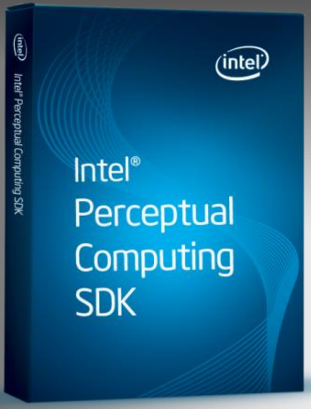
香港中文大學
The Chinese University of Hong Kong

Perceptual Application Development using Intel Perceptual Computing SDK


intel
Intel® Perceptual Computing SDK
Intel® Perceptual Computing SDK

FYP LYU1302

Supervised by Prof. LYU Rung Tsong Michael

Department of Computer Science and Engineering

CUHK
1155026038 HONG Shi Qi

Abstract

In this final year project, we are going to utilize Intel Perceptual Computing SDK and correspondent Creative* camera to developed an application of virtual string lion puppet control. Our study objective is to virtualizes behaviors in the real world to computing world through string puppet simulation. We did researches on the market and not similar project have been done before, thus this is a new idea worth to implement. At the end of last semester, we have built a rough string lion puppet prototype and realized hand movement and gesture control on basic puppet transformation. In this semester, we further developed this virtual lion puppet. Till now, we have reached following stages---1) Physics engine introduction and realistic move pattern construction; 2) A certain degree algorithm optimization and SDK inherent restriction reduction; 3) Lion puppet model rendering using modeling software. High similarity model construction; 4) Real stage scene and control scenario establishment.

After a-whole-year FYP study, we enriched our knowledge not only in perceptual computing but also in many other different areas including software developing, computer vision and virtual reality.

`

Contents

Chapter1: Introduction

Background

At present, high and innovative technology development is soaring all over the world. Creative technology concepts emerge endlessly, new electronic equipment spring up like a mushroom. Electronic generations switch all the time. As consequence of novelties explosion, a bunch of traditional technologies have been eliminated in market because they can not meet people's demands on technology products. Among them, human-machine interactive development is of great popularity in technology battlefield.

Traditional interactive between human beings and machine is based on keyboard and mouse control. However, this simple method is far under people's expectation. The widespread of touch screen in mobile terminal is a typical instance of this tendency. Instead of use indirect keyboards control, people are more inclined to direct and more user-friendly control method like touch screen.

Figure1: Touch-Screen products overview.

Perceptual Computing was first introduced in 2012 by Intel Corporation, which is for research on natural user-machine interaction, that is, using human's gesture, speech and face expression as input to control computer or other devices. Before that, similar concept has already been brought into reality, such as Xbox games with Kinect from Microsoft Corporation.

Perceptual computing is a brand-new and promising field. We can tell its brightness future from nowadays tendency. Perceptual computing technology will certainly be widely applied in future's high-tech design and household electronic production, and play a significant and indispensable role in human's life.

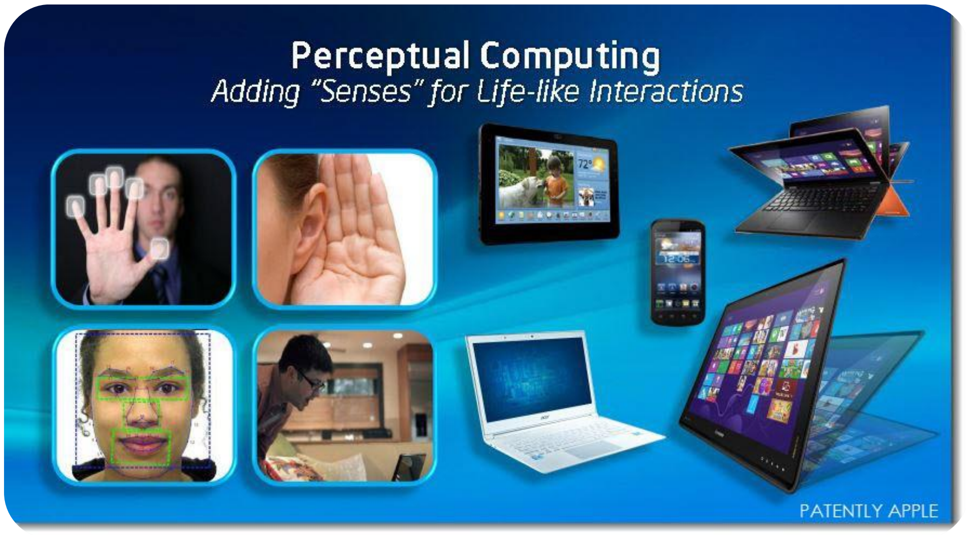


Figure2: Perceptual Computing Idea

Motivation

Lion dance is a traditional Chinese performance. People can see it in festival parties or opening ceremonies. Marvelous dancing skills and beautiful lion suits attract lots of lion dance fans. Lion string puppet or marionette then came into being as expected. String puppets' movements rely on performer's control over several strings attached to puppet body, similar to well-known puppet Pinocchio. In traditional Chinese festivals, kids love to play with it by creating fun puppet shows to share their happiness with others. In reality, puppet performer controls a wooden cross who linked to all strings to move puppet. Once strings on the cross being pulled, with different strength and angles, puppet can make different movements.

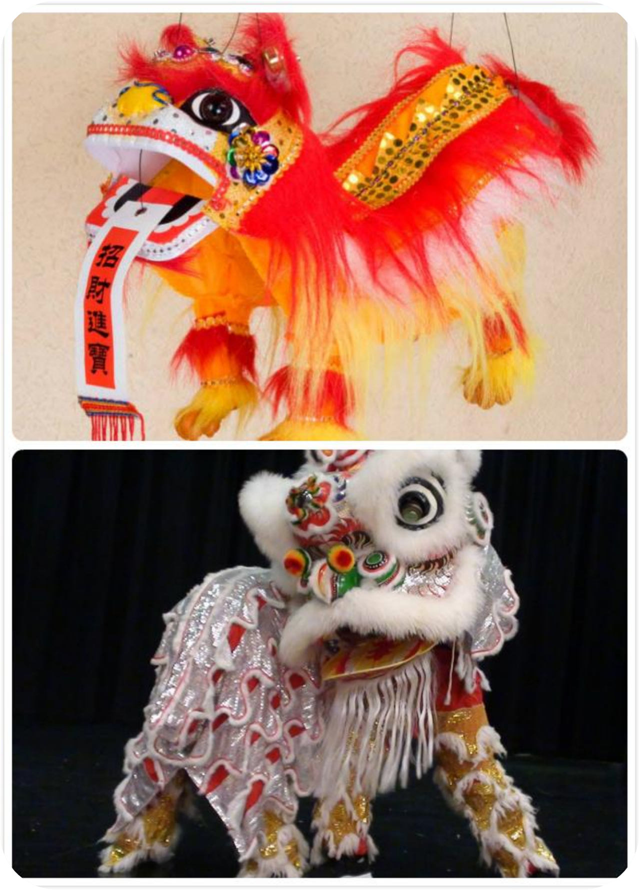


Figure3: String lion puppet vs. lion dance

Thus, the control method of puppet is hand control, which leads to a recent popular topic, gesture recognition. Gesture Recognition is trying to interpret human gestures via mathematical methods. Thus enable gesture recognition control in computing world. Our group is trying to connect puppet movement to gesture recognition method which means to interpret gesture to meaningful puppet move. What we do is to find a way to eliminate physical strings but only trace hand or finger movements to realize puppet control in virtual world.

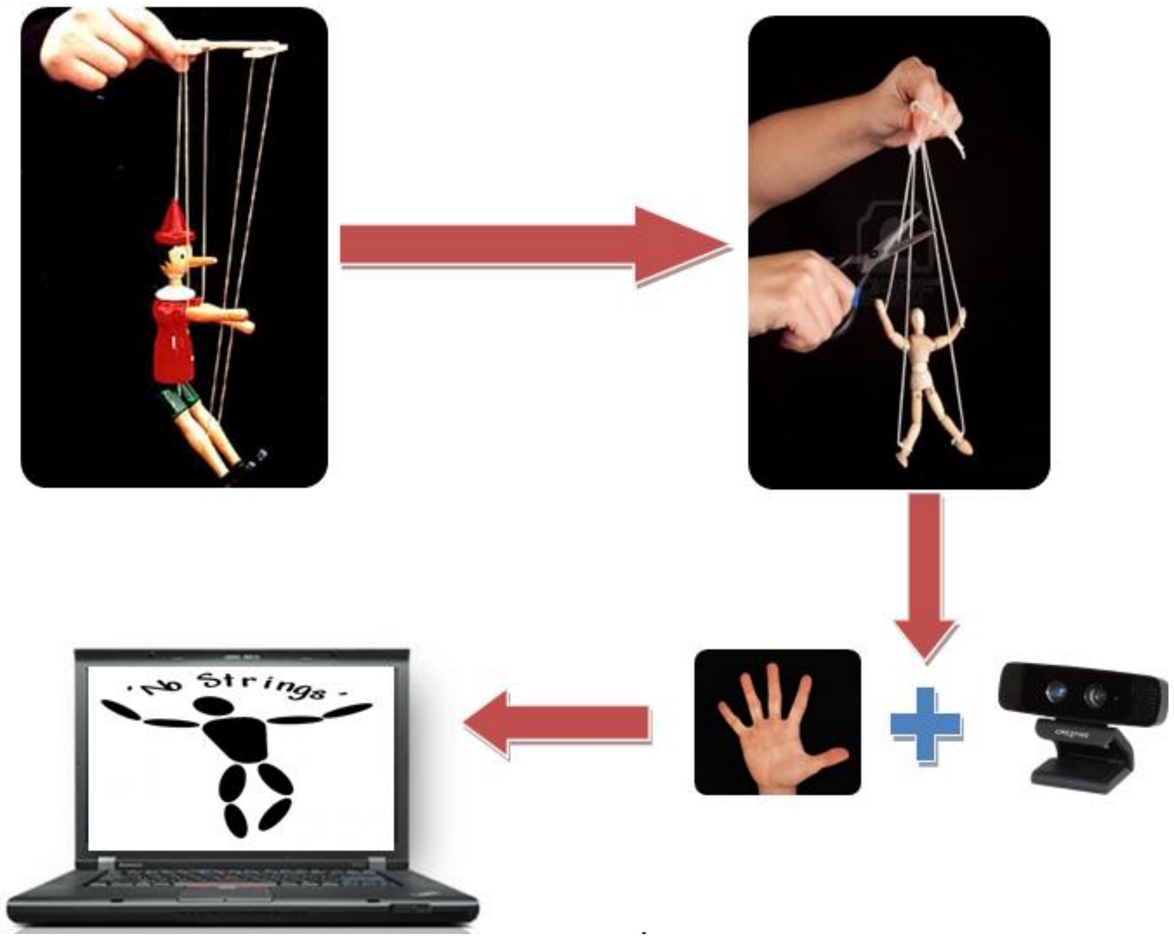


Figure4: Idea illustration

We decided to apply Intel Perceptual Computing into a visual lion puppet computing program, by which to deepen our understanding about perceptual computing, improve our capacities of software development and enhance our ability of perceptual computing programming.

Objective

1. Learn to virtualize real objectives.

In this final year project, we are going to do research on how to map people's action to virtual model's behavior on computer. We plan to reach this objective by writing an application to retrieve human's hand movement and simulate correspondent string lion puppet control process in reality. In this whole study process, we need to find out solutions of how to collect raw data from input devices, how to analyze and process the raw data to make them perceptual intuitive and utilizable. Then we need to design algorithm to map the real lion puppet onto our visualized model, that is, define the mapping points, mapping functions, Simulate real physical environment.

Perceptual computing is a newly and promising technology, it will certainly open a vast vistas. We are going to try our best to understand the deep meaning of perceptual computing during this FYP process.

2. Surmount SDK's limitations.

Due to the facts that Intel Perceptual Computing SDK is still a young SDK related to Nature User Interaction topic, some of its functionalities are lack of high accuracy. Intel Perceptual Computing SDK is not a perfect SDK without any limitations. Although this SDK is keep updating frequently, still, when doing our programming, its performance can not always reach our expectation. The typical problems we encountered during programming include finger tracking lost, indistinguishable on hand sides (can not distinguish left and right hand correctly) and several small but hard to be solved problems.

It is unpractical for us to modify develop library packets code, so we have to use information and data we possessed now to design suitable model and algorithms to avoid technical limitation.

3. Learn to do software development

This final year project contains a whole software development process, that is, general design (system architecture, module division, function allocation, interface design), detail design (algorithm, data structure, hierarchical structure and invoking relationship, exception handling), coding and testing etc. During this whole final year project, we will stand on software developer’s side thinking problem, experience software development steps and better our software development ability.

Project Overview

1. Topic Overview

Our project target is to develop application based on Intel Perceptual Computing SDK. To develop a complete application, we choose game engine Unity3D as our main software developing tool. The application we made is a virtual string lion puppet controlling software. In the application scene, there are string lion puppet model and puppet stage model which are made based real lion puppet and stages. User could use hand to go through the whole application process including application launch, quit and puppet behavior's control. The hand movement and gesture detection part is realized based on Intel Perceptual Computing SDK. Control method is almost the same as puppet control in real life.

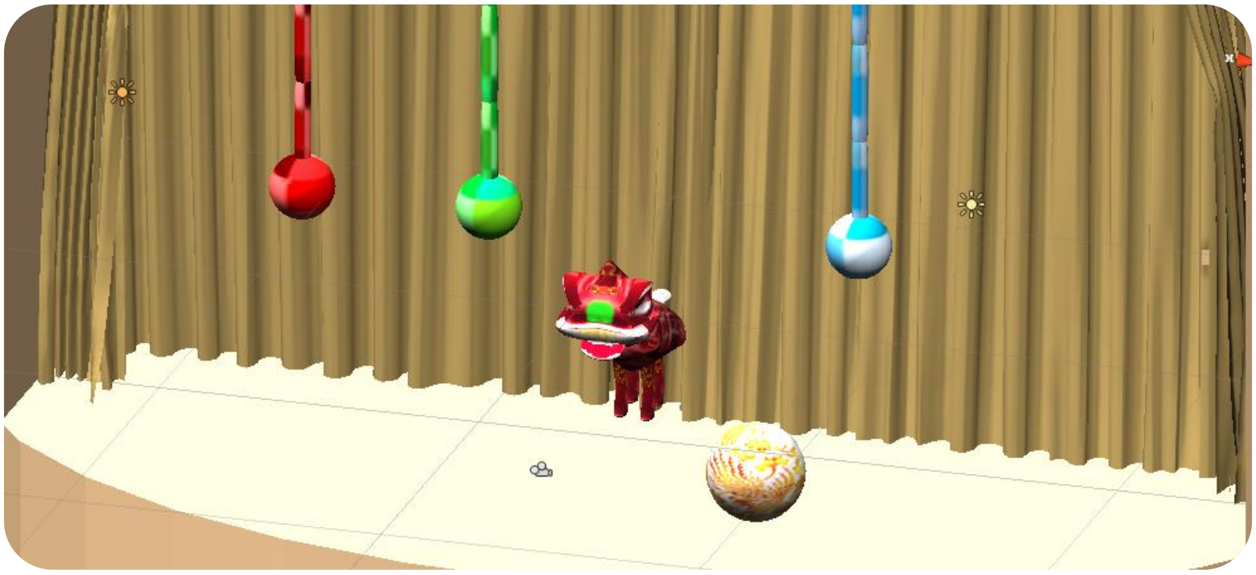


Figure5: Application Scene Overview

Key Features

In our string puppet control application, this lion puppet was put in a highly realistic physics environment. When we use hand to 'drag' those virtual and invisible strings which attached to the puppet, besides the 'force' our hands provided, the lion puppet would also be affected by gravity and inertial force. Moreover, when hit the hanging balls on the stages, those balls would swing as expected.

Lion puppet mouse movements is triggered by specific gesture. This movement is an extra attached animation clip made in Blender.

Lion legs movement is a natural reaction. The movement is not directly generated because user hand rotation or transforming like what we did last semester. On the contrary, we just attach them on the body. All the swing and transform are all generated because physics force, which would make the movement become very natural and similar to real situation.

Background music and special event audio also be added in the application.

Hand control is enough for this whole application, keyboard and mouse is not necessary during the execution.

Working flow is as follows

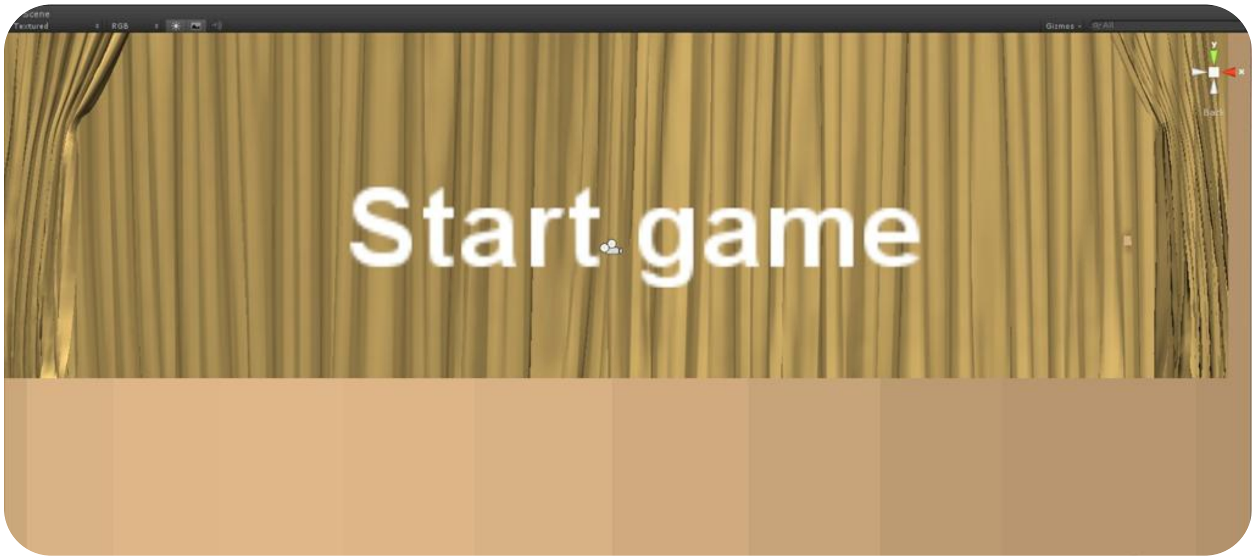


Figure6: Lauching Interface

Use gesture 'PEACE' to start;

Use gesture 'THUMB_DOWN' to quit

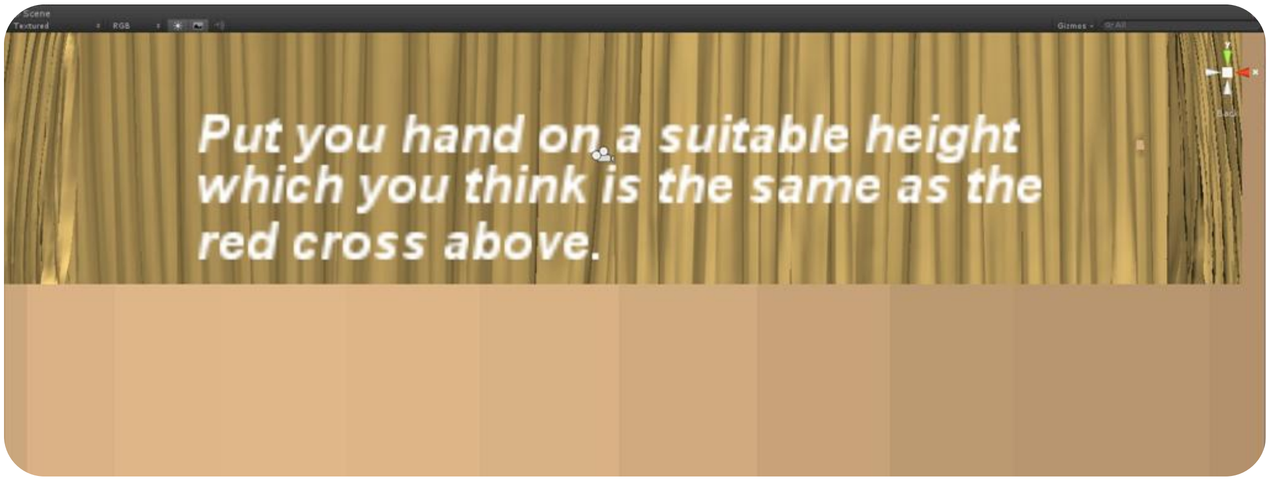

Put you hand on a suitable height which you think is the same as the red cross above.

