

# **Developing a processing application combining the 3D objects and sound**

**Zhang Weihui**

13.11.2006

StudentID: u4070557

Supervisor: Henry Gardner & Alistair Rendell

Subject Name: eScience Project I

**Faculty of Engineering and Information Technology  
Department of Computer Science**

## Abstract

Writing a program, which is combining the 3D objects and sound will use the **Processing** development environment that is open source software as the development environment. All codes should be constructed by using the format of OpenGL1.

Most importantly, this project is asked for using a peripheral device such as joypad or joysticks to control the movement of the 3D objects created. As a result, user can use joystick to control that 3D object instead of mouse and keyboard.

*Note:*

*The 3D object in this project, it will be designed as a Robot arm (see the picture of pic 1).*

---

*More details about OpenGL, please check out the book:  
OpenGL Programming Guide, Fourth Edition (Red book)*

# Table of Contents

Introduction .....	4
Requirements .....	5
Project Timetable .....	6
Modelling .....	7
Design Phase .....	17
Implementation and Testing .....	18
Conclusion .....	19
Appendix A .....	20
Appendix B .....	22
Appendix C .....	25

# INTRODUCTION

## **Background**

By now, there are many approaches to carry out 3D animation such as 3D studio tools or the Java3D language. In particular, OpenGL graphical system has a long history as a software interface; it helps programmers to be easy to create the interactive programs. OpenGL graphical interfaces can be written by using Java or C languages or other languages. In a sense, there is high requirement for programmer to have the knowledge about Java or C and many interface functions of OpenGL self.

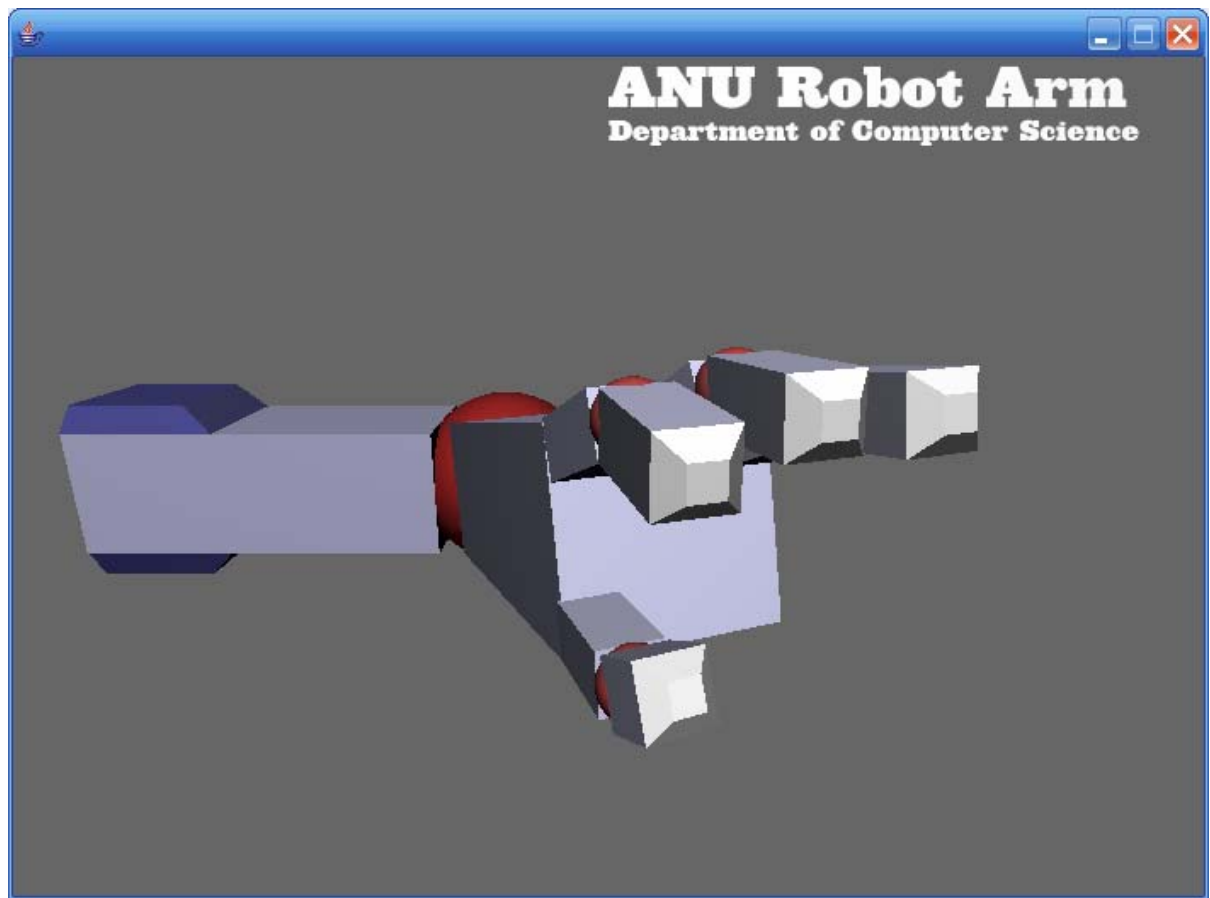
In the meanwhile, **Processing**<sup>1</sup> is an open source software environment for people who want to program images, animation, and sound. Furthermore, OpenGL is one of libraries attached inside Processing. Processing is created to teach fundamental of computer programming within a visual context and to serve as a software sketchbook and professional production tool. Therefore, it is flexible and effective software to implement 3D and 3D animation although its constructive language is kind of different with the normal construction of Java. The basic purpose of this project is to create an application to combine 3D objects (a Robot arm, see pic 1) and sound with OpenGL under **Processing** environment. In particular, the implementation of sound needs a plug-in application called **Ess**<sup>2</sup>. Furthermore, a peripheral device such as joystick or joypad will control the motion of robot arm instead of using keystrokes; the approach is to invoke the API of joystick is to control the movement of Robot arm by using the **proCONTROLL**<sup>3</sup> plug-in. In addition, the application would simulate the VR platform – the ANU Wedge.

