Module Interface Specification for Breaking Effect

Marshall Xiaoye Ma

December 3, 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

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

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**
natural number	$\mathbb{N}$	a number without a fractional component in $[1, \infty)$
real	$\mathbb{R}$	any number in $(-\infty, \infty)$
String	String	represents sequences of characters.
Object	Object	A data structure to store attributes of input target object that provided by Unity3D.
PieceObject	PieceObj	A data structure to store attributes of pieces that generated as intermediate steps.

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 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	Target Object Module Object cutting module Output Module

Table 1: Module Hierarchy

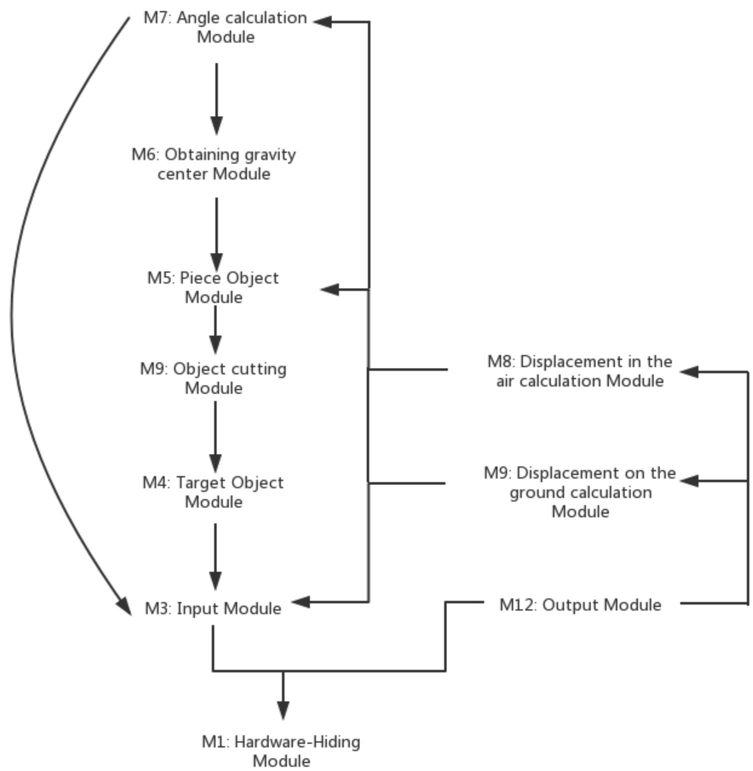

Figure 1: Use hierarchy among modules

6 MIS of Input Module(M3)

This module collect verifies input from user 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
μ_k	$\mathbb{R}$	-	InvalidInput
E	$\mathbb{R}$	-	InvalidInput
$TargetObj$	$Object$	-	InvalidInput
InputVerify()	$\mathbb{R}^2; TargetObj$	void	InvalidInput

[Object is a 3D model in Unity3D, which contains its position (X, Y, Z). User needs provide a 3D model and attach the program to it. —Author]

6.4 Semantics

6.4.1 State Variables

None

6.4.2 Access Routine Semantics

InputVerifiy():

- transition: N/A
- output: Exceptions or None.
- exception:[Different kinds of exceptions for different invalid inputs. —Author]
  exc := (μ_k = null $\Rightarrow$ NoMuException)
  exc := (ExplosionLv = null $\Rightarrow$ NoELvException)
  exc := $(X, Y, Z \notin \mathbb{R} \vee (X, Y, Z \leq -1000) \vee (X, Y, Z \geq 1000) \Rightarrow$ InvalidCoorException)
  exc := $(E \notin \mathbb{R} \vee (E \leq 0) \vee (E \geq 10) \Rightarrow$ InvalidELvException)
  exc := $(\mu_k \notin \mathbb{R} \vee (\mu_k \leq 0) \vee (\mu_k \geq 1) \Rightarrow$ InvalidMuException)

7 MIS of target object module(M4)

Object class 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	$\mathbb{R}$	-	-
position	$\mathbb{R}^3$	-	-
destory()	-	-	-

X, Y, Z are coordinates of object. position is 3D vector that contains X, Y, Z while it is also considered as gravity center location of the object, destroy() function removes the object.

7.4 Semantics

7.4.1 State Variables

KeyCode.Space: Boolean. This bool value indicates if key space is pressed on keyboard.

7.4.2 Access Routine Semantics

destroy():
[We assume the explosion happens when ”space” is pressed. The target object is removed from scene at the same time. —Author]

- transition: target object → null
- output: None
- exception: None

8 MIS of piece object module(M5)

Customize class for pieces that extend object class provided by platform. Pieces are generated after explosion happens to replace original target object from input.

8.1 Module

ObjCutModule

8.2 Uses

Input Module(M3)

8.3 Syntax

8.3.1 Exported Access Programs

Name	In	Out	Exceptions
Tag	String	-	-
x	$\mathbb{R}$	-	-
y	$\mathbb{R}$	-	-
z	$\mathbb{R}$	-	-
position	$\mathbb{R}^3$	-	-
onGround	Boolean	-	-
θ_1	$\mathbb{R}$	-	-
θ_2	$\mathbb{R}$	-	-
Move()	Boolean	-	-
Translate()	$\mathbb{R}^3$	Visualization	-

Assumptions:
Tag, x, y, z and position, Translate() are inheritated from parent class

- Tag is string "piece" that indicates the object is an instance piece object.
- x, y, z are coordinates of object.
- position is 3D vector that contains x, y, z.
- Translate() controls motion of the object.
- onGround indicates if the object is on the ground.
- θ_1, θ_2 are calculated and described in M7.