Figure7: The second Interface

Use gesture 'PEACE' to proceed when suitable position is decided

Use gesture 'THUMB_DOWN' to quit

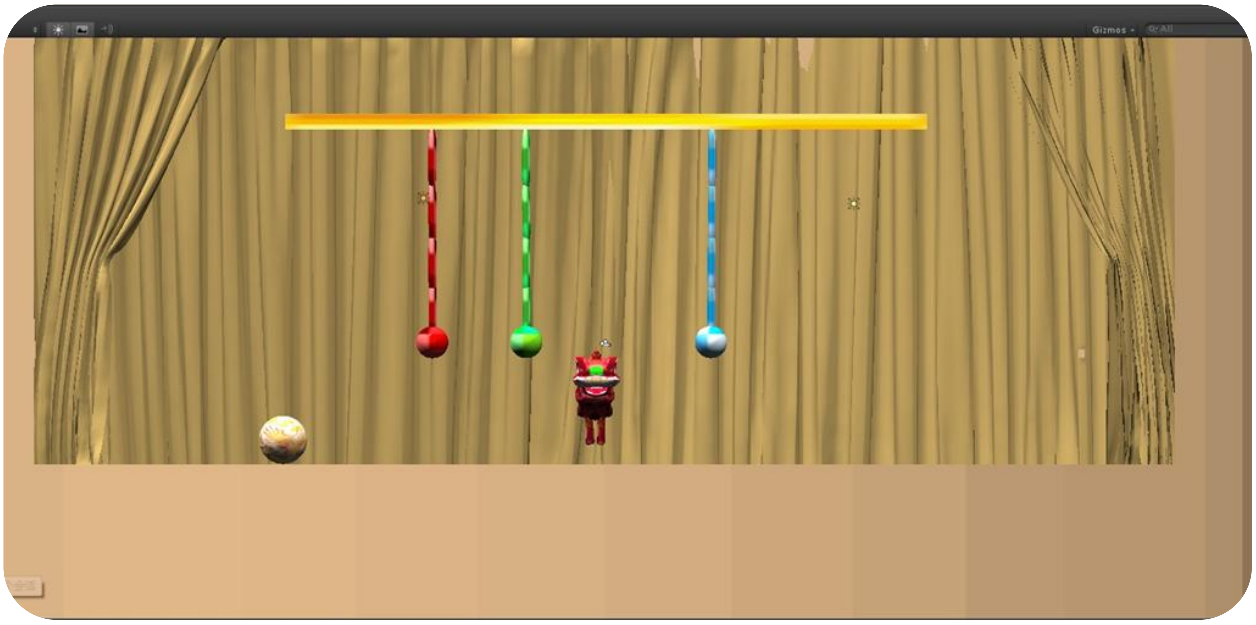
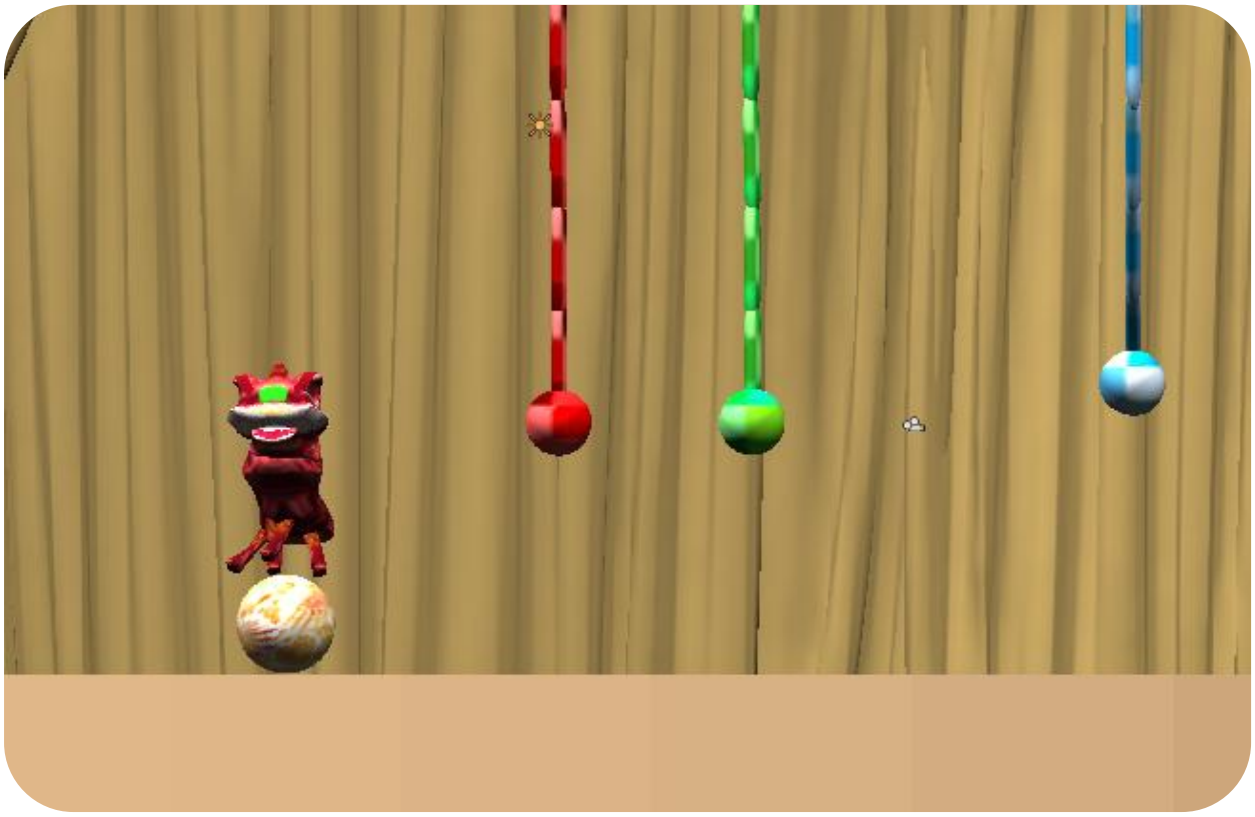
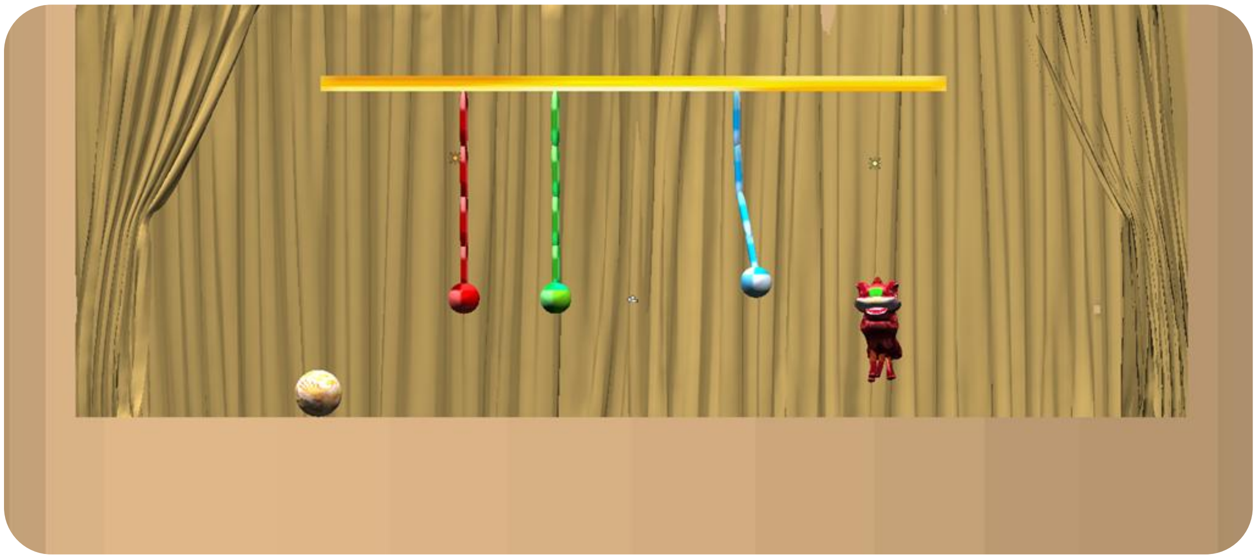


Figure8: Main initial stage scene

Just move your hand freely and the lion puppet will follow your move





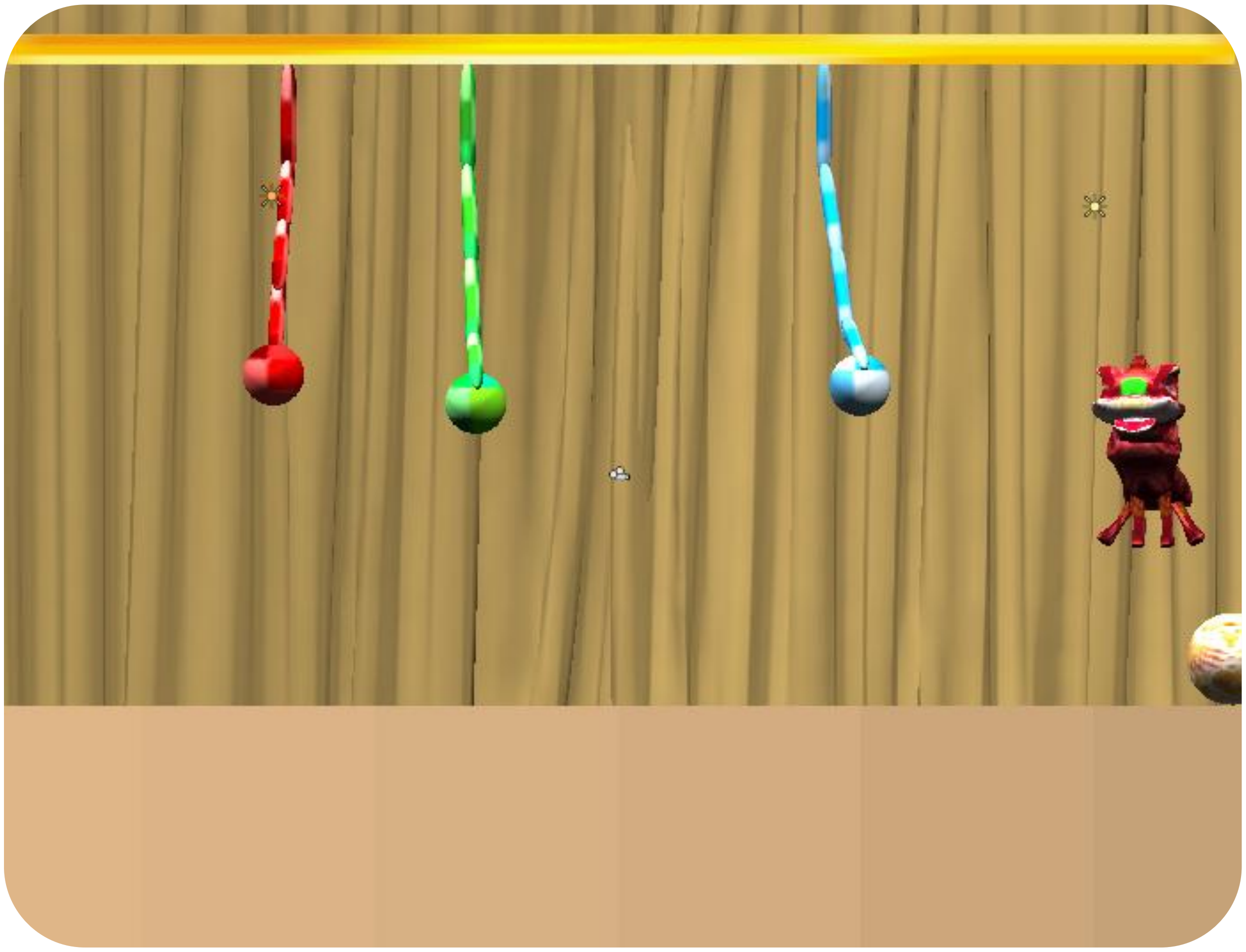


Figure9: Simple movement demo

2. System overview

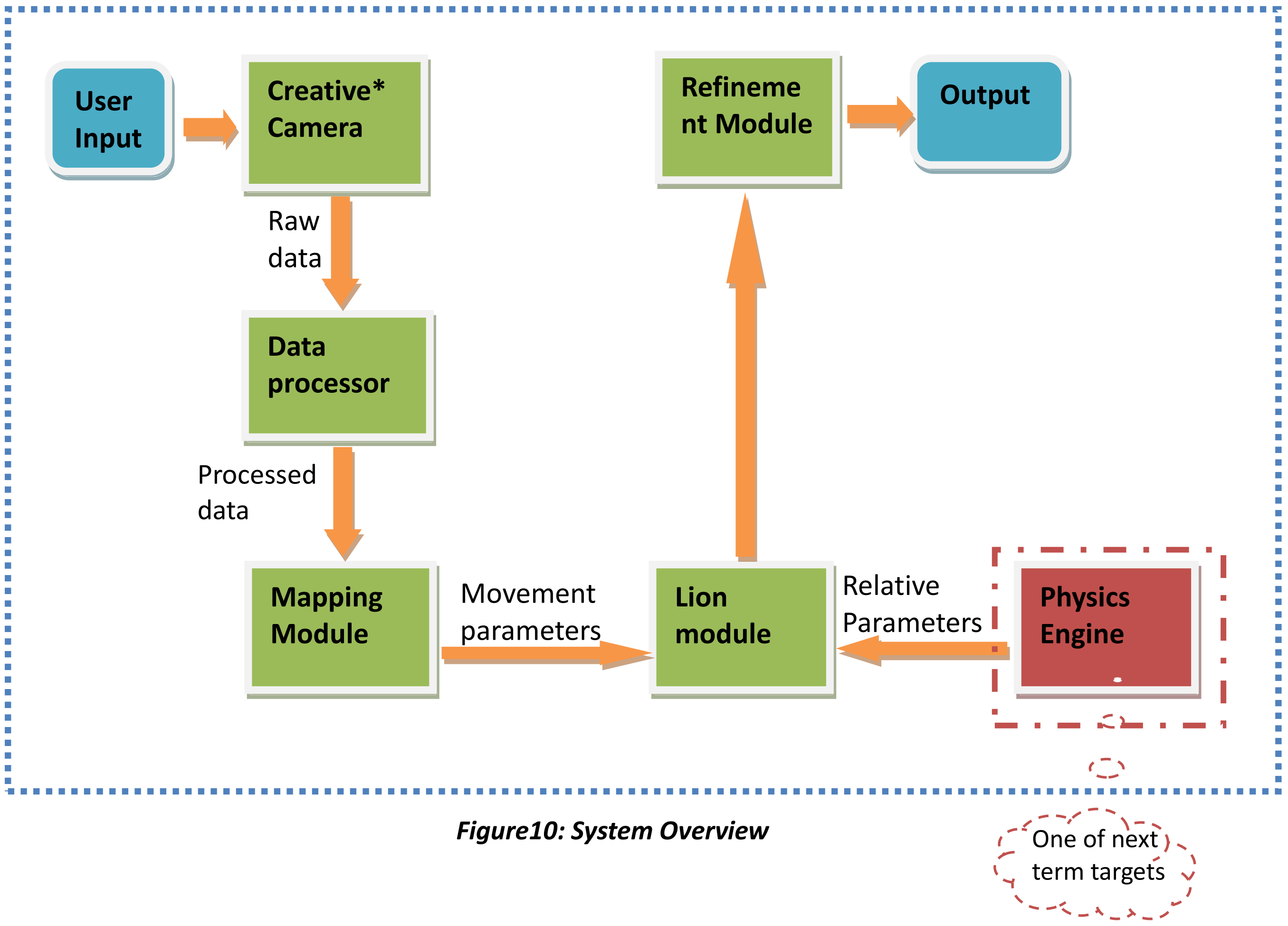


Figure10: System Overview

3. Milestones & schedule

<u>Milestone</u>

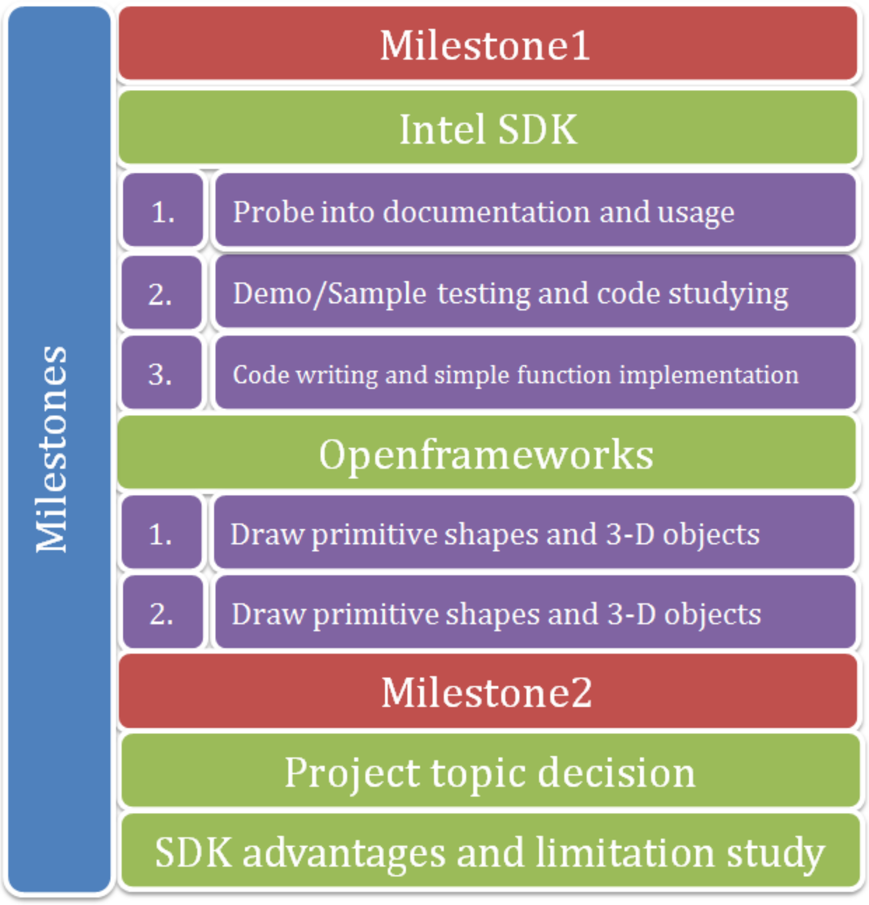
Last Semester:

Milestone1 is about information collecting and development tools understanding.

In milestone1, we have done some basic research and study on Intel Perceptual Computing, including probe into camera capabilities, parameter documentations and application usage. Besides that, we also ran several demo case offered by Intel to find out SDK's functionalities and limitations. Then we followed SDK documentation to write some simple programs to be familiar with kinds of API for future utilization.

Because we chose openFrameworks as model building and rendering platform, thus we also do some research on openFrameworks usage. We try to use openFrameworks to draw some primitive shapes and 3-D objects, and made several animation implementation.

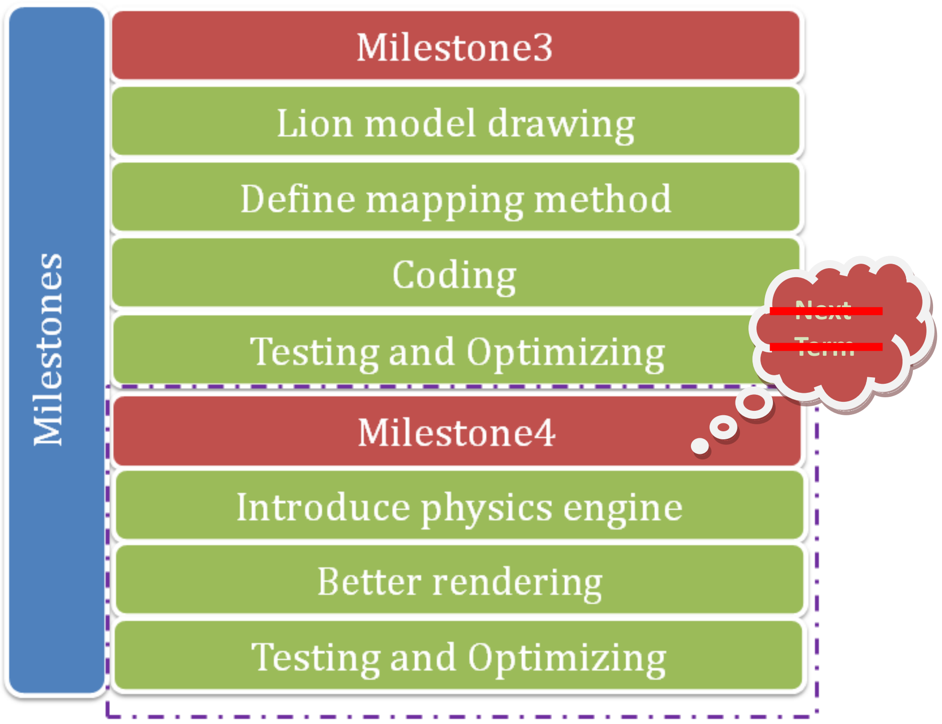
After we are familiar with Intel Perceptual Computing


Milestones
Milestone1
Intel SDK
1. Probe into documentation and usage
2. Demo/Sample testing and code studying
3. Code writing and simple function implementation
Openframeworks
1. Draw primitive shapes and 3-D objects
2. Draw primitive shapes and 3-D objects
Milestone2
Project topic decision
SDK advantages and limitation study

SDK and openFrameworks, we got them combined. At the first combination process, we just showed several points detected by camera in correspondent position. Then we also did some gesture recognition and test some relative animation and transformation.

Milestone2 is about application topic decision.

In milestone2, we have come up with many ideas about how to design and implement our project, including gesture-controlled eBook reader, gesture-controlled media player, gesture-controlled aircraft battle game, visualized piano-playing program and visualized lion puppet control program. We did some research and tried some test on these ideas before we finally made up our mind. During these trials and tests, we deepened our understanding on Intel Perceptual Computing's advantages and limitations. It is precisely because of these understandings that made us finally decide to develop an application



Milestone3 is about application implementation.

In milestone3, at first period, we draw a non-detailed lion in openFrameworks and set up a protocol to map hand data to lion's body parts or lion's behaviors. Then we started coding to implement our design. After we finished primary coding work, we did several tests on accuracy and stability. Based on some unexpected bugs and deficiency, we modified the protocol, revised algorithm and did more testing on it. In the end of this milestone, namely, end of this semester, the basic functionalities of our program are realized.

This Semester

In milestone4, which is the main work we did this semester, we utilized professional modeling tool Blender and built a much more realistic string lion puppet model. Besides, we made several animation clips can be triggered by gesture. After string lion puppet modeling, we introduced realistic physics environment in Unity3D. When we imported the prepared model into Unity3D working environment, we can still use gesture to control it as what we did to prototype last semester. Also, the gravity and inertial force would have impact on it. Collisions would also happen if the lion model hit the stage objects in the scene. At this moment, our string lion puppet behaviors exactly like a string lion puppet in the real world. Another thing is that to plant our program onto Unity3D platform, we rewrote our project codes in C# for scripts language support purpose. Now the whole program is more meaningful and interesting.