---

*2 Ess is built on JavaSound and as such requires no additional plug-ins, just Java 1.4 or better.*

*More information please <http://www.tree-axis.com/Ess/>*

*3 The proCONTROLL library allows Processing to communicate with control devices like joysticks and joypads. More information please go to <http://texone.org/procontrol/>*



Pic 1

Picture one is the shape of robot arm constructed by OpenGL within **Processing** environment.

---

*More information about processing, user can check out the official web site of processing*  
<http://processing.org>

# REQUIREMENT

## **Basic Functionality of this Application**

The original project came from Dr. Henry Gardner who gave me some hints about how to write a nice Processing application, most importantly, I should study how to write OpenGL within Processing before the implementation. At the same time, the project coordinator Pascal Vuylsteker asked me to create the 3D images that could be giving user a feeling of VR. As the outcome, it is important to build necessary functionality for user.

- This software should provide some basic manipulation information for user and be able to display the 3D scene at different viewpoint.
- User can control the 3D objects motion through the keyboard clicking or mouse dragging.
- User can click the several keys at the same time in order to move two or more objects synchronously.
- User can use the sticks, sliders and buttons of joystick to control the movement of 3D object and the changes of sound. Furthermore, it also supports to move two or more objects synchronously.
- User can change the sound playing when moving different 3D objects.

According to the basic functionality listed above, some specific functionality/requirements are listed Appendix A.

# PROJECT TIMETABLE

Week	Dates	Notes	Eventually Done
1	17 - 23 Jul	Choose a supervisor/client/subject and get started!	Y
2	24 - 30 Jul	Requirements and Choice of Technics phase	Y
3	31 Jul – 6 Aug	Presentation of Project Topic, Requirements and Timetable.	Y
4	7 - 13 Aug	Modeling and Learning of Chosen Technics Phase	Y
5	14 - 20 Aug	A Java testing 3D program finished	Y
6	21 - 27 Aug	Modeling phase completed	Y
7	28 Aug – 3 Sep	Beginning of Implementation phase	Y  (First implementation finished)
8	18 - 24 Sep	Comments by supervisors	Y  (Using joystick or joypad instead of keyboard and mouse)
9	25 Sep – 1 Oct	Implementation of Comments	Y
10	2 – 8 Oct	Debugging	Y
11	9 – 15 Oct	Documentation	Y
12	16 – 22 Oct	Implementation completed	Y
13	23 – 29 Oct	Final cleaning and debugging	Y
14	30 Oct – 5 Nov	Documentation	Majority
15	6 – 12 Nov	Project Presentation	Y
16	13 – 19 Nov	Documentation	DONE

# **MODELING**

## **Background**

In the widely accepted software waterfall life cycle, software analysis and design are necessary phases before code implementation. Software analysis and design are tricky processes and sometimes involve more brainstorming than implementing code from certain point of view. Nowadays software is getting bigger and bigger, without deliberate analysis and design, the implementation phase will easily become unmanageable and final code will be in the danger of low reusability.

In the market, design approaches mainly divide into two groups: elaborative approach and translative approach. Elaborative approach is an iterative and incremental design approach, and it tries to use object oriented techniques to bridge the gap between an object oriented analysis model and a design phase. As the result, the analysis models are changing along the whole software life cycle, as do the design and implementation outcomes.

Shlaer-Mellor method is a representative translative design approach and has many successful applications in both structural programming and object oriented programming. Mainly because Shlaer-Mellor method completely separates stages of waterfall life cycle into different concerns, it is desirable to be used to depict software analysis model phase without convoluting with other phases. So it is possible for me to produce consistent models along software development life cycle (SDLC) by using the Shlaer-Mellor method.

## **Analysis**

As part of the preliminary analysis, a use case diagram (Figure 1) was constructed based on the software requirement specification presented in Requirements Phase. It

as assumed that the main users would be students, researchers and some programmers who are followed by the techniques of computer graphics. I also draw out software context diagram as below (Figure 2). Both use case diagram and software context diagram are used to describe the interaction between external entity and software system.