- Move() controls motion of the object by calling Translate(). It check onGround firstly to make sure the object is in the air or on the ground. Based on value of bool variable onGround, that call and provide corresponding destination as input to Translate(). Destination to Translate() is calculated by M8 and M9.
  [The definition of this class doesn't use M8 and M9 while result from M8 and M9 are only passed to function Move() that is called in M12. So I don't put M8 and M9 in Uses part. —Author]

8.4 Semantics

8.4.1 State Variables

None

8.4.2 Access Routine Semantics

Move():

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

Translate():

- transition: None
- output: Visualization
- exception: None

9 MIS of acquiring pieces module (M6)

Call cutting function provided by Unity3D to split target object into pieces then do traversal to acquire all pieces. [Each piece is stored as an instance of class piece object defined in M5. Gravity center is position value in PieceObj —Author]

9.1 Module

ObtainGCModule

9.2 Uses

Input Module(M3)
Object cutting module(M11)

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 Traverse to store all piece objects into a list by filtering tags with "piece". Piece objects are initialed after target object is cutted. [tags for piece objects are defined in M5 —Author]

- 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: $\theta_1 = arctan\frac{y_n}{\sqrt{(x_n-X)^2+(z_n-Y)^2}}$
Convert equation to codes:
Mathf.Atan(PieceObj.y / Mathf.Sqrt(Mathf.Pow(PieceObj.x - TargetObj.x,2) + Mathf.Pow(PieceObj.z - TargetObj.z,2)));

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

AngleXV():
Equation: $\theta_2 = arctan\frac{x_n-X}{z_n-Z}$
Convert equation to codes:
Mathf.Atan2(PieceObj.x - TargetObj.x, PieceObj.z - TargetObj.z)

- 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: $v_0 = 10 * E, S_x = v_0 \cdot cos\theta_1 \cdot sin\theta_2 \cdot \Delta t$ [Based on R8 in SRS that value of initial velocity given by explosion is ten times input E unit length in unity per second. Δt is the gap between each frame that input from unity3D —Author]
Convert equation to codes:
initSpeed * Mathf.Cos(PieceObj.theta1) * Mathf.Sin(PieceObj.theta2) * Time.deltaTime

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

DisAirCalY():
Equation: $S_y = (v_0 \cdot sin\theta_1 - g \cdot t) \cdot \Delta t - \frac{1}{2} g \cdot \Delta t^2$ [t is real time since the explosion happens. So that $v_0 \cdot sin\theta_1 - g \cdot t$ means the initial speed on vertical direction at the beginning of each

frame —Author]
Convert equation to codes:
(initSpeed * Mathf.Sin(PieceObj.theta1) + g * Time.realtimeSinceStartup) * Time.deltaTime + 1 / 2 * g * Time.deltaTime * Time.deltaTime

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

DisAirCalZ():
Equation:$S_z = v_0 \cdot cos\theta_1 \cdot cos\theta_2 \cdot \Delta t$
Convert equation to codes:
initSpeed * Mathf.Cos(PieceObj.theta1) * Mathf.Cos(PieceObj.theta2) * Time.deltaTime)

- 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	$\mathbb{R}$	-
DisGroCalZ	$\mathbb{R}$; PieceObj; TargetObject	$\mathbb{R}$	-

12.4 Semantics

12.4.1 State Variables

None

12.4.2 Access Routine Semantics

DisGroCalX():
Euqation: $a = \mu_k g$; $S_x = (v_0 \cdot cos\theta_1 \cdot sin\theta_2 - at) \cdot \Delta t - \frac{1}{2}a \cdot \Delta t^2$
Convert equation to codes:
(initSpeed * Mathf.Sin(PieceObj.theta2) * Mathf.Cos(PieceObj.theta1) - a * Time.realtimeSinceStartup) * Time.deltaTime - 1 / 2 * a * Time.deltaTime * Time.deltaTime

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

DisGroCalZ():
Euqation: $a = \mu_k g$; $S_z = (v_0 \cdot cos\theta_1 \cdot cos\theta_2 - at) \cdot \Delta t - \frac{1}{2}a \cdot \Delta t^2$
Convert equation to codes:
(initSpeed * Mathf.Cos(PieceObj.theta2) * Mathf.Cos(PieceObj.theta1) - a * Time.realtimeSinceStartup) * Time.deltaTime - 1 / 2 * a * Time.deltaTime * Time.deltaTime

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

13 MIS of Object cutting Module(M11)

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	-
Instantiate()	-	-	-

13.4 Semantics

13.4.1 State Variables

None

13.4.2 Access Routine Semantics

cut():

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

Instantiate():

- transition: None
- output: visualization [draw piece objects in the scene —Author]
- 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	-
start()	-	-	-

14.4 Semantics

14.4.1 State Variables

Screen;
PieceObj

14.4.2 Access Routine Semantics

start():
Start is called on the frame when a script is enabled just before any of the Update methods is called the first time.

- transition: null → target object [target object is initialized in scene —Author]
- output: None
- exception: None

update():
Update is called every frame. In update(), we listen if space is pressed as start point of the explosion. It also keeps updating status of all objects in the scene to convert location of objects to visualization that can be seen on the screen.

- transition: Piece objects → 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`.

15 Appendix