Schedule

Since FYP project is a whole-year project. We have different emphasize points and aspects on the first semester and the second semester.

The first task is mainly about 'functional'. It means that on the first term, we are focused on algorithm study. We need to find a suitable function for data analyzing and correspondent behaviors design. The first step, which is the milestone1 part, is to be familiar with basic SDK methods and run in the camera to have a clear clue that what we can achieve and what is limitations due to inherent defects. After that, in the milestone 2 and milestone 3 we tried to find out mapping method and suitable algorithm to implement the string puppet control function. We did that from late October last year and implemented the final version at the end of last semester. During demo, we used software Openframeworks to build a simple lion puppet model which is just a combination of primitive cubes.

Our focus on the second term can be summarized into one word 'better'. What we have done last semester is a prototype building work. At this stage, functional problem almost solved except for some small imperfect error due to inherent limitation or can be fixed with parameter tuning. So in the second term, as we mentioned in last semester's report, we want to add physics engine on our project to make the string puppet works like a real puppet. So the first thing we need to decide this semester is to use which game engine. After cross check and comparison, we find unity3D is best for our project. Synthesis Intel Perceptual computing SDK on Unity have the best performance. After game engine have been decided, we begin to transfer our codes to unity platform. As we mentioned on Unity specification part, we can see that the scripting language used in Unity do not include C++. Thus our first challenge is to transfer our C++ codes in C# version when C# is exactly one of unity scripting language. When my partner working on the part, I begin to do another job, 3D model generating. Our string lion puppet prototype shown last semester is merely 2D primitive combination. To add physics engine on puppet model, we need to made a 3D model first of all. Besides 3d modeling, animation generating, we also made a good-looking scene and added audio effects. Also, made optimization on Intel Perceptual Computing SDK limitations.

Schedule Tables

2014 / 01

Sun	Mon	Tue	Wed	Thu	Fri	Sat
			1	2	3	4
5	🟩 6	🟩 7	🟩 8	🟩 9	🟩 10	🟩 11
🟩 12	🟨 13	🟨 14	🟨 15	🟨 16	🟨 17	🟨 18
🟨 19	🟨 20	🟨 21	🟨 22	🟨 23	🟨 24	🟨 25
🟨 26	🟨 27	🟨 28	🟨 29	🟨 30		

🟩	Last semester work summery and this semester work plan
🟨	Physics engine study, modeling tools selection, sample testing

2014 / 02

Sum	Mon	Tue	Wed	Thu	Fri	Sat
						1 🟨
2 🟨	3 🟦	4 🟦	5 🟦	6 🟦	7 🟦	8 🟦
9 🟦	10 🟦	11 🟦	12 🟦	13 🟦	14 🟦	15 🟦
16 🟦	17 🟪	18 🟪	19 🟪	20 🟪	21 🟪	22 🟪
23 🟪	24 🟪	25 🟪	26 🟪	27 🟪	28 🟪	

🟦 Code rewriting and modification(for platform planting purpose.)

🟪 Physics element adding, parameter adjusting

2014 / 03

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1 🟥
2 🟥	3 🟥	4 🟥	5 🟥	6 🟥	7 🟥	8 🟥
9 🟥	10 🟥	11 🟥	12 🟥	13 🟥	14 🟥	15 🟥
16 🟥	17 🟥🟩	18 🟥🟩	19 🟥🟩	20 🟥🟩	21 🟥🟩	22 🟥🟩
23 🟩	24 🟩	25 🟩	26 🟩	27 🟩	28 🟩	29 🟩
30 🟩	31					

🟥 Code rewriting and modification(for platform planting purpose.

🟩 Correspondent 3D modeling, animation taking

2014 / 04

Sum	Mon	Tue	Wed	Thu	Fri	Sat
		🟥 1	🟥 2	🟥 3	🟥 4	🟥 5
🟥 6	🟥 7	🟥 8	🟥 9	🟥 10	🟥 11	🟥 12
🟥 13	🟥 14	15	16	17	18	19
🟥 20	🟥 21	22	23	24	25	26
27	28	29	30			

🟥 Final modification, typo and error solving, scene beatifying.

Chapter2: technical support

Development environment

The main project codes was first written in C++ on Microsoft Visual Studio 2010 version 10.0.30139.1 in Chinese Version in Windows 7. When we using game engine Unity3D Version 3.5.1f2 (e9280fee30d7) to further develop our application, because of Unity scripts language limitation, we transferred our codes into C# version. Modeling rendering, texturing, binding and animating is realized in Blender Version 2.69 and GIMP Version 2.8.10.

Intel Perceptual Computing SDK version Now is 1.8.13842.0.

Figure 11: VS2010, Intel Perceptual Computing, openFrameworks, unity

Software tools

1. Intel Perceptual Computing SDK

Introduction

Perceptual Computing (PerC) is an organization within Intel Corporation tasked with the research, development, and productization of technologies for Natural User Interaction [1]. Natural User Interaction is a concept refer to user interface as effectively invisible, or becomes invisible with successive learned interactions while refer to user as interact is based on nature or natural elements [2].

Intel is intended to make a fundamental change on the way people interact with PCs using Perceptual Computing technology. Intel wants to realize communication between human and computers have intuitive, natural and engaging properties.

Intel Perceptual Computing SDK is a develop tool for developers to create new applications related to natural user interaction by take advantage of core abilities of capabilities: speech recognition, close-range hand and finger tracking, face analysis, augmented reality, and background subtraction.

Specification [2]

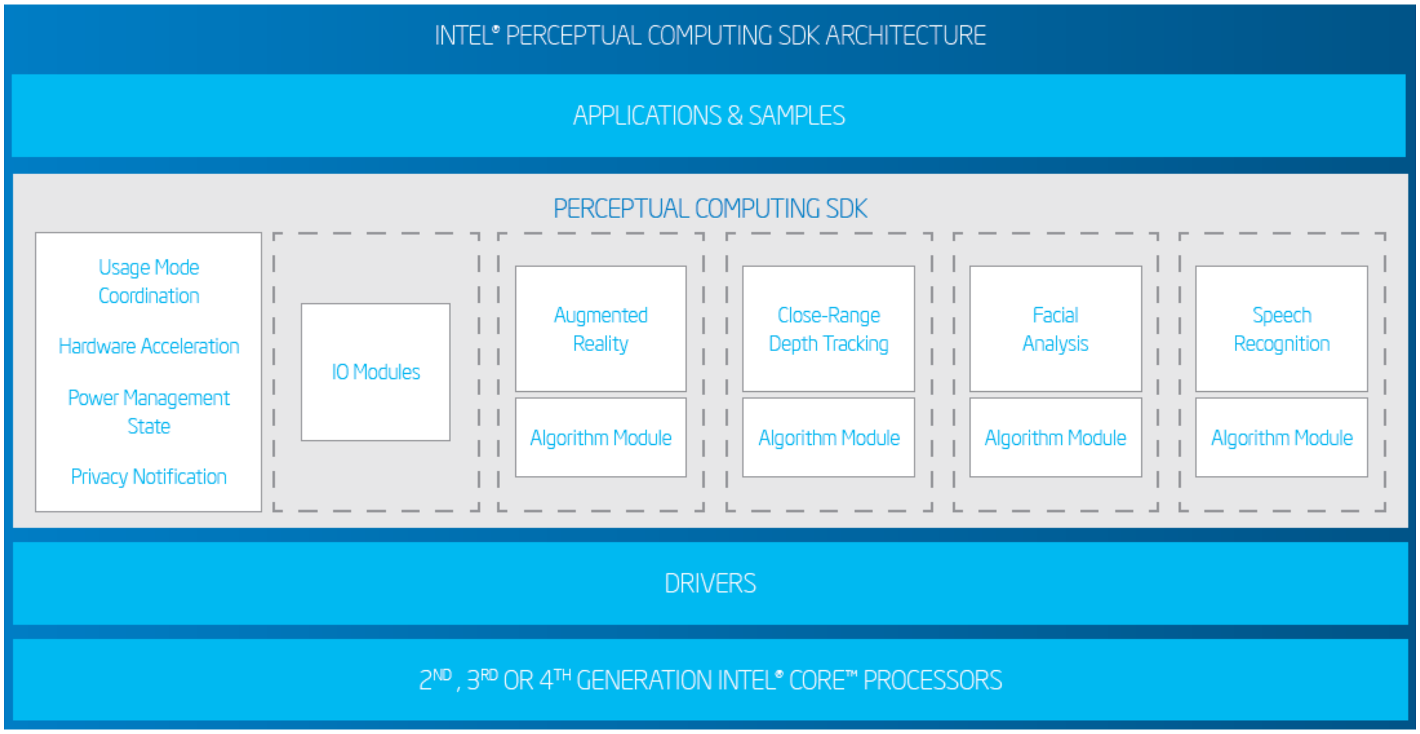


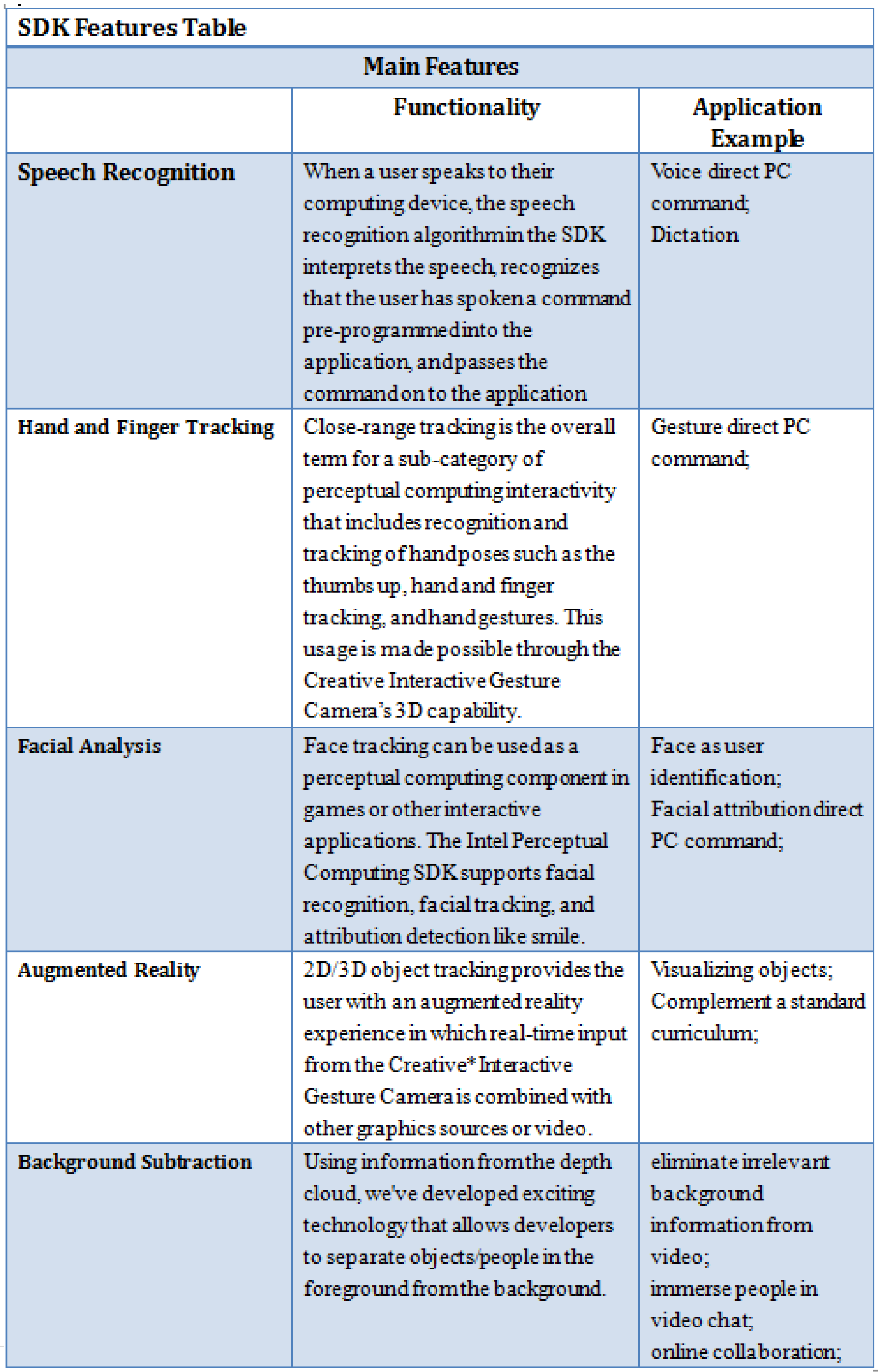
Figure12: Intel Perceptual Computing Architecture

The Intel Perceptual Computing SDK is composed of several layers and components (Layer and component details are stated on diagram above.) Among these modules, I/O and algorithm component are the interface user can access.

The I/O component takes charge of data collection. It is the data source of whole SDK. I/O component works as a data sink; its main duties include retrieving data from input device such as a CREATIVE camera and providing data to output device such as a monitor.

The algorithm component is the core functional unit. It implement the pattern detection and recognition algorithms that are critical ingredients of innovative computer experience, such as facial analyzing, gesture recognition, voice recognition, and text to speech or speech to text.

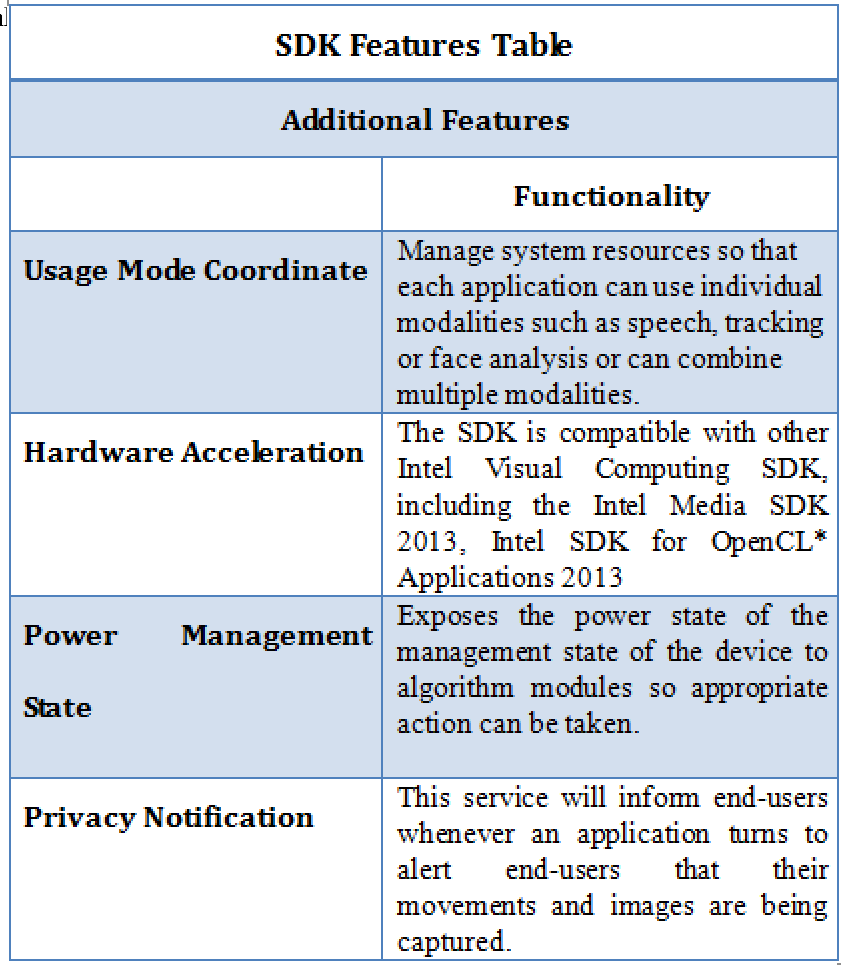
By accessing these two modules, developer can focus on the underlying implementation.



SDK Features Table

	Main Features	
	Functionality	**Application Example**
Speech Recognition	When a user speaks to their computing device, the speech recognition algorithm in the SDK interprets the speech, recognizes that the user has spoken a command pre-programmed into the application, and passes the command on to the application	Voice direct PC command; Dictation
Hand and Finger Tracking	Close-range tracking is the overall term for a sub-category of perceptual computing interactivity that includes recognition and tracking of hand poses such as the thumbs up, hand and finger tracking, and hand gestures. This usage is made possible through the Creative Interactive Gesture Camera's 3D capability.	Gesture direct PC command;
Facial Analysis	Face tracking can be used as a perceptual computing component in games or other interactive applications. The Intel Perceptual Computing SDK supports facial recognition, facial tracking, and attribution detection like smile.	Face as user identification; Facial attribution direct PC command;
Augmented Reality	2D/3D object tracking provides the user with an augmented reality experience in which real-time input from the Creative* Interactive Gesture Camera is combined with other graphics sources or video.	Visualizing objects; Complement a standard curriculum;
Background Subtraction	Using information from the depth cloud, we've developed exciting technology that allows developers to separate objects/people in the foreground from the background.	eliminate irrelevant background information from video; immerse people in video chat; online collaboration;

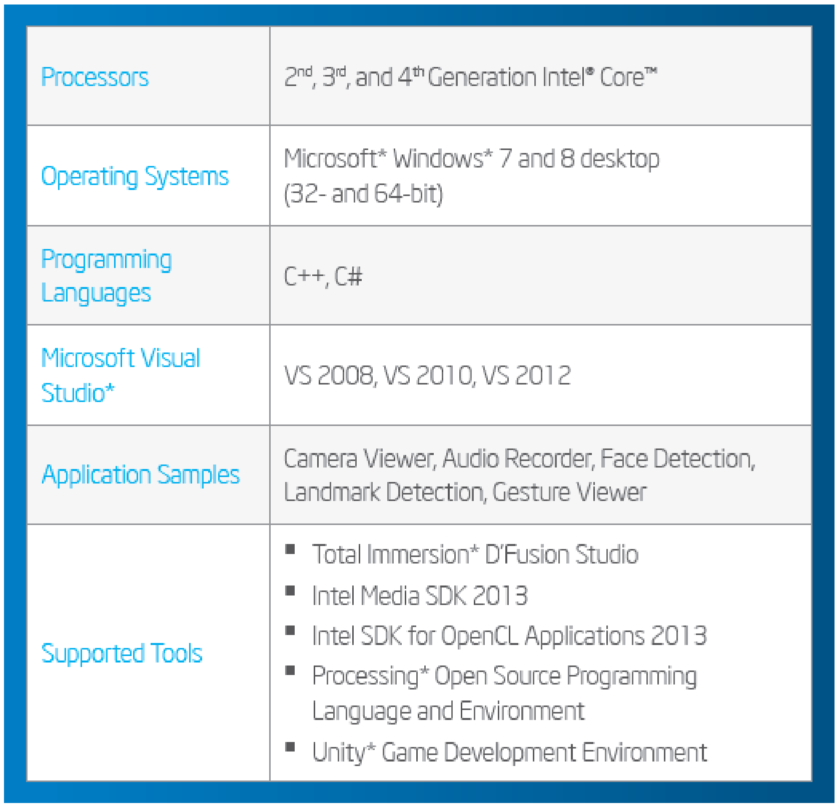
Table1: SDK Main Feature Table



SDK Features Table	
Additional Features	
	Functionality
Usage Mode Coordinate	Manage system resources so that each application can use individual modalities such as speech, tracking or face analysis or can combine multiple modalities.
Hardware Acceleration	The SDK is compatible with other Intel Visual Computing SDK, including the Intel Media SDK 2013, Intel SDK for OpenCL* Applications 2013
Power Management State	Exposes the power state of the management state of the device to algorithm modules so appropriate action can be taken.
Privacy Notification	This service will inform end-users whenever an application turns to alert end-users that their movements and images are being captured.

Table2: SDK Additional Feature Table

Supported Processors, Software, Samples, and Tools



Processors	2nd, 3rd, and 4th Generation Intel® Core™
Operating Systems	Microsoft* Windows* 7 and 8 desktop (32- and 64-bit)
Programming Languages	C++, C#
Microsoft Visual Studio*	VS 2008, VS 2010, VS 2012
Application Samples	Camera Viewer, Audio Recorder, Face Detection, Landmark Detection, Gesture Viewer
Supported Tools	▪ Total Immersion* D'Fusion Studio ▪ Intel Media SDK 2013 ▪ Intel SDK for OpenCL Applications 2013 ▪ Processing* Open Source Programming Language and Environment ▪ Unity* Game Development Environment

Table3: SDK Supported Information Table

Study Log

We started our final project at the end of August. Because we do not have too much knowledge in human-machine interaction field, at the project beginning, we try to have a whole understanding on Intel Perceptual Computing. Thus we read those documentation files offered by Intel repeatedly. Besides reading written material, we also tried may demo or samples offered by Intel, such as 'Gesture Viewer', 'Face Recognition', 'voice recognition'. We found that the recognition ability of Intel Perceptual Computing SDK is based several points, namely, the SDK will trace several geometric nodes of objects and process traced data by internal algorithm to realize gesture recognition and facial analysis. We have also test speech recognition and synthesis capability and found out it has limitation and not accurate enough, thus we did not decide to go voice direction.