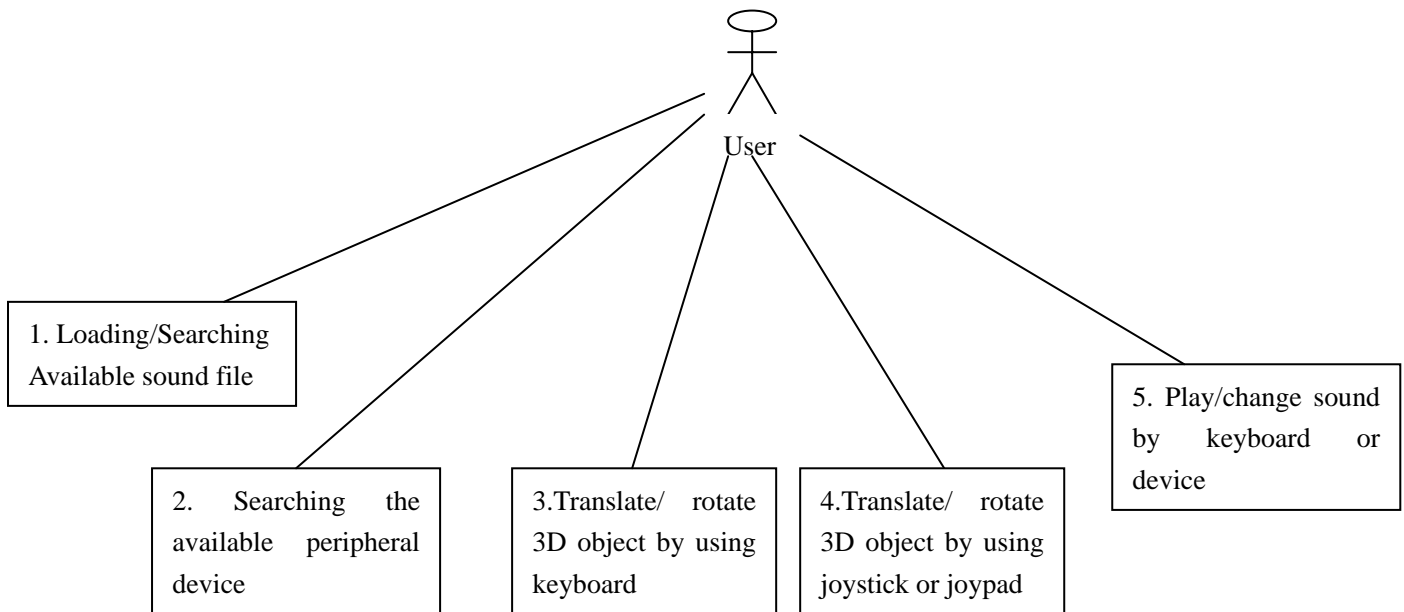


Figure 1: Use Case Diagram

### System Context Diagram

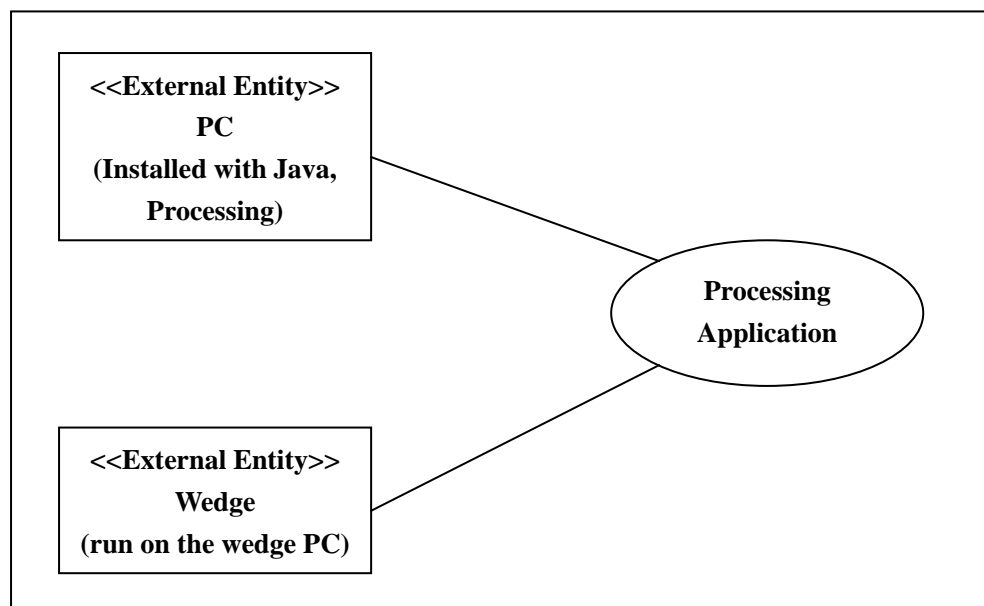


Figure 2: System Context Diagram

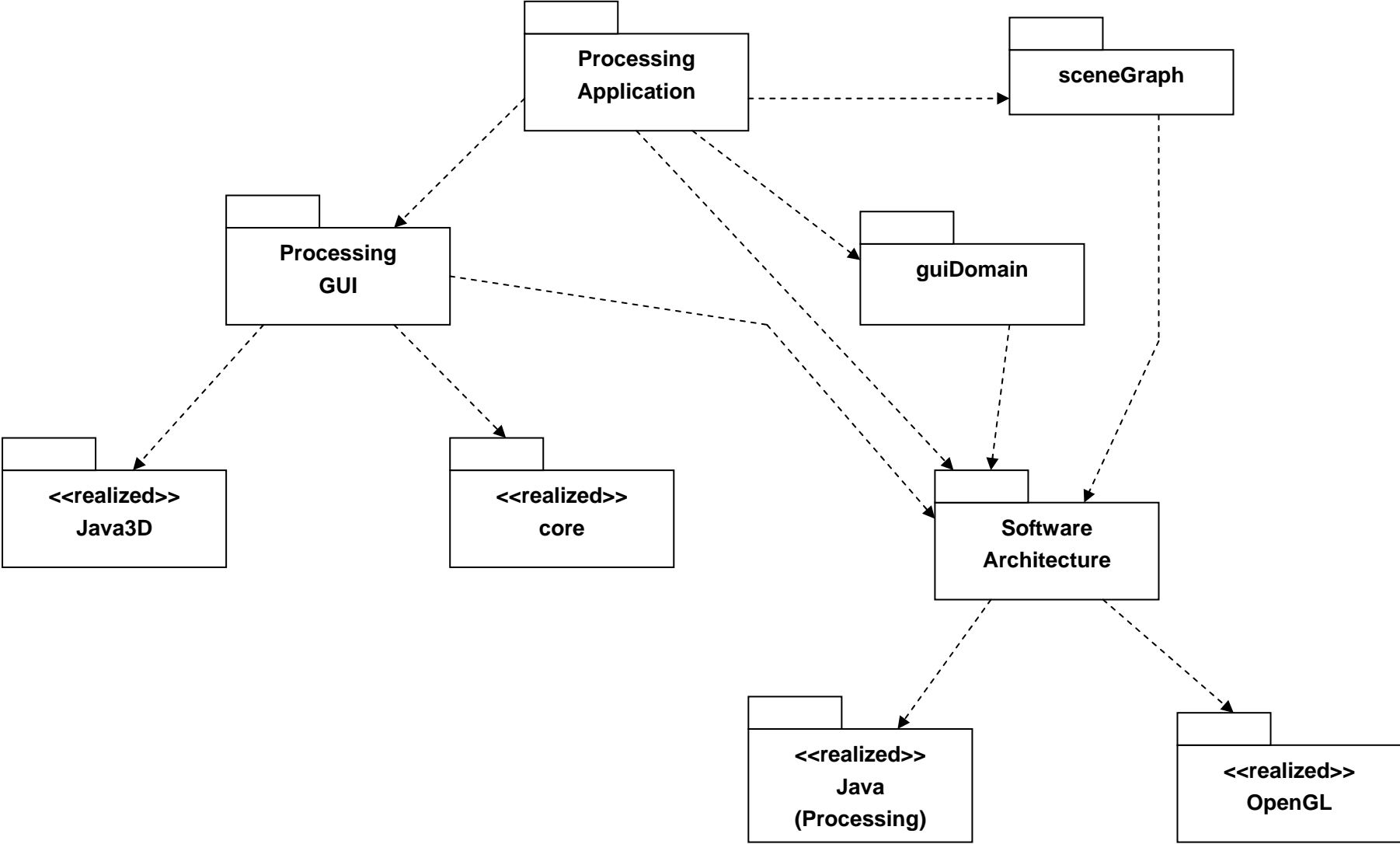
## **(*xT*UML) Object Oriented Analysis**

*xT*UML is the latest incarnation of Shlaer-Mellor method. Compared with Shlaer-Mellor method, it gives up traditional Shlaer-Mellor notation and replaces with a subset of Unified Modeling Language (UML). As a result, the outcome of *xT*UML analysis is UML models that consist of UML "pictures" and *xT*UML action language. Another notable difference between Shlaer-Mellor method and *xT*UML is that *xT*UML refines Shlaer-Mellor into Executable UML and Translatable UML. Executable UML is mainly oriented toward Software Analysis Phase and the analysis model depends on the model compiler to compile models into final code according to the translation rules and regulations built in the model compiler. Translatable UML mainly deals with how to build up model compiler by setting up software architecture and translation rules. It is more oriented towards Software Design Phase. In the report, I will use Executable UML to model the application in the Analysis Phase and apply Shlaer-Mellor in the Design Phase because Translatable UML is not discussed in Mellor's latest book <sup>2</sup>.

---

<sup>2</sup>S. J. Mellor and M. J. Balcer, *Executable UML: A Foundation for Model-Driven Architecture*, Addison-Wesley, 2002

**Domain Chart**



In an ideal domain model, an application domain deals with logical structure of the application; general service domain consists of scenegraph and guiDomain. Actually, in my domain chart, it should not put the application domain due to it is a difficult and the logic inside application domain is trivial such as creating/remove objects and relationship. So in the following content, I would not concern the application domain. guiDomain just deals with AWT/Swing and events handling. However, the sceneGraph deals with the real 3D world content representation of application.

Therefore, in my domain model, I assume that most application logic is inside the sceneGraph and guiDomain domains. Whereas, the following analysis, I would analyze the application logic, the sceneGraph without logic and guiDomain without logic. This might be different with the original domain chart, which is because the domain chart is the finalized product.

## **The Application Logic**

### 1. Requirements

- Allow user to run the **Processing** program.
- Allow user to rotate the whole 3D scene through clicking related keys.
- Allow user to load sound and play it when movement.
- Allow user to move 3D objects (including every components of robot arm).
- Allow user to press several keys at the same time to control objects
- Allow user to use joystick to control 3D objects movement and changes of sound.
- The generated code can be run at the Wedge PC.

