObj	PieceObj	-

11.4 Semantics

11.4.1 State Variables

None

11.4.2 Access Routine Semantics

cut():

- transition: TargetObj $\rightarrow$ PieceObj
- output: PieceObj
- exception: None

12 MIS of Output Module(M10)

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

12.1 Module

Output Module

12.2 Uses

Displacement in the air calculation module(M6)
Displacement on the ground calculation module(M7)

12.3 Syntax

12.3.1 Exported Access Programs

Name	In	Out	Exceptions
update()	-	Visualization	-
traverseDis()	List of PieceObj	-	-
translate()	$\mathbb{R}$	Visualization	-

[Call translate() function for all PieceObj each frame —Author]

12.4 Semantics

12.4.1 State Variables

12.4.2 Access Routine Semantics

update():

- transition: $\mathbb{R} \rightarrow$ Visualization
- 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.

13 Appendix

[Extra information if required —SS]