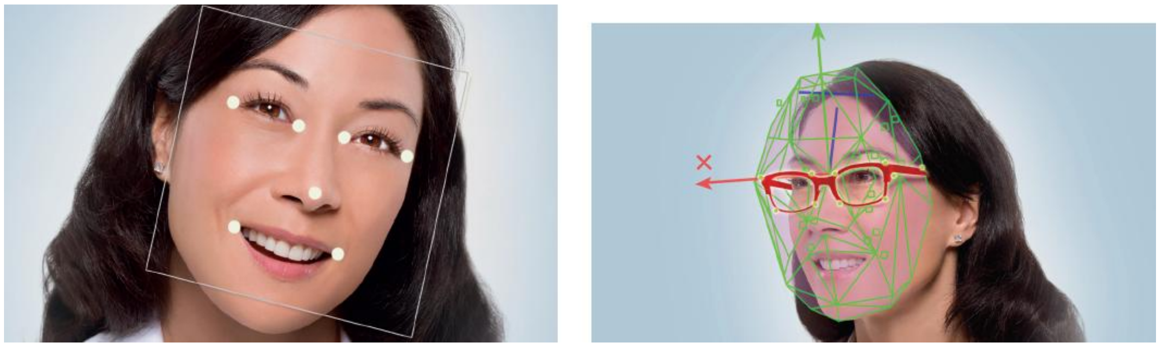


Figure13: Facial Analysis

As I mentioned above, because the fact that Intel Perceptual Computing SDK is a young product and just came into market, thus it have some imperfect in functionality. Besides the voice recognition limitation, which could not always get the right answer when you say a word to it, let along phrases and long sentences, it also have some limitation in other field. For example, when recognize a face, it only judge it through trivial image information analysis but not related to any depth information. It means that if it just a photo of a human face, it will also consider it as a face. We tried use a photo on IPad screen, it turns out it has the same reaction. Some attributes in SDK library have not been implemented successfully, like there have 'elbow' tag in library but the SDK actually can do nothing about recognize an elbow.

Gesture recognition function is relatively robust. Gesture recognition is based on both image and depth information, which means if we put a hand photo in front of camera, the SDK would not consider as real hand. We put our emphasis on gesture direction, we have studied samples related to gesture and read documentation related to gesture carefully. We knew that within effective range, Intel Perceptual Computing SDK is able to recognize two hands' basic gestures like 'thumb up', 'thumb down', 'peace', 'big 5', 'swipe left'， 'swipe right', 'swipe up' and 'swipe down' simultaneously.

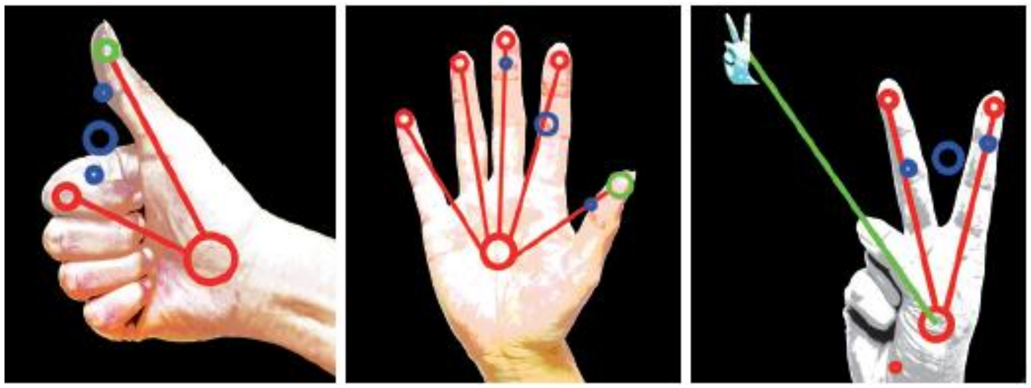


Figure14: Gesture recognition Samples

Besides, Intel Perceptual Computing SDK offered us many interface to retrieve many geometric nodes, like finger tips, finger joints, whole hand and etc., and their position's raw data. Raw data include x, y, z coordinates based on both image coordinate system and world coordinate. We can utilize those raw data and specific gesture recognition capability to build meaningful application.

Limitation

Although Intel Perceptual Computing has advantages over gesture recognition, it still has some limitations.

One of the uneasy limitations is that the SDK could not distinguish left and right side. Intel Perceptual Computing treats the first hand shown on camera as left hand, which lead to some inconvenience in those programs need to specify clearly hand sides. For example, if we want to control a small robot and when a person put up his left or right hand, the robot would put up his left or right hand respectively, it is easy to run into a unexpected situation that human's left hand mapped to robot's right hand and human's right hand controlled robot's left hand.

Although this problem can be solved if we restrict users must put left hand in front of camera first, it would largely worse user experience then. This limitation also caused us some problems when doing our own implementation. Finally we came up with a tricky method to avoid this problem. (Details about this problem solution were stated in chapter implementation.)

Another fatal limitation is about identifying precision, it is not a problem about accuracy, and it is about resolution. The typical problem caused by this limitation showed in the situation that when two feature points within effective range are too close to each other or when one point overlapped the other, these points would be lost tracking state. Because Intel Perceptual Computing does not have a very large effective detective range, that makes this problem worse.

In fact, of any kind of camera, if a point being overlapped by another, its information losing is understandable and not avoidable. So if when a point lost tracking it can throw a exception or return a default value to indicate this situation, it would be relative easy to handle. The problem is Intel Perceptual Computing SDK used a subtle method that it returns any finger’s correspondent information which did not lose tracking. So if a finger, say thumb lost tracking, if we read thumb information at the moment, we may get pinky finger or ring finger information. It is a dilemma and caused some awkward problems.

In general, applications based on Intel Perceptual Computing SDK have the best performance when user’s fingers separate widely and no crossing or overlapping.

Another small limitation is about stability. Sometime during tracking process, each feature nodes’ position some time would show discontinuity, which means jump to a ridiculous position for a very short time period or two detected position information exchanged with each other. Fortunately, this problem is just a very rare event which can be ignored.

During this semester's study and testing process, we found a new system limitation which would dramatically affect application performance. When there is a strong glowing object in the detective range of Creative Camera, like bright bulb, the light will jeopardize Camera's recognition ability, thus made the string lion puppet control become inaccurate.

<u>Similar SDK comparison</u>

Kinect for Windows SDK is the SDK for developers of Kinect, it doesn’t focus on game on Xbox but for human-machine interaction on PC, which has the same motivation with Intel Perceptual Computing SDK. But the Microsoft’s SDK is focus more on human’s skeleton while Intel’s SDK focus more on short range, using gesture as input.

Kinect windows SDK can capture at most 6 people’s skeleton and supply two people’s simultaneous operation. Kinect windows SDK can capture almost all body’s geometric nodes as input for developer to manipulate. Intel Perceptual Computing put emphasis on hand tracking, it can track human hands with much higher resolution compared to Kinect windows SDK. Intel SDK gives strong support on precise gesture recognition and identification. These two SDKs have lots of similarities and also many differences. They have their own focus.

To conclude, their advantages and disadvantages are summarized as follows.

Kinect

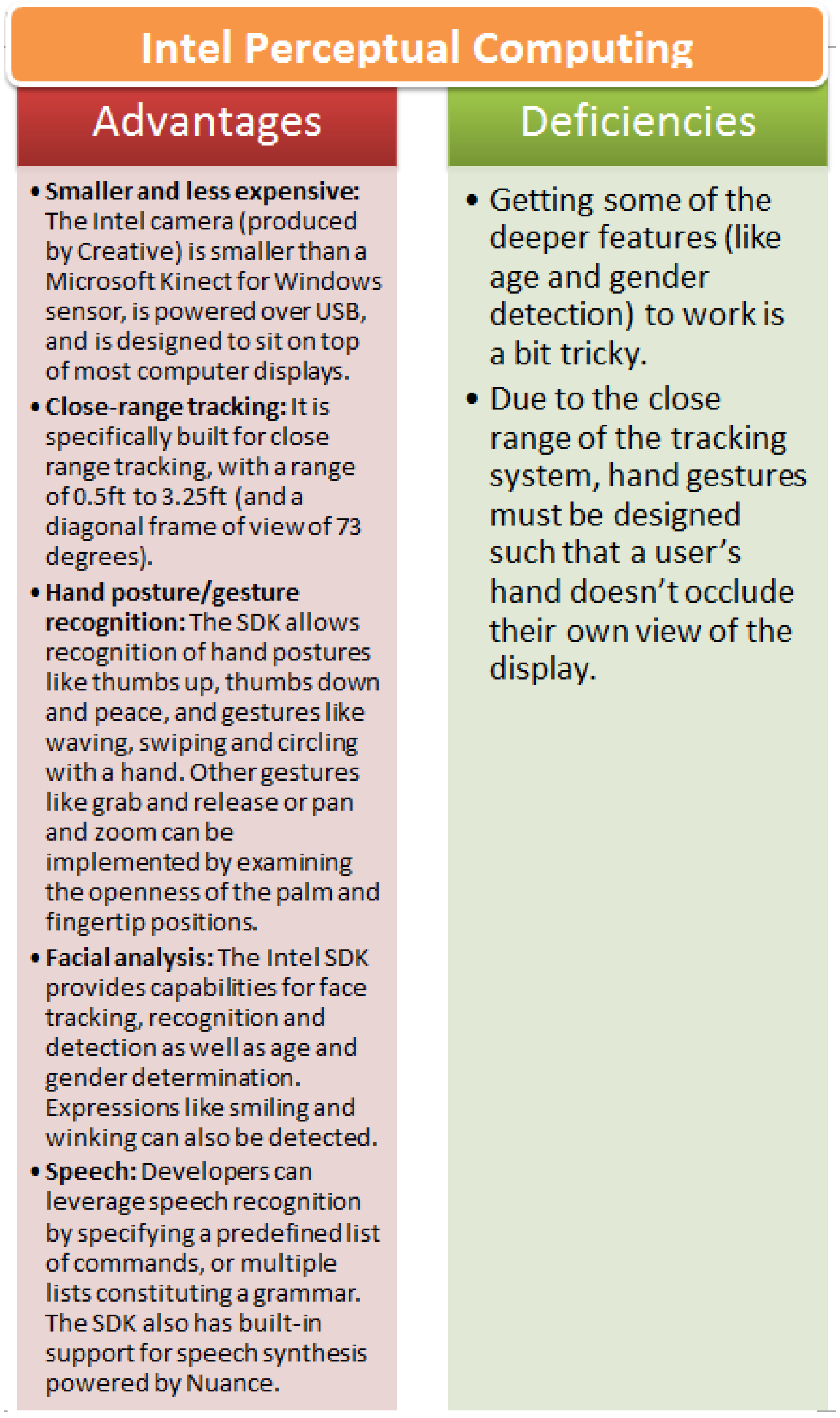
Advantages

- **Skeletal tracking:** Applications can track the positions of user's joints (head, shoulders, hips, hands, etc.) in space. Two people can be tracked simultaneously, in either standing or sitting poses.
- **Face tracking:** Various attributes of the user's face can be tracked, including the relative positions of lips and eyebrows, which can be interpreted as facial expressions.
- **Multiple sensors:** Multiple Kinect sensors can be used together in order to track more users simultaneously, or to get a wider view of a space.
- **Raw data:** The Kinect SDK provides access to the raw depth data from the sensor, as well as images from the IR and RGB cameras. This can be processed by the application developer as they see fit.
- **Voice control:** The microphone array in the sensor can be pointed toward the user to better capture speech. Developers can pre-define a set of commands constituting a grammar for an application. Users are informed when one of a set of commands has been spoken.

Deficiencies

- Compared to other sensors, the device is fairly large making it more difficult to conceal in an installation.
- A dedicated power cord is required because the sensor uses more power than can be provided by a USB bus. The amount of data generated by the sensor also tends to saturate a USB controller, so it's recommended that each sensor be on its own controller.
- The sensing resolution is not as advanced as other sensors on the market. For example, the Kinect cannot easily distinguish individual fingers on a hand, which means gestures tend to involve more gross movements than simple pointing.
- Up to six users can be recognized in the field of view of the sensor but only two users can be tracked in detail.
- While there are open-source toolkits for working with the Kinect on non-Windows platforms, most of the features listed above require the Microsoft SDK, which is only supported for desktop applications on Windows 7 and 8.

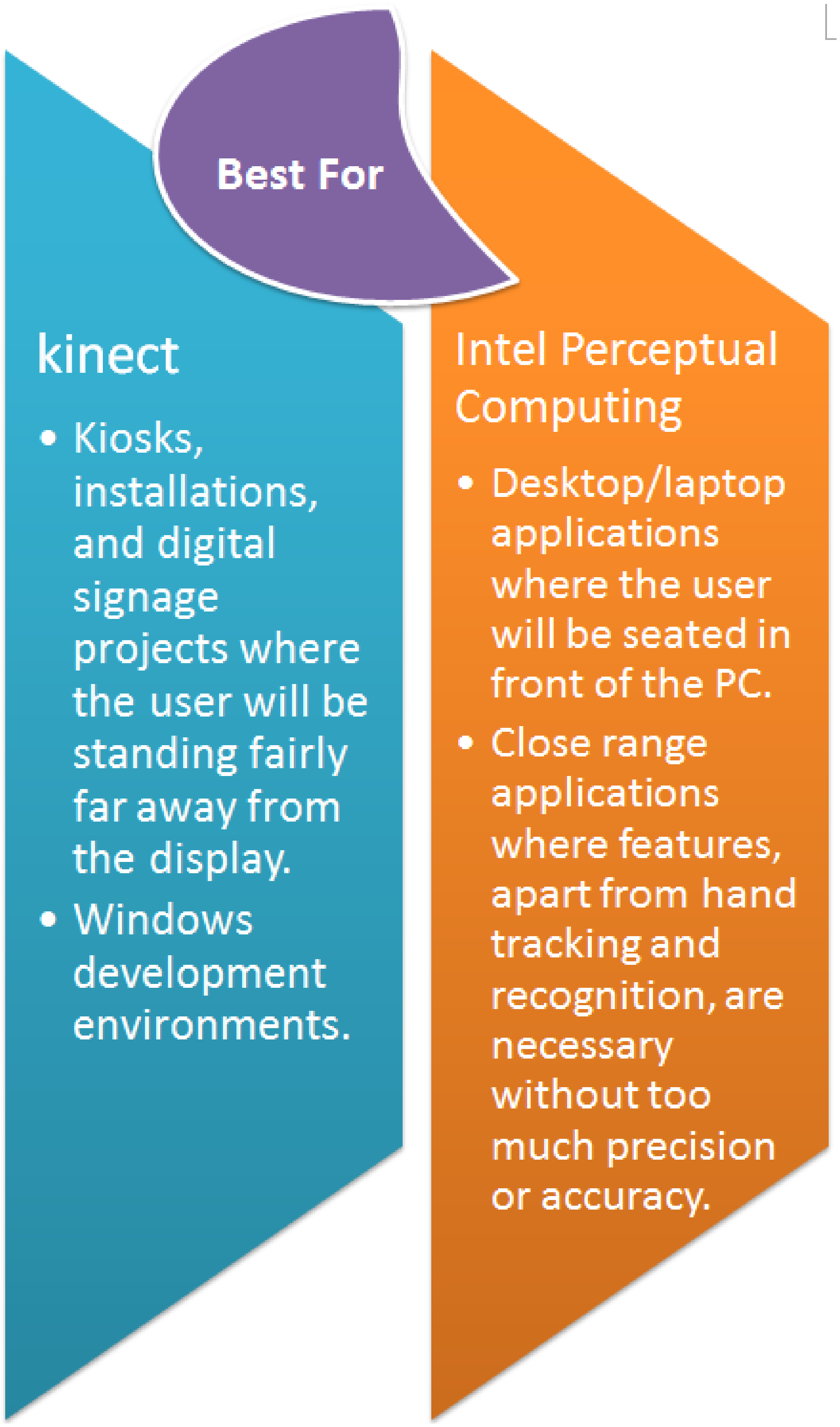
Table4: Kinect SDK Information Table



Intel Perceptual Computing

Advantages	Deficiencies
• **Smaller and less expensive:** The Intel camera (produced by Creative) is smaller than a Microsoft Kinect for Windows sensor, is powered over USB, and is designed to sit on top of most computer displays. • **Close-range tracking:** It is specifically built for close range tracking, with a range of 0.5ft to 3.25ft (and a diagonal frame of view of 73 degrees). • **Hand posture/gesture recognition:** The SDK allows recognition of hand postures like thumbs up, thumbs down and peace, and gestures like waving, swiping and circling with a hand. Other gestures like grab and release or pan and zoom can be implemented by examining the openness of the palm and fingertip positions. • **Facial analysis:** The Intel SDK provides capabilities for face tracking, recognition and detection as well as age and gender determination. Expressions like smiling and winking can also be detected. • **Speech:** Developers can leverage speech recognition by specifying a predefined list of commands, or multiple lists constituting a grammar. The SDK also has built-in support for speech synthesis powered by Nuance.	• Getting some of the deeper features (like age and gender detection) to work is a bit tricky. • Due to the close range of the tracking system, hand gestures must be designed such that a user's hand doesn't occlude their own view of the display.

Table5: Intel Perceptual Computing SDK Information Table



Best For

kinect

- Kiosks, installations, and digital signage projects where the user will be standing fairly far away from the display.
- Windows development environments.

Intel Perceptual Computing

- Desktop/laptop applications where the user will be seated in front of the PC.
- Close range applications where features, apart from hand tracking and recognition, are necessary without too much precision or accuracy.

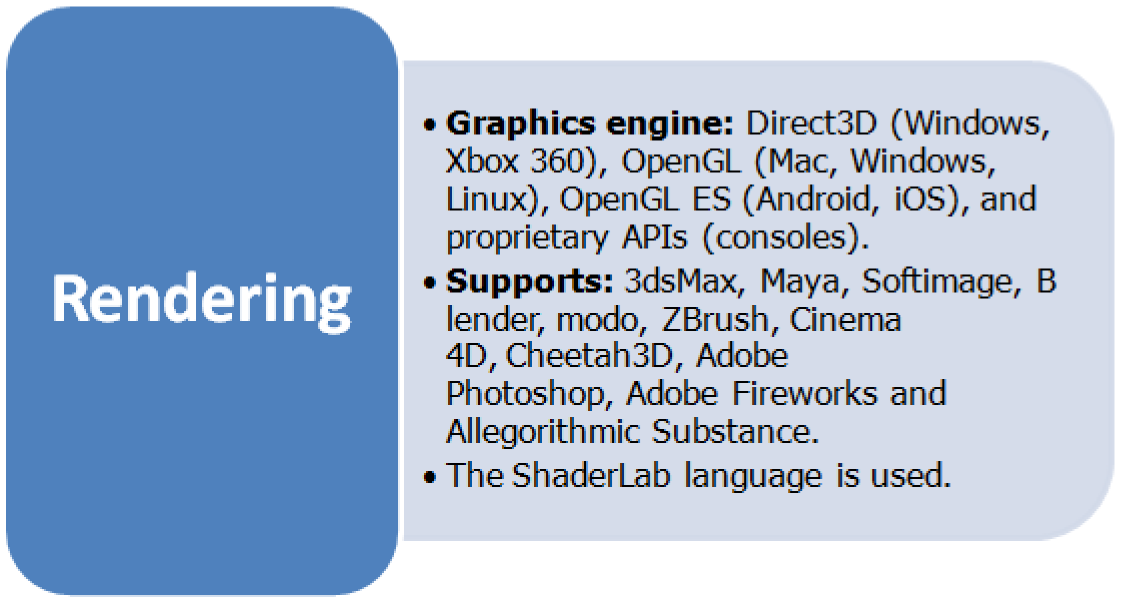
Table6: Best For what applications

2. Unity 3D[3]

Unity is a cross-platform game engine with a built-in IDE developed by Unity Technologies. It is used to develop video games for web plugins, desktop platforms, consoles and mobile devices. It grew from an OS X supported game development tool in 2005 to a multi-platform game engine.

Intel perceptual computing SDK supports Unity application with default Unity Version 3.5.1. which is the version we used in our project. Unity 3.5 was one of the largest releases for the Unity development platform and added many new features and improvements compare with old version. No matter in physics engine part or rendering part. Like better particle system, HDR rendering, obstacle avoidance so and so forth. Also it made adobe flash plug in supported.

<u>Specification</u>[3]



Rendering

- **Graphics engine:** Direct3D (Windows, Xbox 360), OpenGL (Mac, Windows, Linux), OpenGL ES (Android, iOS), and proprietary APIs (consoles).
- **Supports:** 3dsMax, Maya, Softimage, B lender, modo, ZBrush, Cinema 4D, Cheetah3D, Adobe Photoshop, Adobe Fireworks and Allegorithmic Substance.
- The ShaderLab language is used.

Scripting

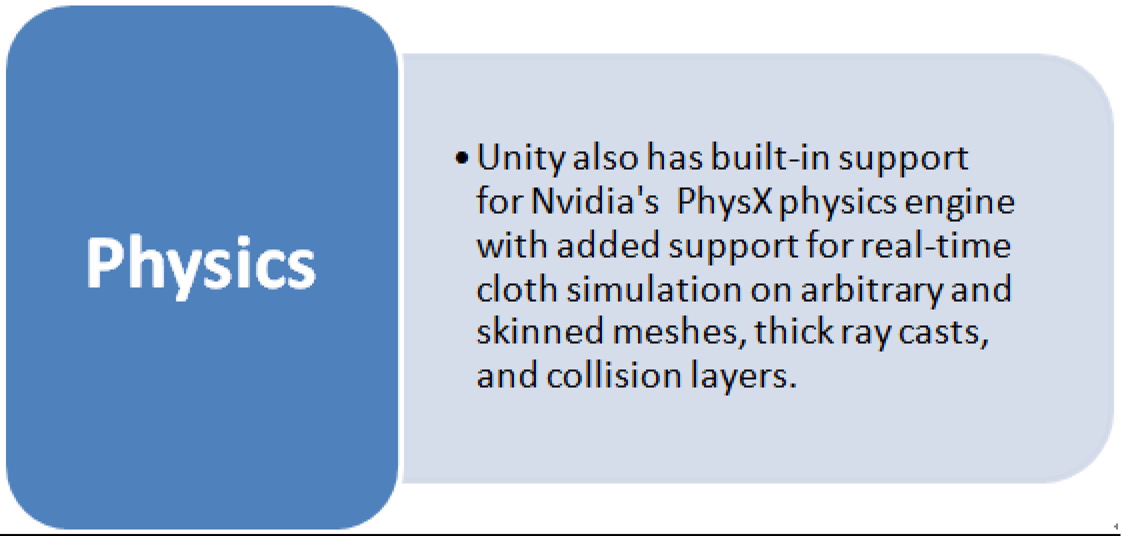
- Built on **Mono**---An open-source implementation of the .NET Framework.
- UnityScript (a custom language with ECMAScript-inspired syntax, referred to as JavaScript), C#, or Boo.
- Starting with the 3.0 release, Unity ships with a customized version of **MonoDevelop** for debugging scripts.