### 2. Description

A user would be able to run the program and operate the 3D object. Moving it and play the relevant sound files (\*.wav) basing on the movement of 3D objects.

### 3. Candidate classes

3dModel

Sound

ProcessingApp

Device

### 4. Class Diagram

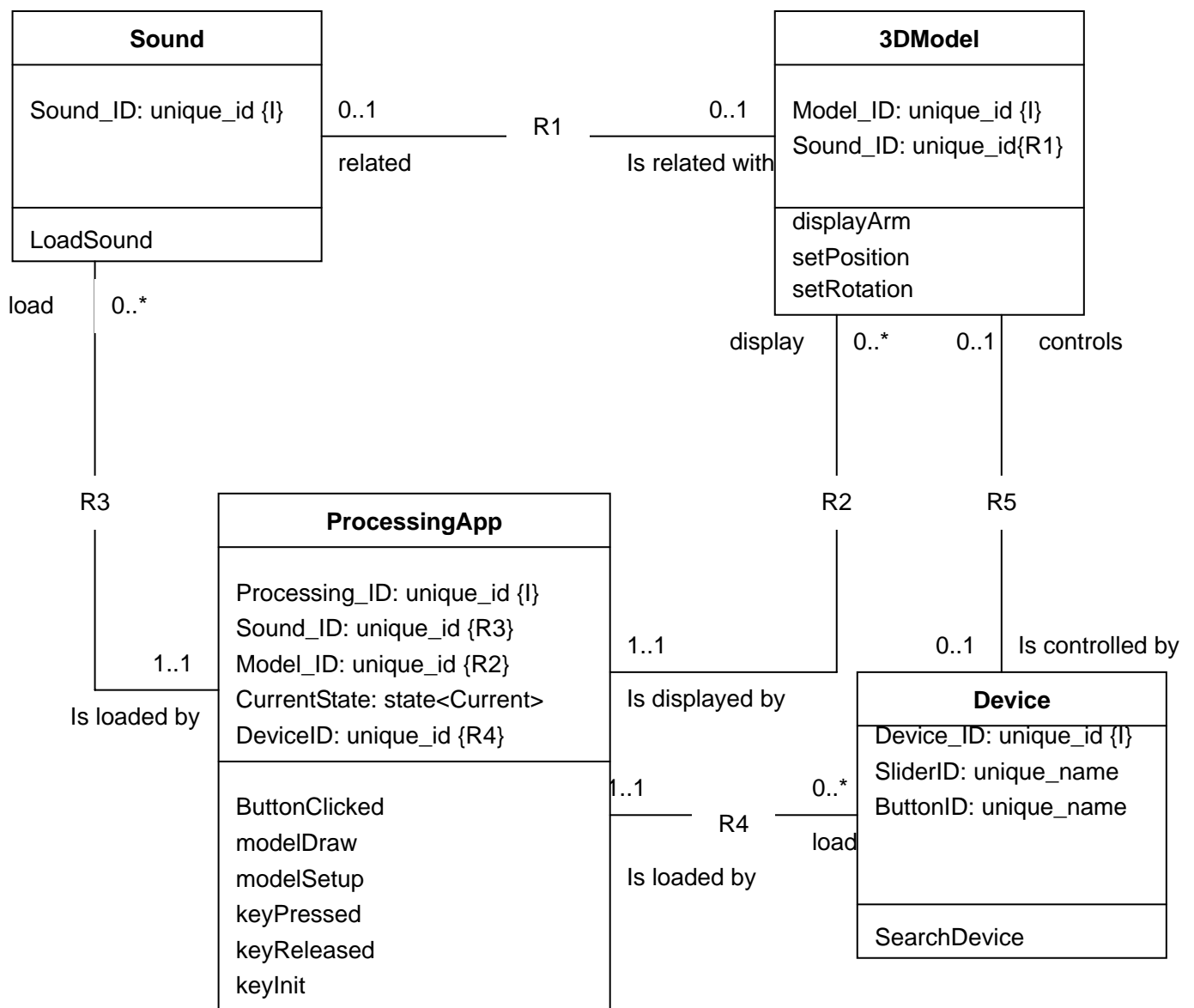


Figure 4: Class diagram of application logic

## 5. State Machine of ProcessingApp

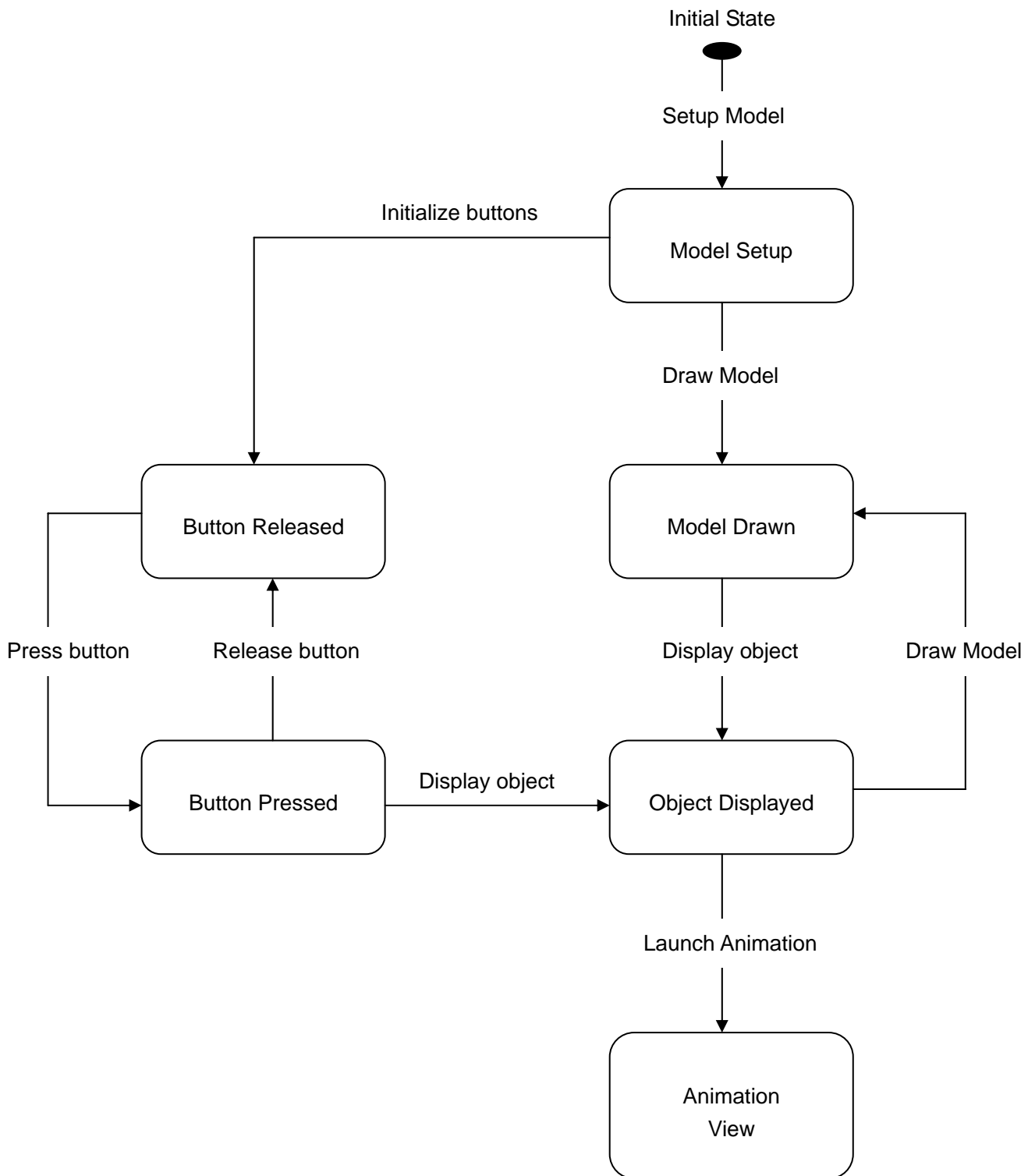


Figure 5: Non-trivial State Machine of ProcessingApp

## guiDomain without Logic

### 1. Requirements

- Actually, the application generated by **Processing**, there is no menubar or others popup windows. So in my **Processing** application, the GUI does not supply a complete usable workspace for use to utilize functions.
- However, in my application, the GUI needs to deal with event handling mechanisms. Such as key clicking.