Asset Tracking

- Unity Asset Server included - a version control solution for the developer's game assets and scripts.
- PostgreSQL; FMOD library; Theora codec; built-in lightmapping and global illumination with Beast, multiplayer networking using RakNet, and built-in pathfinding navigation meshes.

Platforms

- **Current:** Playstation 4, Xbox One, BlackBerry 10, Windows8, Windows Phone8, Windows, Mac, Linux, Android, iOS, Unity Web Player,Adobe Flash, PlayStation 3, Xbox 360, Wii U and Wii
- **Upcoming**:PlayStation 4 and Xbox One.


Physics

- Unity also has built-in support for Nvidia's PhysX physics engine with added support for real-time cloth simulation on arbitrary and skinned meshes, thick ray casts, and collision layers.

Figure 15: Unity specification

3. Blender

Blender is a free and open-source 3D computer graphics software product used for creating animated films, visual effects, art, 3D printed models, interactive 3D applications and video games. Blender's features include 3D modeling, UV unwrapping, texturing, rigging and skinning, fluid and smoke simulation, particle simulation, soft body simulation, sculpting, animating, match moving, camera tracking, rendering, video editing and compositing. Alongside the modeling features it also has an integrated game engine.

Blender is a kind of free, small-size but useful modeling and animating software. Compare with professional and large modeling software like Maya and 3dsMax, blender can already achieve our expectation and have simple operating methods. Thus we chose it as our modeling software.

As for model texturing. We first use Blender to generate a suitable uv map and used GIMP to color it.

Hardware tool

1. Hardware - Creative* Interactive Gesture Camera

Introduction [4]

The Creative* Interactive Gesture camera Developer Kit (the "Camera") is a small, light-weight, USB-powered camera optimized for close-range interactivity. It is the camera produced by Creative Company as Intel required, it includes an HD webcam, depth sensor and built-in dual-array microphones for capturing and recognizing voice, gestures and images. The Camera, when paired with the Intel® Perceptual Computing SDK Beta 2013, enables developers to create the next generation of natural, immersive, innovative software applications that incorporate close-range hand tracking, speech recognition, face analysis and 2D/3D object tracking on Intel Core ™ processor-powered Ultrabook™ devices, laptops and PCs.

It should be noticed that the Camera is intended solely for internal use by developers with the Intel® Perceptual Computing SDK Beta 2013 solely for the purposes of developing perceptual computing applications. The Camera may not be used for any other purpose, and may not be dismantled or in any way reverse engineered.

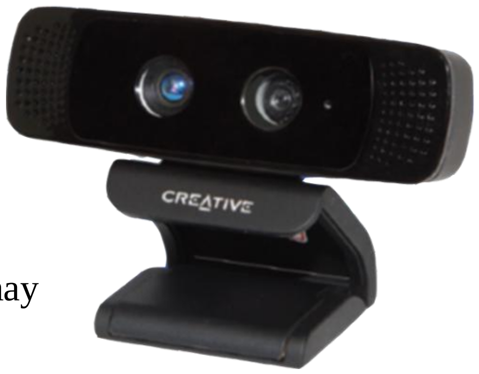


Figure16: Creative* Camera

Specification [5]

The Creative* Interactive Gesture Camera is intended to build a total new way of interaction. It main features include 3D gesture control technology, 3D facial sensing, 3D facial analysis and etc.

Detail specification is as follows.

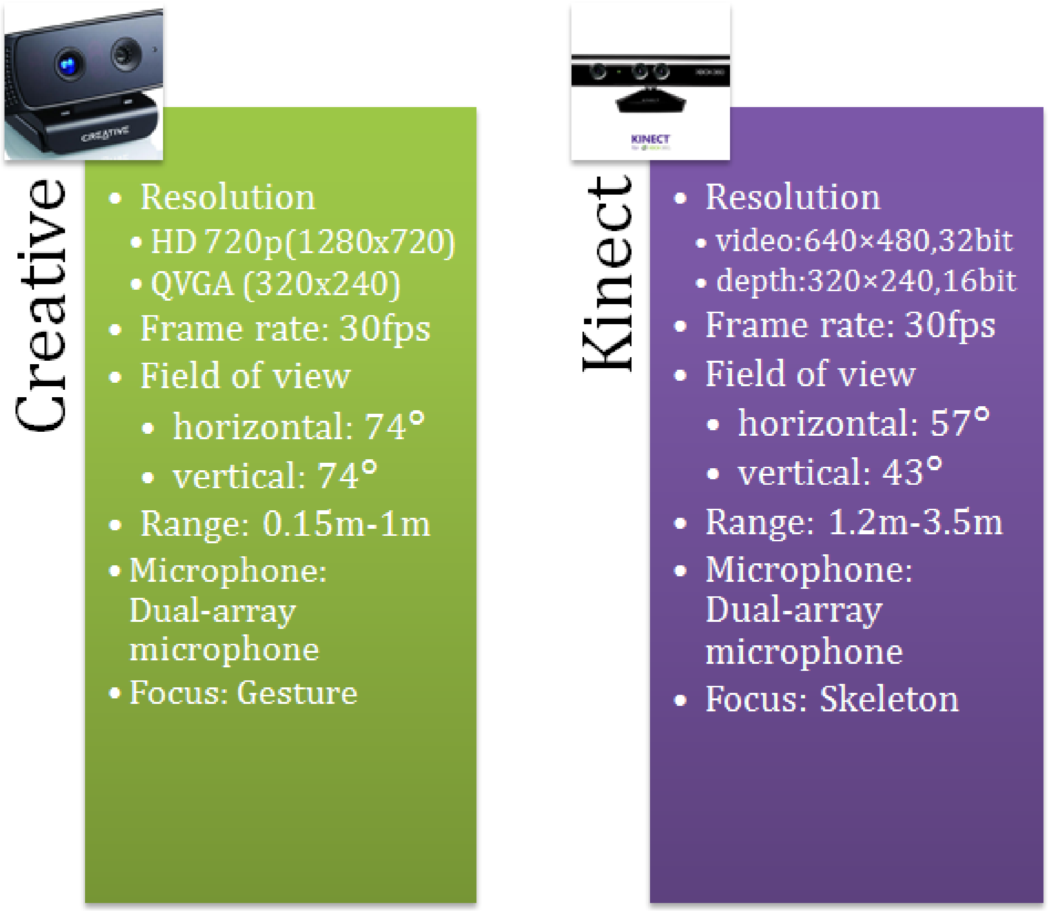
- RGB video resolution
  HD 720p (1280x720)
- IR depth resolution
  QVGA (320x240)
- Frame rate
  Up to 30 fps
- FOV (Field-of-View)
  74 °
- Range
  0.5ft ~ 3.25ft
- Dual-array microphones
- Size
  4.27” x 2.03” x 2.11”
- Weight
  271g
- Multi-attach base
- Cable length
  1.8 meters
- USB 2.0 Hi-Speed

System requirement is as follows.

- PC or laptop with 2nd Generation Intel® Core™ processor or higher
- Microsoft® Windows® 8 / Windows 7 with Service Pack 1 or higher
- 4GB RAM
- USB 2.0 or USB 3.0 port
- 4GB free hard-disk space

Similar Camera comparison

We will focus on comparing Creative* camera with Kinect camera. Here is a detailed parameter comparison table of these two cameras.



Creative

- Resolution
  - HD 720p(1280x720)
  - QVGA (320x240)
- Frame rate: 30fps
- Field of view
  - horizontal: 74°
  - vertical: 74°
- Range: 0.15m-1m
- Microphone: Dual-array microphone
- Focus: Gesture

Kinect

- Resolution
  - video:640×480,32bit
  - depth:320×240,16bit
- Frame rate: 30fps
- Field of view
  - horizontal: 57°
  - vertical: 43°
- Range: 1.2m-3.5m
- Microphone: Dual-array microphone
- Focus: Skeleton

Figure17: Creative* Camera vs. Kinect Camera

Although these two are both cameras with depth information, we can still tell their difference form the above table. Microsoft Kinect focus on tracking skeleton information as input, it has a further and wider working range, 1.2 to 3.5 meters. It support much more on user stand in front of it with some distance, which is not very suitable for PC application. Compared to it, Creative* camera has a shorter and narrower working range. And it has a higher resolution. Thus it support user near to it and use finger move or gesture to do operation. It is more suitable for PC. At the same time, Creative* camera has a wider FOV (field of view) with value is 74 degree. So it can do better in near distant operation, like gesture recognition we mentioned before and facial analysis. This also the reason we chose Creative* camera as our hardware tool, because it offered more supplement on gesture and hand move input. Creative* can have more interface and more accuracy information. It has better performance in close range manipulation.

Chapter3: Design and Implementation

<u>Design overview</u>

1. Overall Architecture

Our project contains several modules; include IO modules, data process module, data mapping module, lion-drawing module, and refinement module and physics engine. The relationship between each module is shown below.

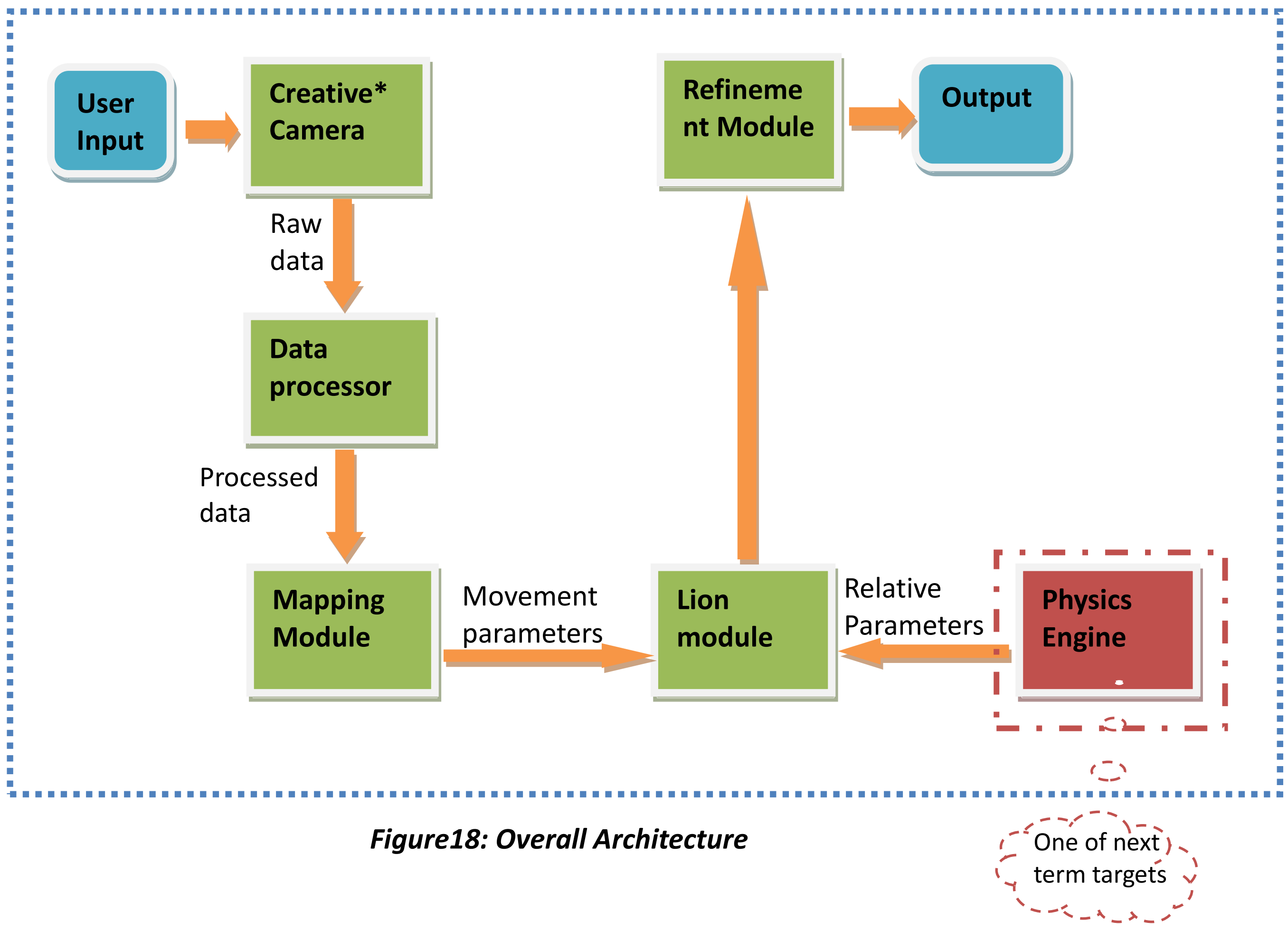


Figure18: Overall Architecture

2. Module description

IO module: Input module takes charge of receiving user input. It mainly acquires image information and depth information from CREATIVE* camera and forward raw data to data process module. Input module do not take care of any data manipulation, it just a data courier. Output module also plays a data courier role. But it transfer processed data and display lion movement on screen.

Data process module: This module is the first data manipulation module. It transferred raw data got from input module to data could be utilized by Intel Perceptual Computing SDK and openFrameworks. The concrete transformation work includes format transformation, numeric scaling, constant computing and such simple operation to make data become natural and easy to do further computing. Those predigested data than being transferred to data mapping module.

Data mapping module: Data process module takes charge of analyze processed data and use relevant data on mapping algorithm. Through computing, data mapping module would output necessary figures need in puppet lion transformation, like angles, rotate axis and displacement etc. basic arguments. Then, that relevant information will be transferred to lion-drawing module to manipulate string lion puppet model.

Physics module: Physics module is a kind of physics engine. It provides a real time approximate simulation of some physics systems. This simulation could provide appropriate gravity effect or inertia effect to make puppet's movement become more realistic. This module combine with lion drawing module would lead to a high real puppet effect simulation, which we will put our effort on next semester.

Lion-drawing module: Lion-drawing module is a model rendering module. As its name implicated, its duty is to render a lion puppet on screen and make correspondent transform to simulate lion puppet movement in reality. Those results will be sent to output module. openFrameworks play the most important role in this module.

3. Functional Specification

This section will show general functions that the project can perform

a. Puppet lion translation with user hand movement

Lion puppet can translate on the screen with user's hand move left and right, up an down, and the move direction keeps in accordance with hands move direction. This control method would give user more intuitionistic experience on puppet control.

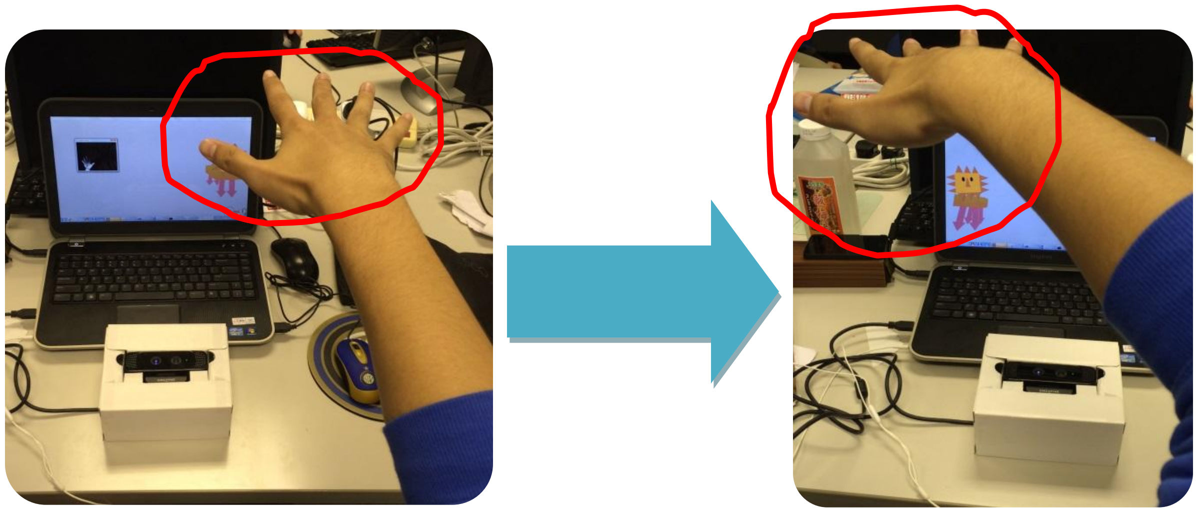


Figure19: move hand from right to left

b. puppet lion rotate along any axis with user hand rotation

When the user hand rotates around different directions in camera's effective range, string puppet lion on the screen can rotate in accordance with user hand rotation direction. We fix lion head direction is the same as wrist direction, tail direction is the same as finger tips direction. But we need pay attention that because Intel Perceptual Computing SDK limitation we mentioned before, when user rotate his hand there are possibilities some finger tips may lose tracking.

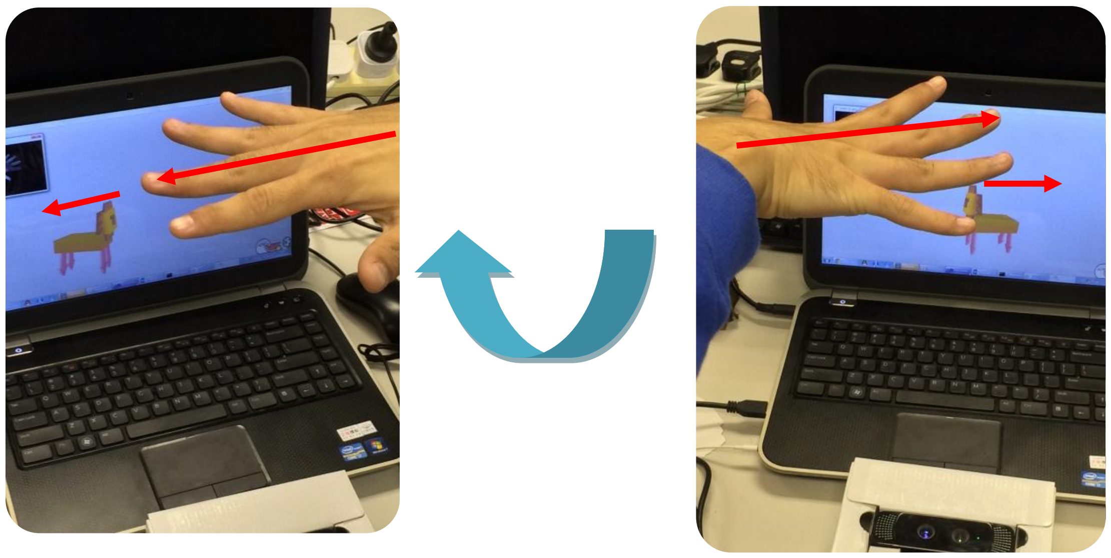


Figure20: Rotate hand clockwise

c. <u>puppet lion's body and legs swing with user hand swinging</u>

When user hand swings left and right, forward and backward, lion puppet's body and legs would do similar swing action.

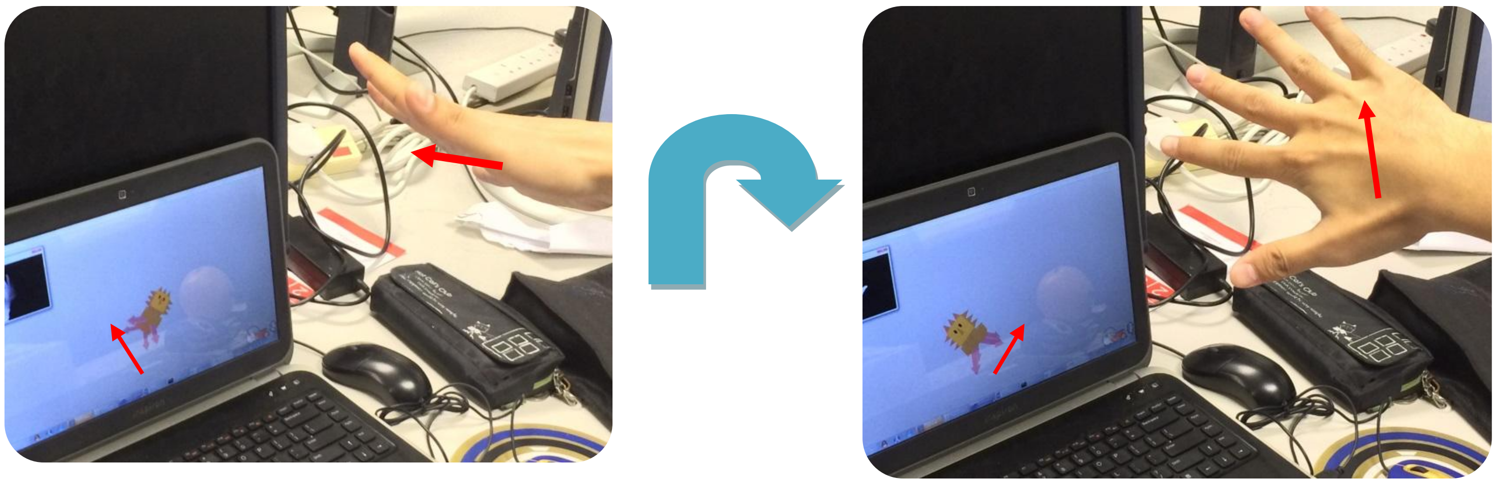


Figure21: Rotate hand clockwise

d. <u>Puppet lion's eyes can change with user's gesture change</u>

The puppet lion's eyes can change from '0 0' to '> <' if user gives a 'peace' gesture and restore '00' if user do not hold peace gesture (figure14)

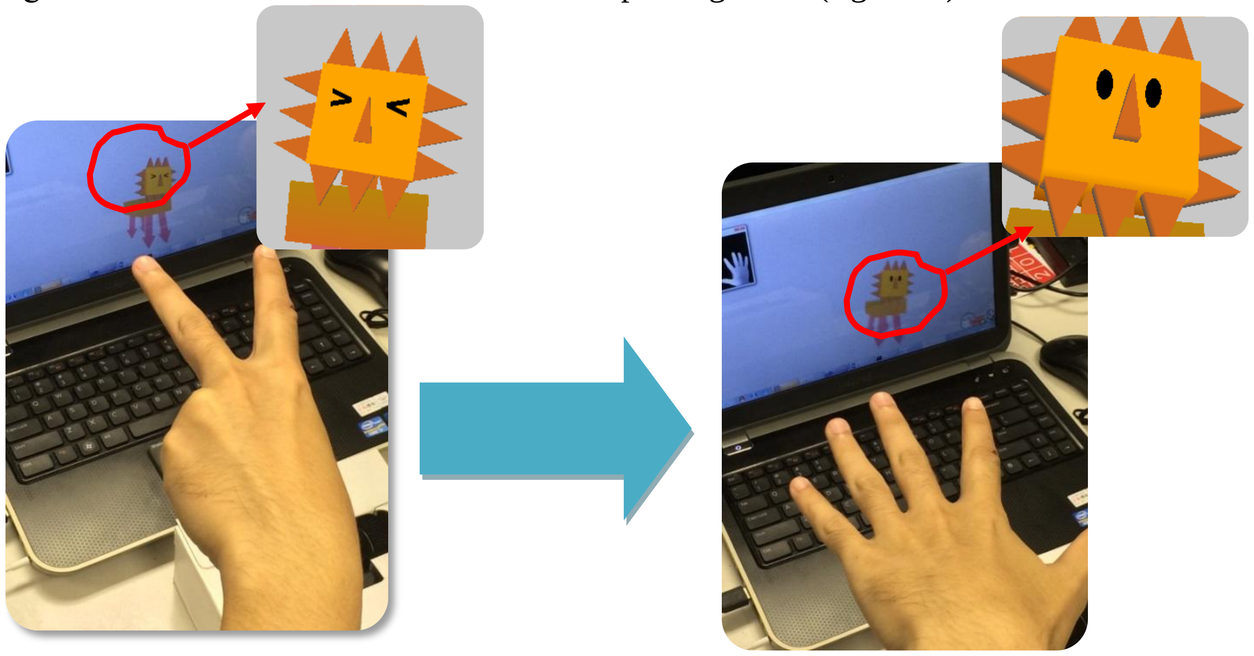


Figure22: Eyes' shape change with different gesture input

e. <u>Lion head sway with fingers lifting up and down</u>

The puppet lion's head will sway if left hand index and middle fingers lifted up and down. If lift index finger up, lion head will sway left and if lift middle finger up, lion head will sway right.

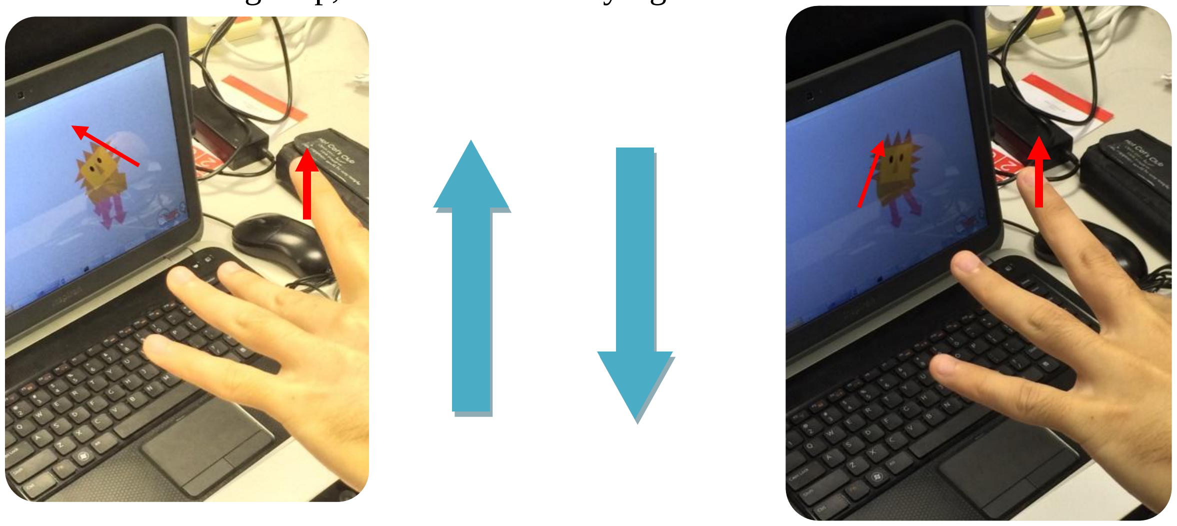


Figure23: Head sway when lift fingers up and lay down

Implementation detail

1. Lion model

Initial

The lion puppet model prototype we made last semester is quite simple and trivial. We used Openframeworks to draw some simple primitives, like cubes, arrows, and then do transformation and combination to made them into a cartoon-looking lion puppet. Basically the lion puppet we made last semester looks like a logo game robot. No texture, no lighting and no physics element.

We can see from next picture, it is just some cubes.

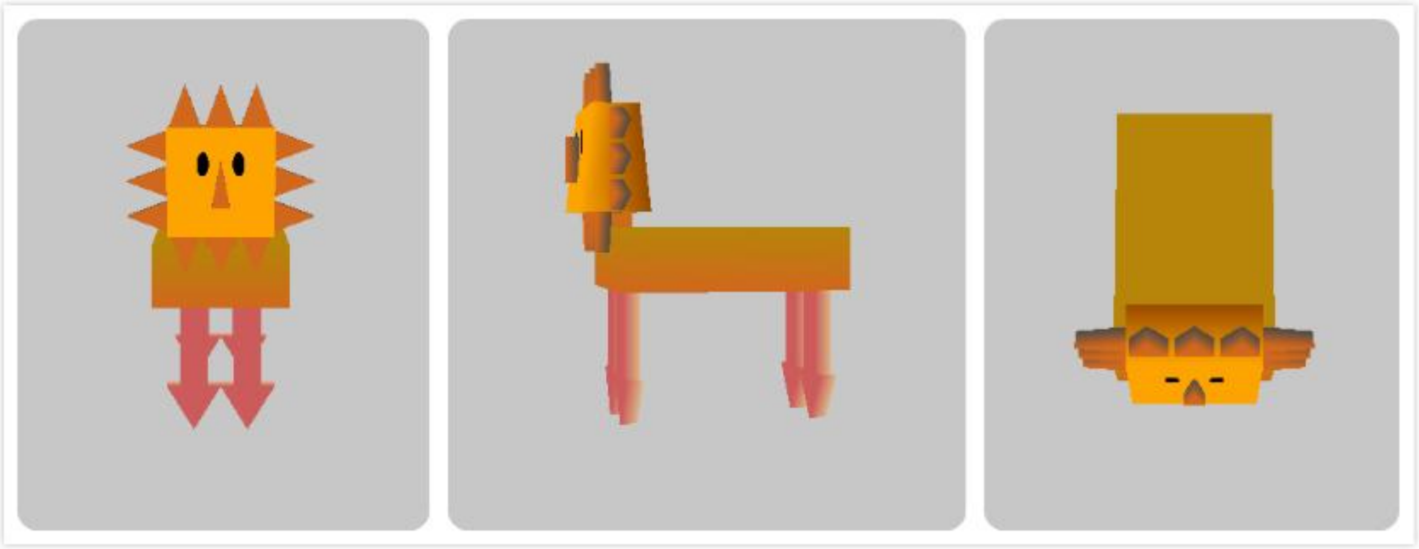


Figure 24: iNITIAL VERSION---Lion model three view drawing

Finalized

Since in this semester, our focus moves to bettering our project. Model rendering becomes important in this semester. Besides, to add physics engine, we also need a solid model. Although there are many model material website offering free model downloading, but there is no string lion 3D model on the internet. Unfortunately, we have to made this model by ourselves.

We did modeling part using Blender Version 2.6. After study into this tool, we build a relative similar string lion puppet. Since we are not art student, although it is not very perfect, we are still satisfied with it.

After modeling, because we need to append several animation clips on this lion, like mouse open and close on it, we need to build bones and bind it. Still, we used Blender. Actually, before using Blender, we tried Maya. And when we reached animation baking step, problems shows, Unity Version 3.5.1 have conflict with Maya model. So we changed to blender and found the result workable on Unity.

The final looking is as follows.

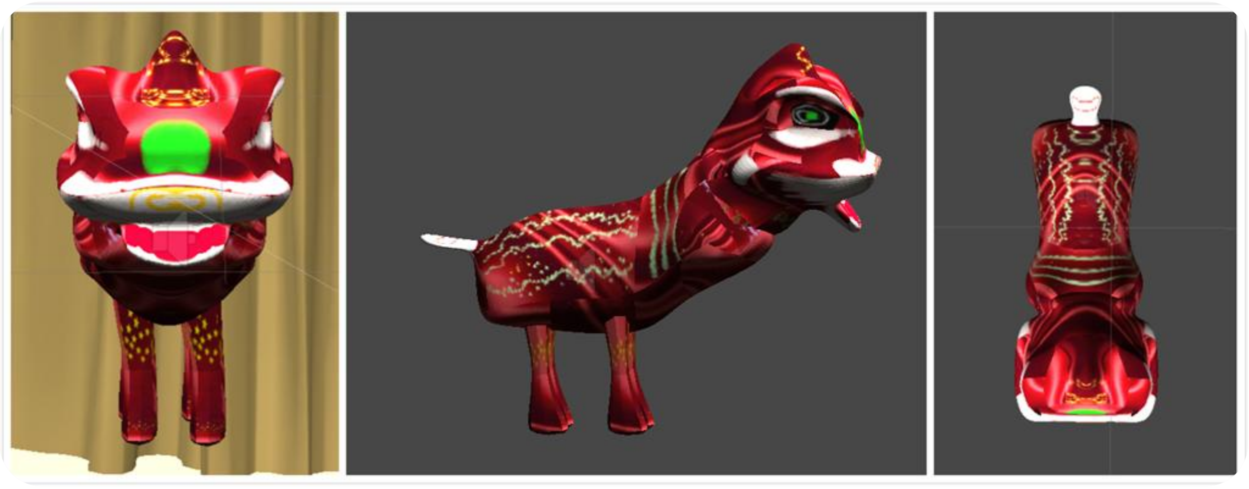
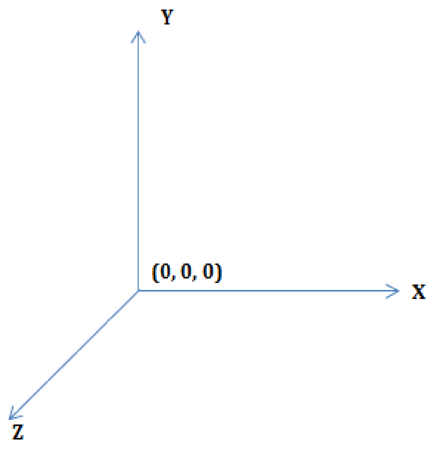


Figure 25: FINALIZED VERSION---Lion model three-view drawing

2. Geometric Nodes tracing

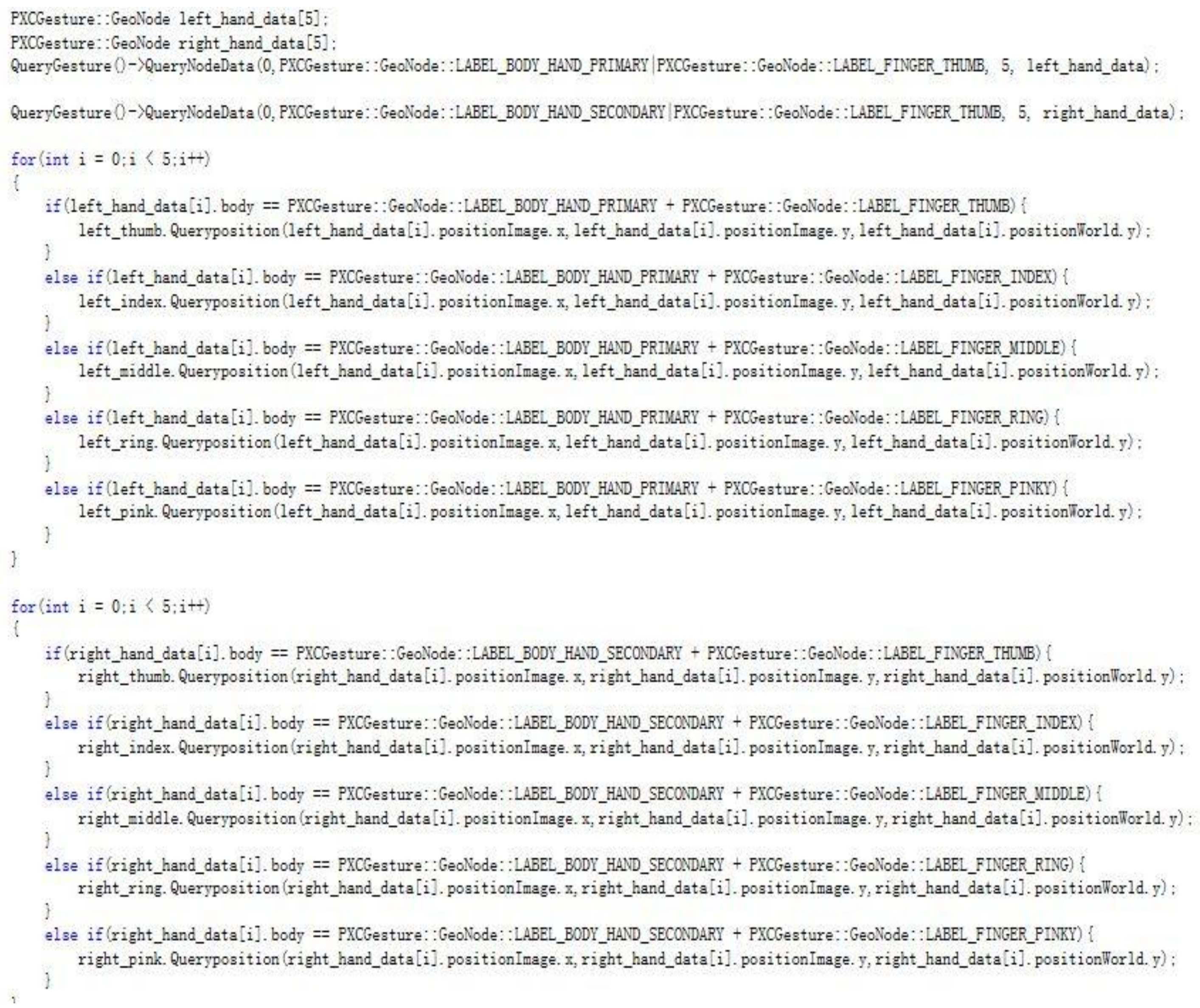
****Without specific notification, x-axis, y-axis and z-axis mentioned in later chapters are all based on a coordinate system those x-axis points to right direction, y-axis points to upward direction and z-axis points to outside direction.*



Term1 Implementation

The Intel Perceptual Computing SDK provides us many interface to retrieve all fingers' information. Utilize QueryGesture( ) in Intel Perceptual Computing SDK, we are able to obtain specific geometric nodes' (abbreviate to GenNode) position on x, y and z-axis simultaneously in each frame, and access attributes of hand's gesture. There is a class named GenNode in Perceptual Computing SDK library to do this job. For example, function QueryGesture()->QueryNodeData(0,PXCGesture::GeoNode::LABEL_BODY_HAND_PRIMARY, 5, left_hand_data) read exactly five left hand fingers' information and store them in left_hand_data and array structure. There is one thing need to be announced is that we use LABEL_BODY_HAND_PRIMARY instead of LABEL_BODY_HAND_LEFT is because a limitation we mentioned before that this SDK default consider the first hand shown in camera is the left hand.

The access demonstration is as follows.



```
PXCGesture::GeoNode left_hand_data[5];
PXCGesture::GeoNode right_hand_data[5];
QueryGesture()->QueryNodeData(0,PXCGesture::GeoNode::LABEL_BODY_HAND_PRIMARY|PXCGesture::GeoNode::LABEL_FINGER_THUMB, 5, left_hand_data);

QueryGesture()->QueryNodeData(0,PXCGesture::GeoNode::LABEL_BODY_HAND_SECONDARY|PXCGesture::GeoNode::LABEL_FINGER_THUMB, 5, right_hand_data);

for(int i = 0;i < 5;i++)
{
    if(left_hand_data[i].body == PXCGesture::GeoNode::LABEL_BODY_HAND_PRIMARY + PXCGesture::GeoNode::LABEL_FINGER_THUMB){
        left_thumb.Queryposition(left_hand_data[i].positionImage.x,left_hand_data[i].positionImage.y,left_hand_data[i].positionWorld.y);
    }
    else if(left_hand_data[i].body == PXCGesture::GeoNode::LABEL_BODY_HAND_PRIMARY + PXCGesture::GeoNode::LABEL_FINGER_INDEX){
        left_index.Queryposition(left_hand_data[i].positionImage.x,left_hand_data[i].positionImage.y,left_hand_data[i].positionWorld.y);
    }
    else if(left_hand_data[i].body == PXCGesture::GeoNode::LABEL_BODY_HAND_PRIMARY + PXCGesture::GeoNode::LABEL_FINGER_MIDDLE){
        left_middle.Queryposition(left_hand_data[i].positionImage.x,left_hand_data[i].positionImage.y,left_hand_data[i].positionWorld.y);
    }
    else if(left_hand_data[i].body == PXCGesture::GeoNode::LABEL_BODY_HAND_PRIMARY + PXCGesture::GeoNode::LABEL_FINGER_RING){
        left_ring.Queryposition(left_hand_data[i].positionImage.x,left_hand_data[i].positionImage.y,left_hand_data[i].positionWorld.y);
    }
    else if(left_hand_data[i].body == PXCGesture::GeoNode::LABEL_BODY_HAND_PRIMARY + PXCGesture::GeoNode::LABEL_FINGER_PINKY){
        left_pink.Queryposition(left_hand_data[i].positionImage.x,left_hand_data[i].positionImage.y,left_hand_data[i].positionWorld.y);
    }
}

for(int i = 0;i < 5;i++)
{
    if(right_hand_data[i].body == PXCGesture::GeoNode::LABEL_BODY_HAND_SECONDARY + PXCGesture::GeoNode::LABEL_FINGER_THUMB){
        right_thumb.Queryposition(right_hand_data[i].positionImage.x,right_hand_data[i].positionImage.y,right_hand_data[i].positionWorld.y);
    }
    else if(right_hand_data[i].body == PXCGesture::GeoNode::LABEL_BODY_HAND_SECONDARY + PXCGesture::GeoNode::LABEL_FINGER_INDEX){
        right_index.Queryposition(right_hand_data[i].positionImage.x,right_hand_data[i].positionImage.y,right_hand_data[i].positionWorld.y);
    }
    else if(right_hand_data[i].body == PXCGesture::GeoNode::LABEL_BODY_HAND_SECONDARY + PXCGesture::GeoNode::LABEL_FINGER_MIDDLE){
        right_middle.Queryposition(right_hand_data[i].positionImage.x,right_hand_data[i].positionImage.y,right_hand_data[i].positionWorld.y);
    }
    else if(right_hand_data[i].body == PXCGesture::GeoNode::LABEL_BODY_HAND_SECONDARY + PXCGesture::GeoNode::LABEL_FINGER_RING){
        right_ring.Queryposition(right_hand_data[i].positionImage.x,right_hand_data[i].positionImage.y,right_hand_data[i].positionWorld.y);
    }
    else if(right_hand_data[i].body == PXCGesture::GeoNode::LABEL_BODY_HAND_SECONDARY + PXCGesture::GeoNode::LABEL_FINGER_PINKY){
        right_pink.Queryposition(right_hand_data[i].positionImage.x,right_hand_data[i].positionImage.y,right_hand_data[i].positionWorld.y);
    }
}
```

The code above is in accordance with ten fingers data accessing in one frame.

Then we stored them in an array represent the whole hand and each element represents one finger. Queryposition( ) is a member function defined in class finger which is a class defined by ourselves. We use function Queryposition( ) to pass parameters to fingers corresponding position information. It is worth noting that image coordinate system is different from world coordinate system. In image coordinate system, only x and y axis matters and they are correspondent to horizontal and vertical direction. However, in world coordinate system, the three arises defined as following pattern.

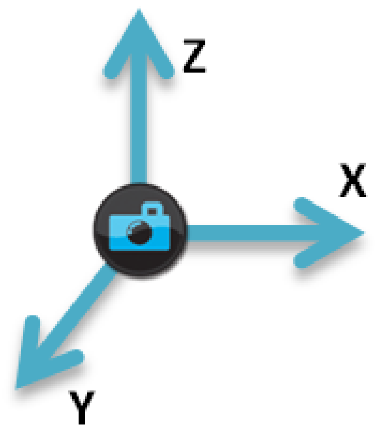


Figure 26: World coordinate system

Thus we can see that in world coordinate, y-axis is related to depth information instead of traditional z-axis. This detail is easy to be neglected; we have also be trapped for quite a while. So in our code, we read hand_data[i].positionWorld.y as depth information.