### 2. Description

User activates the event-handling such as he/she click the relevant keys to move the robot arm. GUI will catch the event to response the user's motion.

### 3. Candidate Class

ProcessingApp (as the main frame)

EventHandle

### 4. Class Diagram (Information Model)

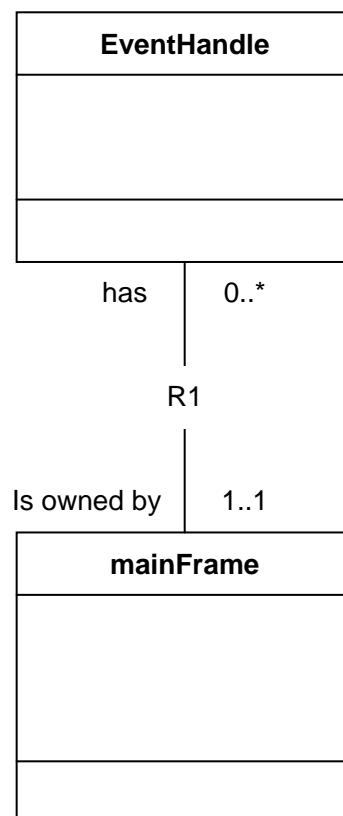


Figure 6: Class Diagram of guiDomain

## 5. State Machine

There is no meaningful state machine in guiDomain

## DESIGN PHASE

In executable UML, design stage is a trivial process in the SDLC. This is due to the function of model compiler that can translate xtUML models into final code.

Unfortunately, today's model compilers are far away from practicality. Even the newest version of BridgePoint does not support GUI and multi-thread translation. To translate xtUML models manually, we need design our own architecture and archetype by following the Shlaer-Mellor methodology to process the translation.