Term2 Implementation

Task in Term2 is similar to Term1. Because we have already implemented a reasonable algorithm in C++ in term1, what we need to do this semester is to transfer it in C#. The main difference between C++ and C# in our project is related to perceptual computing SDK functions. For example, Query node information function format differenceses, using C++ we write QueryGesture()-

>QueryNodeData(0,PXCGesture::GeoNode::LABEL_BODY_HAND_PRIMARY, 5, left_hand_data) while using C# we write QueryGeoNode(PXCMGesture.GeoNode.Label.LABEL_BODY_HAND_PRIMARY, out hand_data). We use this function to get Geonodes information of different fingers. As for raw data processing, we still choose x-axis and y-axis from image coordinate system attributes and z-axis from y direction data of world coordinate system.

3. Raw Data manipulation

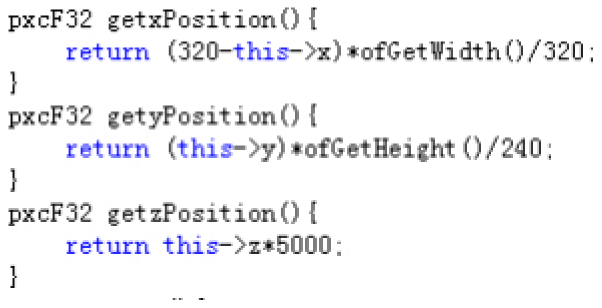
Term 1 Implementation

Input module is our data receiver. In this module, our program read all ten fingers' position information and return ten groups of x, y and z values where x, y use x and y value under image coordinates system and z use z value under world coordinates system. This is for more flexible manipulation.

What we need to do is to map each finger's position information data map to a appropriate position on display window. Each image frame we read has resolution 320 * 240. Thus under this resolution, x has position region from 0 to 320 and y has position region from 0 to 240. However, the actual display window size depends on user's computer display's actual resolution. Because different computer display may has different resolution, thus the object output position should depends on correspondent computer resolution, hardcode is not applicable in this case. Fortunately, there is two functions ofGetHeight() and ofGetWidth() in windows program library can be utilized. That we can dynamically set coordinate to each position.

First, we read current computer's resolution information, and use formula accx = x/320*ofGetWidth(), accy=y/240*ofGetHeight() to get actual coordinate, where accx is computed x coordinate, x is image x coordinate, accy is computed y coordinate and y is image y coordinate. As for depth information, because image coordinate system is a two dimension coordinate system that z coordinate is always zero, we can only use world coordinate depth information. However, the unit of world coordinate system is meter which is quite awkward for manipulation. So have to find out a scale to make it suitable for us to do further computing. After many times testing, we decide accz = 5000 * z can meet our expectation.

Code details about raw data processing are as follows.



```
pxcF32 getxPosition(){
    return (320-this->x)*ofGetWidth()/320;
}
pxcF32 getyPosition(){
    return (this->y)*ofGetHeight()/240;
}
pxcF32 getzPosition(){
    return this->z*5000;
}
```

Term 2 Implementation

In Unity3D project, when we run an application, the field of view or resolution is totally dependent on the main camera in each scene. Thus we can not directly based on computer resolution to map raw position data to the screen range. This is also a main difference between this semester's work and last semester's work. To set up a suitable field of view, we did different tests based on different parameters. We adjust raw position data by transforming and rotating. When our camera is fixed at origin point and has field of view with 90 degrees, the way we deal with raw position data read from camera is shown as follows. (FIGURE B)

Because we want our project to be a complete application which means our string lion puppet should work properly as we built those codes as a executable file. We want our lion puppet could dance on a stage. To fulfill that requirement, we actually want the string lion movement is constraint at a certain range. Imagine when you go to see a puppet play, there is a box-like stage within which the puppet will shown but the performer and performer is hide from audiences. So we add several 'max' and 'min' position constraint on it and not allow the puppet to exceed them.

To simulate realistic situation, when we using the virtual strings to control that lion puppet, we are not doing linear or direct mapping on the lion's position as what we did last semester. We are intended to make it looks like real string drag movement on it. To reach this target, we first try to use Unity physics element by adding some forces on it. But that would not work because the movement is still depends on the position data we got from camera. So we decide to manipulate those data to simulate the drag-like movement. Our method is to use average value. The string movement is not a linear map of our hand movement in one frame, but a map of average movements of ten recent frames. Thus during the beginning, the lion puppet's movement will lag behind our hands' movement and once our hand stop moving, the puppet position would finally catch up with the move. Just similar to we drag puppet in the real world.

4. Hand Position & Orientation decision

String lion puppet's move should be highly synchronized with user hands' move. So it is crucial to map hand or finger's relevant information with puppet lion model's transformation. In order to get best result, we tried several different map methods. We will introduce them at the next paragraphs.

1. Translation

Whole puppet body's translation can be down at the most beginning of lion-drawing module. We only need to invoke one function in openFrameworks library, ofTranslate (x, y, z) to realize it. The function ofTranslate (x, y, z) will move object to positive direction on X-axis, Y-axis and Z-axis with displacement equal to x, y and z respectively.

Then the key point became to get appropriate values of x, y and z.

Our idea is to treat a position p (xp, yp, zp) as whole hand's central position, then map p to computer screen and consider puppet lion at the same position p. Thus the whole lion body movement is synchronized with hand movement.

At first, we are trying to use a geometric node point represents middle of palm in Intel Perceptual Computing SDK with function attribute 'LABEL_LEFT_HAND_MIDDLE' in function QueryGesture()->QueryNodeData(0,PXCGesture::GeoNode::LABEL_LEFT_HAND_MIDDLE, 1, left_hand) we used in geometric nodes tracing.

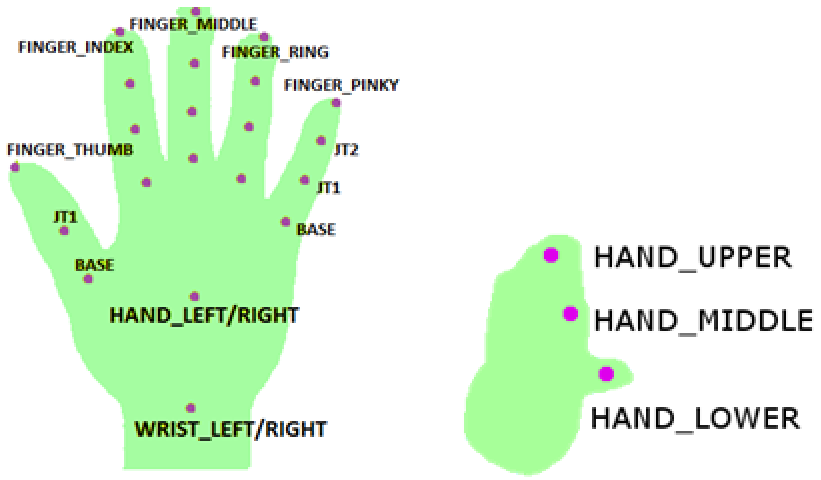


Figure 28: Geometric/Feature Nodes illustration

However, because of stability and accuracy deficiency, this SDK can not always track to expected position and kept in the same point. For example, in the next diagram, mark red 'x' represents an expected position. In practice, it will jump randomly in a small region sometimes. Then it will affect lion puppet's position correspondently. So only use this point to decide string lion puppet's position is not desirable. This plan is abolished

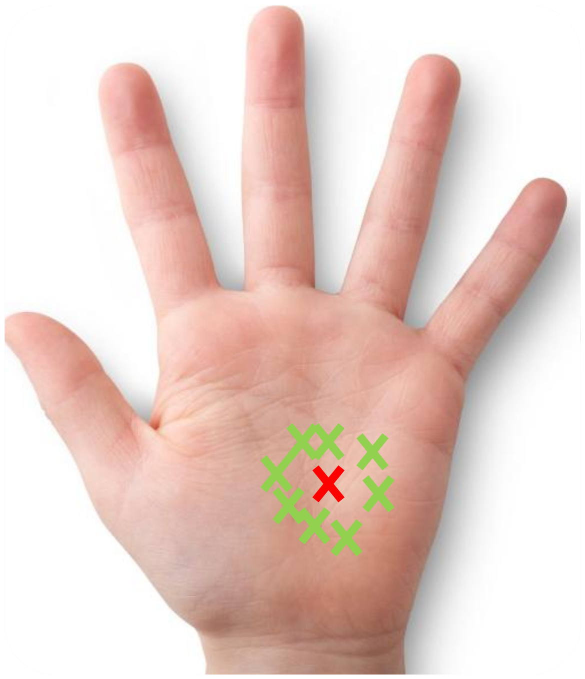


Figure 29: Instability Analysis

after many times testing.

Then we tested many other geometric nodes, upper and lower position of hand, ten finger tips and etc. The result shows that 1) finger tips has better stability, but they are the main move resource and very flexible that can not represent the whole hand 2) geometric node on palm moves less but with higher unpredictability that can not solve our problem.

It shows that only use one point to represent whole hand position is not workable. So we decide to use multiple points to decide that x, y, z value related to translation, which in code is like ofTranslate(centerx, centery, centerz) where centerx, centery and centerz represent values of the ten fingers x, y and z values respectively.

This mapping method can approximately reach our expectation. But it used too much fingers' information that added restriction on doing other mapping with those fingers' information.

We finally decide use one hand to work as main position control and the other hand can make puppet to do other movement as supplementary. We use right hand as that control hand, then in detail, we use mean value of five fingers' points coordinates as x, y, z wanted. Such that, ofTranslate(right_centerx, right_centery, right_centerz). This mapping method is slightly unstable

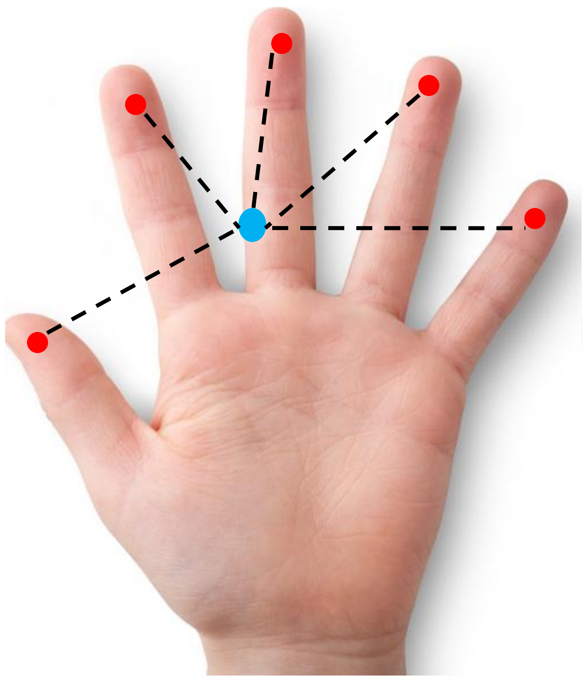
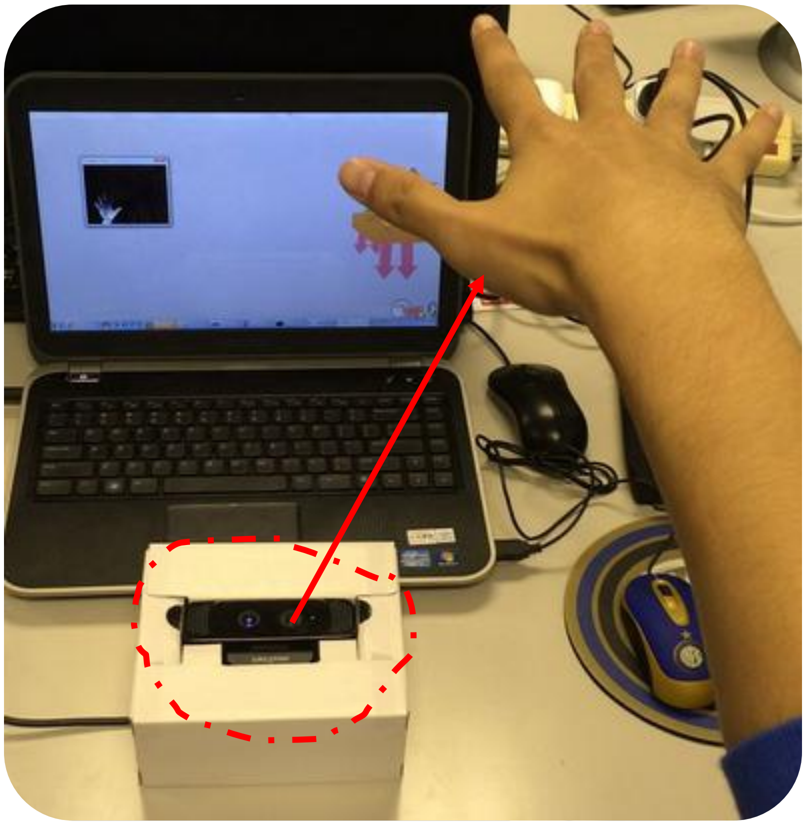


Figure 30: Mean value computing

compared to the former method related to all ten fingers, but it is still acceptable and can give us freedom to do other control use left hand.

Another point need to be paid attention is that in our practical operation, CREATIVE* camera faces upward, thus we actually put our hand over camera. It is for better simulation of real puppet control. Thus the depth data we read from camera is ultimately mapped to y coordinate in display, y coordinate of camera is mapped to z coordinate on screen and x is still mapped to x.



2. Rotation

In openFrameworks, rotation operation can be realized using a trivial function ofRotate(angle, x, y, z). The usage is similar to opengl, angle means rotate angle and x, y, z represent rotate axis. In default, rotate axis would pass primitive center.

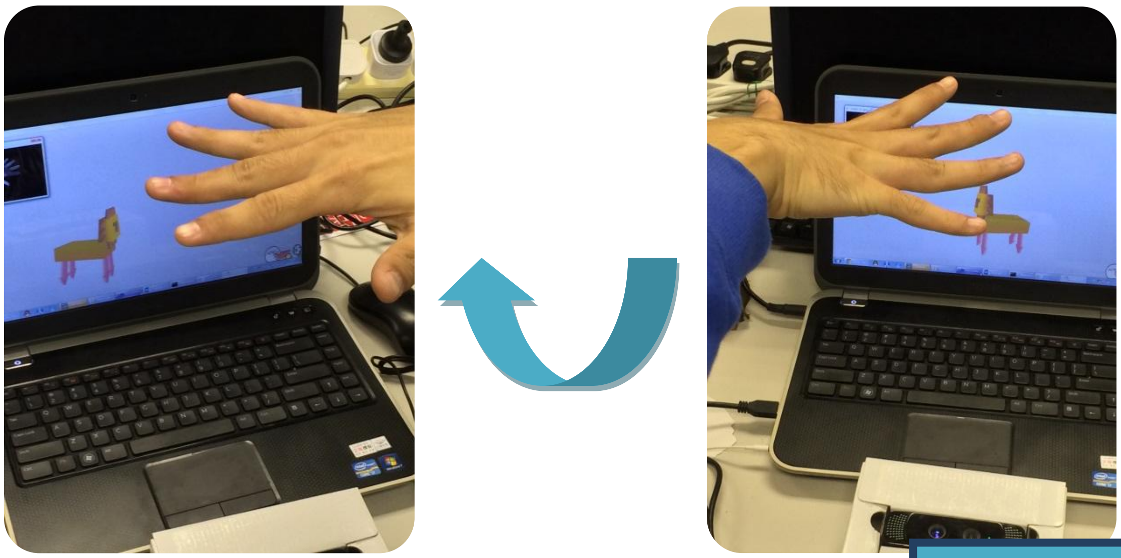
Since we use right hand as main control over string lion puppet translation, we do not hope its function just restricted to move towards horizontal or vertical direction. We try to add rotation on right hand also. When right hand rotate, which means palm plane is not kept horizontal or fingers do not point to screen, that virtual lion puppet can maintain the same direction as right hand’s direction.

To this end, we need to work out a group of suitable x, y, z and angle value to reach or expectation that make virtual lion puppet rotate with our hand rotation. Due to complexity and magnitude of finger position data, we cannot find out a effective algorithm to compute those four values at the one time. So we separate this process into several steps. We let the virtual puppet lion rotate three times with rotate axis are y-axis (0,1,0), z-axis(0,0,1) and x-axis(1,0,0) respectively. Use this method could reduce unknown variable to one, that is rotate angle. It largely reduced computation complexity and workload.

Detail angles computation algorithms.

a). Angle_y

First is about rotate angle over y-axis(0,1,0). In the initial state, right hand fingers point into computer screen while virtual lion puppet face towards computer screen outside. When a hand rotate around y-axis, say, in anticlockwise direction for a specified angle, thus at this moment, fingers point into inside left of computer screen. So we need our virtual lion puppet follows this hand to rotate the same angel, that its head should face to the right corner outside the computer screen. From figure at the left, we can see that the actual rotate angle can be represented by the angle between middle finger tips and negative direction of z-axis (0,0,-1).



At this point, what we need to do is to use right hand's five fingers to represent this middle finger's pointing direction. Of course we can not use middle fingers coordinates because of the instability and unexpected problems we mentioned above. We need to find another way to go to destination. After many times testing and verification, we got a quite reasonable formula to compute right middle finger's direction vector.

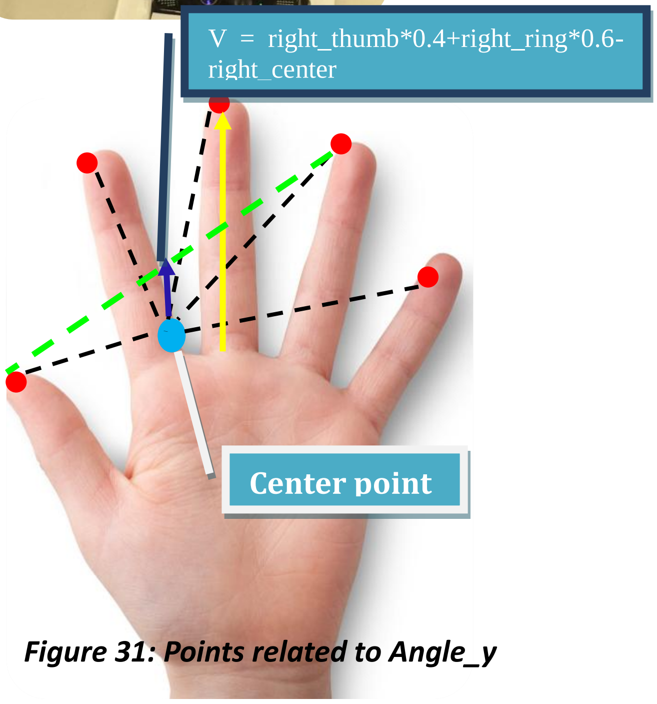


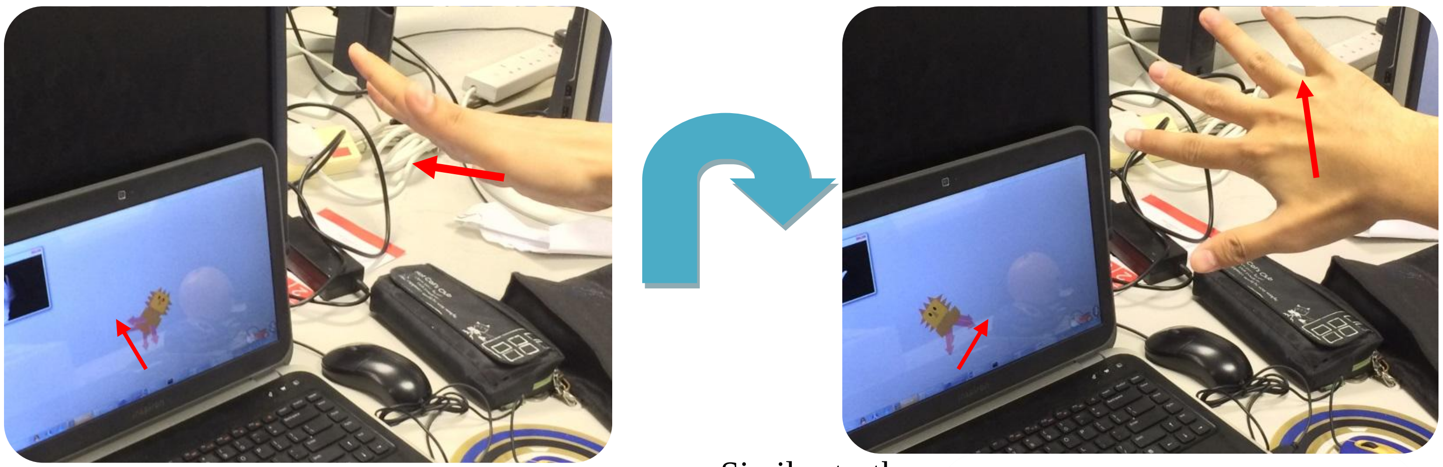
Figure 31: Points related to Angle_y

Correspondent direction vector = right_thumb*0.4+right_ring*0.6-right_center.

So we decide to use this correspondent direction vector computed from above formula as middle finger direction. Concrete algorithm is to be shown in the next diagrams. Denote the vector we found as V (x,0,z).

b). Angle_z

Rotate around z-axis(0,0,1) means keeping middle finger parallel to z-axis and sway hand left and right. In this rotation process, we expect the virtual lion puppet rotate at the same angle as our hand also.