Before this project, I have used **Processing** to create some animation of 3D objects. At the last semester I chose the course of Java Introductory using Processing to create some 2D images<sup>3</sup>. Most importantly, I have enough knowledge about the OpenGL and how to use its functions to create 3D or 3D animated objects. Therefore I would be clear how to create an 3D object in a Processing environment with the OpenGL basically. Furthermore, xtUML model I learned before would be well applied in implementation stage of this project.

Basing on the domain chart (Figure 3), each domain is represented as a UML package. In the step by step of my program, subdirectory of guiDomain and sceneGraph represented the modeled domains in the domain chart. Additionally, some implementation domains represent the realized domains in my chart.

Class descriptions and attributes are listed in Appendix B.

---

<sup>3</sup> Java Introductory: Assignment 1 is related to Processing program, if you want to know about processing and related information, please contact [Henry.Gardner@anu.edu.au](mailto:Henry.Gardner@anu.edu.au)

# IMPLEMENTATION AND TESTING

## **Implementation**

As a part of SDLC, the implementation stage is to build the real code according to the analysis and design. In particular, an experienced programmer can write nice construction code basing on the job from analyzer and designer. As for me, I spent half time of doing the whole project. However, I learned a lot of new knowledge and techniques of writing a processing application with OpenGL.

According to xtUML theory, if there is a model compiler available, then it would translate the models into code automatically. However, if there is no model compiler, then the programmer must translate the model into code by following software architecture manually.

In this project, I will apply the OpenGL to construct 3D object within **Processing** environment. There are some pseudo codes of how to construct a robot arm by using OpenGL. Furthermore, used some plug-in library, such as loading the Ess plug-in as the API of how to invoke and play sound file as well as the proCONTROLL supports the relevant API for joystick. All of these coding in **Processing** will be based on Java language.

In addition, I could use the API such as loadSound(), getButton() and so on to search related sound file and external device then loading into the program.

## **Testing**

In this project, the testing stage would find out the bugs from implementation code basing on the execution test case. In the xtUML, the testing stage will work with a suitable tool available.

Test scenarios are listed in Appendix C.

The pseudo code of constructing robot arm

```
glBegin
//construct the upper arm
Translate (x, y, z)
Rotate (degrees, x, y, z)
Draw (upper_arm)

//construct the elbow
Translate (x, y, z)
Rotate (degrees, x, y, z)
Draw (elbow)

Translate (x, y, z)
Rotate (degrees, x, y, z)
Draw (lower arm)
glEnd

//construct the hand
PushMatrix()
  translate ()
  rotate ()
  draw (hand)
PopMatrix()
```

# CONCLUSION

## ◆ Unimplemented requirements (Appendix A)

1). How to divide one processing file into several runnable files. Just like the Java or C language.

## ◆ Run in Linux OS

**Processing** has the version of Linux. Then the application program will work well on the Linux platform. Furthermore, the exhibition from Linux platform is same as the Windows platform.

## ◆ Future extension

Using the joystick or joy pad to control the 3D object of robot arm is not totally intuitionistic for the users. The better way is to use the data glove such as the popular one that is P5 model. When user wears the data glove, he/she will better understand how to control the robot arm by using stuff wearing. There is no need some user guides, it's easy to operate and non-memorized.

## ◆ About **Processing** (Experience) and Future

Processing is easy-using, functional open-source software. It is similar to Java programming, but it is easier to create a 3D objects or animation than Java 3D. Furthermore, it supports some useful libraries for user to invoke related API to apply into source code. Most importantly, some high level programmer could write their own library to add into Processing as the build-in package by using Java language. It will be getting more powerful and integrated in future.

PS: Some system requirements could not finished in this project.

# APPENDICES

## Appendix A

Number Item	Requirements/Functionality	Priority	Eventually Implemented
1.1	When user start the application, he/she would see a window with a 3D objects of robot arm.	1	Y
1.2	The GUI is 800x600 window and there is no buttons and menubar on it.	1	Y
1.3	The sound files have been loaded into <b>Processing</b> .	1	Y
1.4	Searching peripheral device and loading into <b>Processing</b> environment.	1	Y
2.1	User shall have the related knowledge about Processing.	5	N (Not necessary to know about this software)
2.2	The program could search relevant sound file from the specific loading path.	3	Y
2.3	User can know whether the sound file loaded or not through the implementation of the application program.	4	N (it is hard for a normal user to know)
3.1	The program could search the related peripheral device and load into this program.	1	Y
3.2	The program shall remainder user to plug in the related device connecting to the computer	1	N
3.3	User could know the correct device to connect to the computer through the user guide.	5	Y
3.4	The program will apply the initialization of button and slider to the fit peripheral device.	1	Y
4.1	User can control the movement of the fingers and joints of robot arm by using keyboard.	1	Y
4.2	User can press one key from one joint motion of robot arm.	1	Y
4.3	User can press several keys at one time, and then the relevant joints would move together.	1	Y
5.1	User can control the movement of robot arm by using the peripheral device (joystick or joypad).	1	Y

5.2	User could press the buttons from joystick to move the fingers movement of robot arm. And then several buttons can be pressed at the same time.	1	Y
5.3	User could press the slider of joystick to rotate the joint of robot arm.	1	Y
5.4	The program offers a reset button from joystick; it would offer two different movement functions for robot arm. For example the same button finished the two movements: scratch and extend the hand.	1	Y (User shall know through user guide)
6.1	When user press key to activate a movement of a joint or a finger, then the relevant sound should be played and changed.	1	Y
6.2	When user press the slider or buttons of joystick to activate a movement of a joint or a finger, then the related sound should be played and move different part of arm, then sound changed.	1	Y

## Appendix B

### Class Attributes and Description

#### Sound

Inheritance: Interfaces from the library of Ess plug-in

Description: This plug-in software application was developed in Java.

Attributes	Data Type	Function
Channel	AudioChannel	Declare a reference of sound file
Channel50	AudioChannel	Declare a reference of sound file
Methods	Return Value	Function
loadSound()	Class	Load the sound file into program

#### 3D Model

Inheritance: Java APIs from Processing

Description: this 3D object would be created within Processing that is developed by Java.

Attributes	Data Type	Function
shoulder , elbow, elbow1	Float	The variable for the joints of the upper arm and the lower arm
thumb, FirstF, MiddleF, SmallF	Float	The variables for the upper fingers
P_thumb, P_first, P_middle, P_small	Float	The variables for the lower fingers
Turn, turn1, turnZ	Float	The rotate rate for rotating the whole robot arm.
Keys	Boolean[]	Keys from keyboard are for controlling the movement of robot arm.
FontA	PFont	Some statements are shown on the scene.

<b>Methods</b>	<b>Return Value</b>	<b>Function</b>
Setup()	Void	Set up the size, basic exhibition for the screen shown after program runs
keysInit()	Void	Initialize the keys for the purpose that several keys can be pressed same time.
Draw()	Void	Be similar with the OpenGL function 'display()', displaying the all of components.
textShow()	Void	The text shown on screen
displayArm()	Void	Using openGL to construct the robot arm including the every joints and fingers.
drawQuads()	void	draw the equipment for protecting upper arm
drawFingerTip()	Void	the vertex collection of finger tip
boolKeys()	Void	press two or more keys at the same time to control movement of robot arm
keyPressed()	Void	The key events from the keyboard
keyReleased()	Void	keys release call back

## Device

Inheritance: Interface from the proCONTROLL plug-in

Description:

<b>Attributes</b>	<b>Data Type</b>	<b>Function</b>
controll	ControllIO	As a reference to get the instance of device IO
device	ControllDevice	Referring to control, get the device's name or number.
sliderX, sliderY	ControllSlider	Reference for the silder x and y from the joystick or joypad.
button,button1, button2,button3,button4, button5,button6,button7,button8, button9	ControllButton	Reference for the buttons from the peripheral device.
<b>Methods</b>	<b>Return Value</b>	<b>Function</b>
controllerInit()	Void	Initialization of the control device including references
Toggle()	Void	Changes for the reset button
buttonInit()	Void	Refer to the new reference of control device – getButton()
buttonClicked()	Void	Set the buttons changes plus the effects when the reset button clicked

## Appendix C

### Test Scenarios

Number Item	Event (external stimuli)	Action (System response)	Condition (state required)	Is it Successful?
1.1	Double click the Processing software.	1. application starts 2. Main GUI displayed	In the Windows Platform	Y
1.2	Click “Close” cross at the top right of GUI window	1. Program exits 2. return to environment of Processing	Main GUI in the active state	Y
2.1	Searching the sound file	1. From the correct path 2. There is no path error or no such file	Main GUI in the active state	Y
2.2	Loading the sound file	1. Using loadSound function to load the pointed sound file. 2. Return to main window	1. Main GUI in the active state 2. Loaded sound into sceneGraph	Y
3.1	Searching the available peripheral device	1. Making sure device has been connected to PC 2. No such device error or not	Main GUI in the active state	Y
3.2	Getting the available peripheral device	1. Using get device function to get the control of peripheral device 2. Return to main window	1. Main GUI in the active state 2. Loaded model into sceneGraph	Y

4.1	Click some specific keys to rotate those joints of Robot arm	<ol style="list-style-type: none"> <li>1. The joints related to keys will be rotated.</li> <li>2. clockwise or anti-clockwise movement.</li> </ol>	Main GUI in the active state	Y
4.2	Click several keys at the same time	<ol style="list-style-type: none"> <li>1. several joints will be moving at the same time.</li> <li>2. Stopping movement as long as key released</li> </ol>	Main GUI in the active state	Y
5.1	Click slider of joystick to rotate robot arm	<ol style="list-style-type: none"> <li>1. load the device model</li> <li>2. Rotating the robot arm</li> <li>3. Movement by clockwise or anti-clockwise</li> <li>4. load the sound model</li> <li>5. The sound file is playing</li> </ol>	1. Main GUI in the active state	Y
5.2	Click buttons to move the finger	<ol style="list-style-type: none"> <li>1. Make sure working on the device model</li> <li>2. Do the movement by extending the finger</li> <li>3. the sound file is playing</li> </ol>	Main GUI in the active state	Y
5.3	Click the reset button	<ol style="list-style-type: none"> <li>1. Working on the 'reset' model</li> <li>2. Do the movement by scratching the fingers</li> </ol>	Main GUI in the active state	Y
5.4	Click several buttons at the same time	<ol style="list-style-type: none"> <li>1. Several fingers will move at the same time.</li> </ol>	Main GUI in the active state	Y

		2. The sound file is playing.		