Similar to the method we compute Angle_y, we set up an axis as benchmark at first. Then compute the angle between it and another vector, this angle is Angle_z. In this method, we choose x-axis(1,0,0) as our benchmark, and use vector (dist(right_thumb,right_pink),right_thumb_y-right_pink_y,0) to compute difference angle with

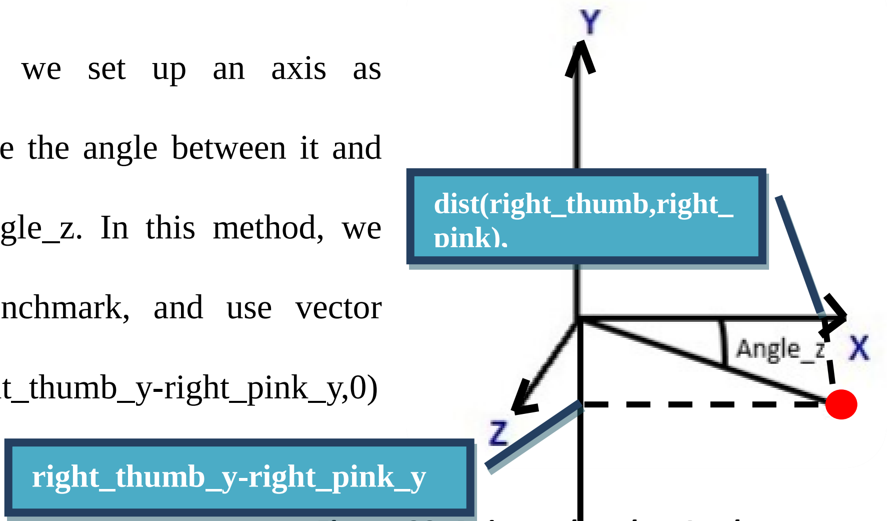
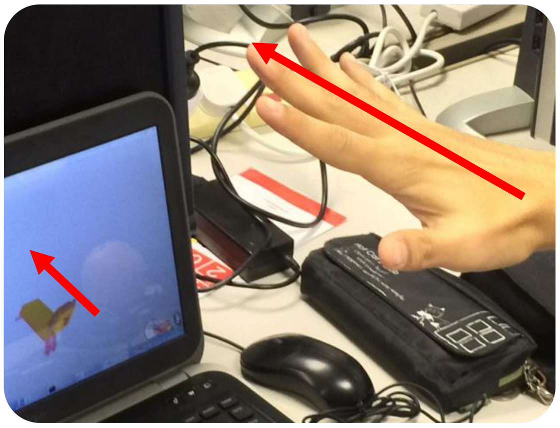


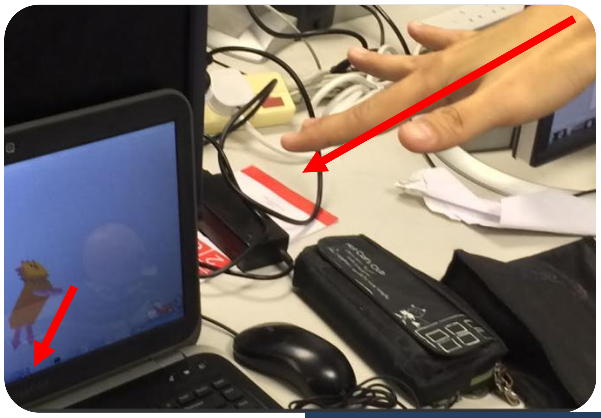
Figure 32: Points related to Angle_z

it. The result angle is Angle_z then. The dist( ) function mentioned before is to work out the distance between two points as input parameters. Concrete algorithm is the same as Angle_y's algorithm.

c). Angle_x

Rotate around x-axis is still similar to the former two conditions. Right hand rotate around axis which is parallel to hand back and orthogonal to finger bottom. We also have the expectation over that virtual lion puppet. it should rotate the same angle as our hands do. After rotation, the lion puppet face should face to upwards direction outside the computer screen.





Follow the same pattern of the above two algorithm, set negative z-aixs(0,0,-1) as benchmark, use same formula right_thumb*0.4+right_ring*0.6-right_center to compute desired angle. However, in this process, the result of that formula is not the vector we

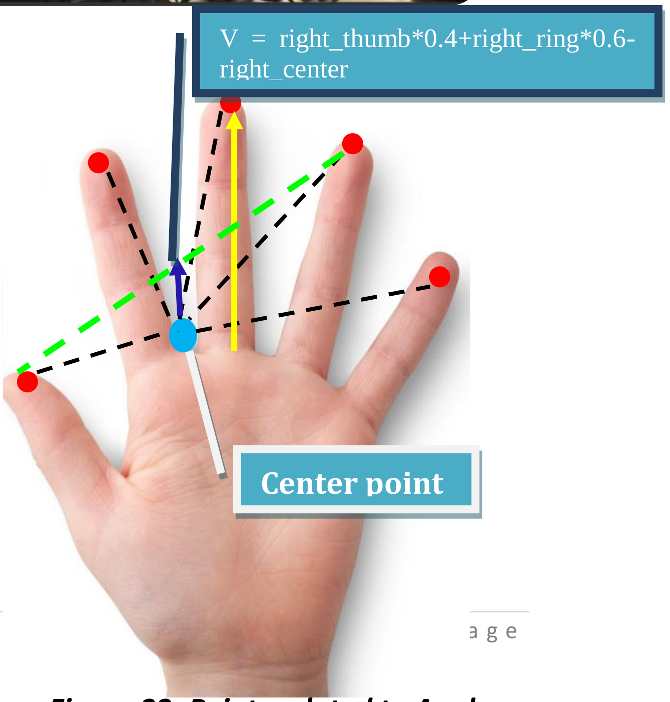


Figure 33: Points related to Angle_y

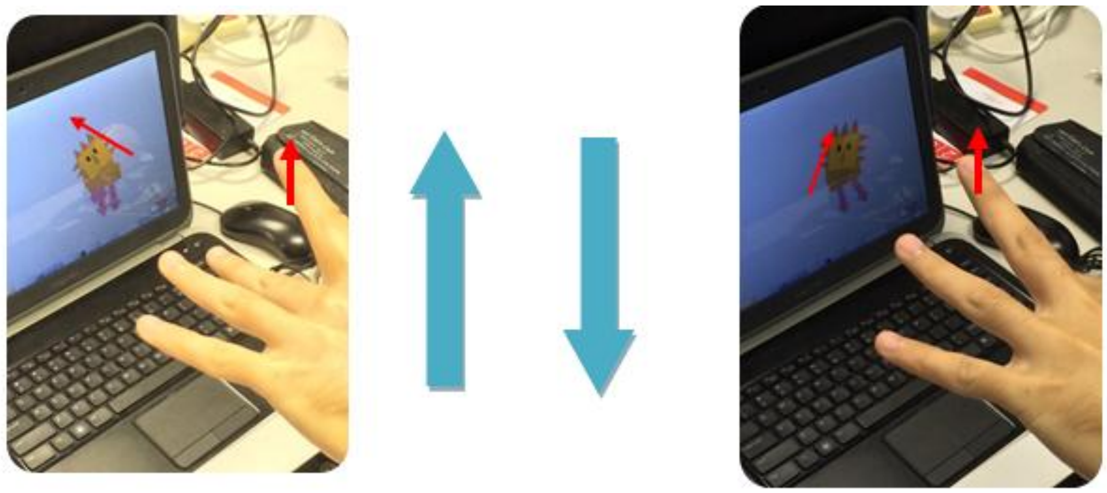
denoted V(x,0,z) but V(0,y,x).

Eventually we use the same algorithm to get final answer.

All above is about how to realize right hand controlling over the whole puppet lion's body transformation. The reason we choose to use right hand to control the lion is because it is conform to user habit in reality on the one hand, and one hand control will reduce instability caused by fingers' overlapping due to the fact that those fingers are harder to interfere with one another compared to use to hand to control at the same time. In our first design, we tried to add some other animation control on right hand fingers also, but it turns out that it will dramatically influence string lion puppet's stability. So we decide to use left hand to do those special animation, which we will mention later. Although there is still some distance between this virtual string lion puppet performance and real string lion puppet performance, it has already reach a status that lion can be transformed following real hand movement stably and reasonably.

This virtual puppet lion's leg rotation functionality is actually related to whole lion body's rotation functionality, which means legs' rotation angles are the same as the whole body's. The only difference is they have different rotation direction or rotate axis. This setting is based on string lion puppet's construction consideration but not the hand data manipulation problem.

This virtual string lion puppet's head move is controlled by left hand middle finger and index finger together. The lion will sway its head if we move those two fingers. This case can also refer to former angles' computation method. Because lion head only need to rotate around z-axis, thus we only need to computer the angle between left hand index finger and ring finger. We can directly use the angle between ‘left_index - left_middle’ vector’s component in (x,y,0) direction and positive x-axis direction.



In conclude, out design idea about general transformation is that we use one point or multiple points’ linear combination result as parameters of translation. As for rotation, we will first try to compute rotate axis (x,y,z) and rotate angle at the same time, if it is not possible, we will consider rotating three times and compute rotate angle respectively. It is just the algorithm we designed until now. If we can find better solution, we will further improve our algorithm.

Term 2 Implementation

As for translation, as we explained before, instead of directly mapping hand's position (x_h,y_h,z_h) to string lion puppet's position (x_p,y_p,z_p), we mapped average hand position of latest 10 frames to string lion puppet's position for the purpose of real string-drag movement simulation. Then the lion puppet would follow but lag behind hand movement, just like string drag movement in real life.

As for rotation, we inherited last semester's method. We compute angle_x, angle_y and angle_z based on our algorithm, and rotate the lion puppet based on 3 angles individually. The advantage of using Unity3D compare to our last semester's work in Openframeworks is that the space coordinate would not rotate when we do rotating on our model. Thus this method reached better performance this semester. At the same time, we did some adjustment on rotation pattern. We get rid of leg rotation is controlled by body which was a shallow design if we think of real life. Also we decide to use finger to control head rotation just like drag some switch to control head move in real puppet performance.

The primary movement control hand is still right hand.

5. Gesture and mapping

Term1 Implementation

Besides trivial raw data information, Intel Perceptual Computing SDK also offered gesture recognition function. As we introduced in chapter2, it can distinguish several basic gestures, such as ‘Big five’, ‘Peace’, ‘Thumbs up’ and ‘Thumbs down’. Also it has ability of simple motion recognition, like ‘Swipe right’, ‘Swipe left’, ‘Swipe up’, ‘Swipe down’ and ‘Zoom’ action. Gesture recognition is more stable than motion recognition, because it hard to judge whether it is a intended move or unconsciously, say if a human swiped right and want to swipe right again, then he may wan to put his hand back to left side; then if his movement path is similar to swipe left, then the SDK of high probability will consider that is a swipe left motion. So we decide to use static gesture recognition as supplementary function.

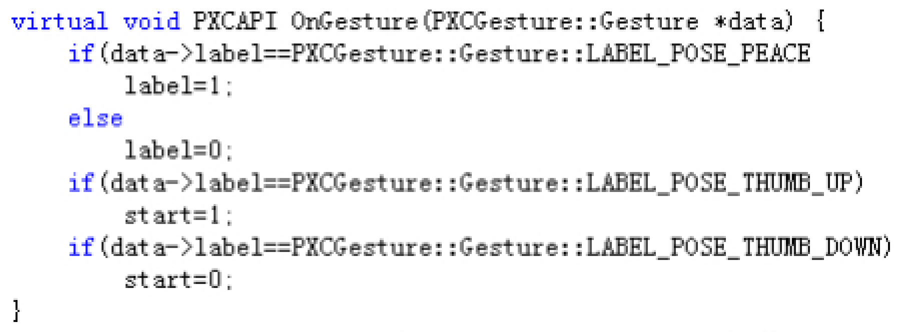
We added a gesture control on left hand. In normal situation, lion eyes will show like ‘0 0’. Instead of ‘0 0’, when user use left hand to make a ‘peace’ gesture, they eyes will become like ‘> < ’. Once user released gesture, lion puppet eyes will restore to normal situation.

Because we haven’t do very life-like rendering, thus the lion eyes are just represented by simple primitives. After rendering, we may use more realistic control method to control lion eyes.

Details of gesture recognition application is as follows.

In Intel Perceptual Computing SDK, besides depth and position information of necessary fingers and palm, we can also get gesture information directly. Intel Perceptual

Computing SDK use internal algorithm to keep detecting whether user make special gesture towards camera or not. Once some special gestures detected, SDK will invoke function OnGestrue() and pass into a structure which stored gesture information. Then we can do manipulation on this gesture.



```
virtual void PXCAPI OnGesture(PXCGesture::Gesture *data) {
    if(data->label==PXCGesture::Gesture::LABEL_POSE_PEACE
        label=1;
    else
        label=0;
    if(data->label==PXCGesture::Gesture::LABEL_POSE_THUMB_UP)
        start=1;
    if(data->label==PXCGesture::Gesture::LABEL_POSE_THUMB_DOWN)
        start=0;
}
```

At this stage, we did manipulation on gesture with label named 'LABEL_POSE_PEACE'. By judging whether at current frame 'LABEL_POSE_PEACE' gesture is detected, we will draw different eye pattern on that virtual puppet lion.

Intel Perceptual Computing SDK dealing hand trivial information data and gesture data in a stream line's process way. Thus they can be processed in synchronized manner. So two hands' movement or gesture making will not have influence on each other. Thus when this virtual lion puppet is moving on the way, its eyes can also be controlled at the same time. It is worth noting that when we do that 'Peace' gesture using left hand, the other finger's information of left hand would be not accessible which is because they will be lost tracking when doing 'Peace' gesture. So if we need to use the other left hand fingers, we will add some restriction on it related to 'Peace' gesture.

Term2 Implementation

Because we have already set up a gesture control application prototype last semester, this semester's implementation is a process of adjusting and bettering what we have done last semester. Connect to what we see in real life, string lion puppet's mouse control is a vivid behavior which will attract people's attention. So we decide to add mouse control this term. At first, we considered using simple rotation and transformation on Unity3D to simulate string lion puppet mouse open and close. However, the result is not so good. It looks stiff and factious. So we decide to add extra animation clip on it. We built that mouse close-and-open animation clip using Blender. This short animation section would be triggered by a specific gesture 'peace'. Using Intel Perceptual Computing SDK, when we read the gesture property is 'LABEL_POSE_PEACE', it means that a peace gesture is detected and we trigger the correspondent animation. Besides animation, for diversity consideration, we added background music and event music on our project. Background music will be playing in the whole application running process while event music would only be played when specific event triggered. Instead of detecting all user's gestures, we just focus on the gestures we need in each frame. Once target gesture is detected, we make relevant actions. To a certain degree, this method improved application efficiency. Scene switch and application quit is also implemented by gesture control. Thus gesture control is not just limited in puppet behaviors control, it run through the whole application process.

6. Physics environment construction

As we mentioned in the first beginning, our focus on this semester is about physics attributes adding. We want to add physics elements on our lion puppet and also the circumambient environment.

This part is mainly realized in Unity3D. First, we made the imported string lion puppet a rigid body at first. Then we add gravity and collision detecting on it. Then if the string lion puppet hit a rigid body on the scene, there would be a correspondent reaction on both object just like what we see in real life. Of course, there are more than one objects on our main application stage. We made them all rigid body. Also, for those string-like item like the string on the stage and lion puppet's legs, we combined fixed joints and hinge joints to make them move smoothly and reasonably.

7. Optimization

This part talks about how we deal with those Intel Perceptual Computing SDK limitations mentioned before. Main limitations are as I mentioned before, the SDK can not distinguish left hand and right hand, finger lost-tracking and stability issues. In order to avoid that limitation, optimize our project, we add lots of codes on the basic framework of our program to optimize its performance.

Dealing with left-right hand indistinguishability

For the problem that the Intel Perceptual Computing SDK cannot distinguish left and right hand, although in our design we set the right hand to control the string lion puppet, if we don’t deal with this problem, in an actual execution, Intel Perceptual Computing SDK may regard the left hand as the right one and vice versa. So in fact the user may use his/her left hand to control the string lion puppet’s transform. Even Intel Perceptual Computing SDK distinguish two hands correctly at start, due to some dropped frames or lost tracking problems, Intel Perceptual Computing SDK’s distinguish may suddenly goes wrong during the execution. It makes our program unstable and hard to handle, to solve this problem, we have tried two methods.

The first one is to make up the problem according to its cause. Since Intel Perceptual Computing SDK in fact regard the first hand it successfully tracked as left hand, so this method is not involve any technical things, that is, remind use each time when they execute this string lion puppet program that put their left hand into camera first and then right hand. With this method, the program runs as our expected from beginning. But when lost frame or lost tracking occur, Intel Perceptual Computing SDK may still make a mistake in distinguish left and right hand, so this method is not a complete solution but only a make-up method.

The second method is the one we are using now, that is, for each frame, before we call functions in data mapping module, we add an conditional judgment to judge whether Intel Perceptual Computing SDK distinguish our hands correctly or not. The specific judgment method is quite simple; it is just comparing the center point's x coordinate of the two hands. If right hand's x value is larger than left hand's, it means Intel Perceptual Computing SDK's distinguish is correct, then the following function calls are as general. But if the left hand's x is larger, means Intel Perceptual Computing SDK make a wrong distinguish, and then in the following we exchange the function calls of left hand and right hand. In this way, for each frame, as long as Intel Perceptual Computing SDK tracks two hands successfully, the right hand always controls string lion puppet's transform.

At present, the second solution works best. It can solve almost all indistinguishable problems on left and right sides. Only in some extreme cases, such as user cross his hand on purpose or two hand overlap too much, Intel Perceptual Computing SDK may show indistinguishable problem. Thus it can eliminate this problem caused by unintentionally overlap.

Dealing with lost tracking

lost tracking is a kind of knotty problem. When lost tracking problem happened, the information read from each fingers will have saltation. Because the string lion puppet position is based on hand or fingers position, it would manifest a suddenly jump on lion puppet, bring a very unhappy experience to user.

As for this problem, we haven't found a very effective way to avoid it. Because when lost tracking problem happened, Intel Perceptual Computing SDK still can read some information, although it is actually some other finger's data. No lost tracking warning would make us reluctant on dealing with this problem. What information we can do about lost tracking problem is to judging whether the finger data we got is the same as the other. If there are two or more fingers has exactly the same position information, it means that lost tracking happened.

Our solution to this problem is do lost tracking judging before invoke data mapping module. If we detected lost tracking, we will not continue to invoke functions in data mapping module. Instead, we keep lion's position at the same position as before. In fact, this solution is just relieved the problem but not eliminate it fundamentally. Because if the lost tracking problem happened, the lion may stay at the former position and can not move, it will also affect user experience.

May be this problem can be easily solved if Intel Perceptual Computing SDK have remarkable upgrading. But we can not put our hopes on that. We also conceived of a theoretic solution, it is that we keep recording each fingers' position information all the time, have a statistics record. Then when lost tracking happened, we can estimate the should-be position based on former statistics record and some algorithm and set the untracked position as the lions position at that moment.

Improve stability.

So called stability problem in Intel Perceptual Computing SDK means when tracking some fingers, there will be some slight random error happened. It happened seriously at the beginning of this semester. It turns out this situation turns better when Intel updated several version of Intel Perceptual Computing SDK. At that time when the instable problem happened, we figure out a method that record the most recent ten frames' data and have their mean value as this time's value to reduce the random error.

Eliminating distraction caused by glowing object

Problem-caused glowing objects include those shining bulbs, electronic device screen and other high brightness objects shown in the detective range of Creative Camera. The illuminated items would have impact on hand and gesture information collection that made hand locating become inaccurate even wrong. The interesting thing is that this problem did not show quite often last semester while do appears several times this semester. It might happened because of hardware breakdown or because of SDK updating bugs. No matter what the real reason is, we do not have too much to do about it. So what we can do about this problem now is to avoid strong illuminated objects shown directly in front of the camera.

Chapter4: Contribution

My jobs in the Final Year Project could be summarized into exploring, study and creation. At the very beginning, when my partner and I have almost zero understanding of this new software development kit, I did a lot of testing and information collecting on the internet to make the Intel Perceptual Computing SDK becomes our best friend. I read official documentation and reflected important tips to my partner to help him do algorithm decision which is his main job. Also, I set up a whole working framework to utilize mathematic model generated by my partner to realize puppet control. Since my partner is good at raw data manipulation part, I have to set up reasonable target for him to get desired result and make application functional and query the expected raw data with the help of SDK. I wrote code prototype for my partner to expand and add his own idea.

After the project frame settled, I took charge in modeling part. Because our objective is not building a very delicacy puppet model on the first semester, I need to found a easy to manipulate modeling method to make out project has the best working effect. After cross check different simple computer graphics tools, I chose Openframeworks as our modeling tool of the first semester. After basic prototype has been established, I also did several adjustment on that simple model to make it have the best demonstration result.

The whole task in the second semester is slightly different from the task in the first semester. Because we have done the most of algorithm and mathematical model design part last semester, this semester's task becomes how to make a complete application and perfecting our application. I still be the pioneer. I tried different platforms and discussed with my partner to choose a best suitable one, which is Unity3D. After that, I tried different modeling tools and decide to use Blender as our main modeling tool. I also took charge of model rendering, texturing and animation making part. Since I mainly take charge of modeling part, stage and lighting are also my job. Besides, audio and background music adding is also done by me.

It is the first time I work independently on a software project. Those so called projects I did before is just another name of a course assignment. I learned a lot from it. It is a great valuable experience for me. In this year, I learned a lot not only in specialty area but also in many other areas. Programming skills are enhanced while divergent thinking ability is also improved.

Doing a software project means start everything from the very beginning. Obstacles and errors show up all the way end. Thanks to this one-year's hard-working, I become more strong in both study and daily life. The final year project really benefits me a lot.

Chapter5: Conclusion

Conclusion

The purpose of our final year project is to virtualize string lion puppet in reality to computing world based on Intel Perceptual Computing SDK. Through this FYP process, we should learn how to reconstruct behaviors in real world in virtual computing world. At the same time, enhance our skills in software developing and programing.

What have been done this year

1. Gesture information collection and recognition
2. 3D lion puppet construction and rendering.
3. Virtualized model (string lion puppet) behaviors pattern design.
4. Different dimension movement mapping.
5. Physics engine utilization.
6. System (Intel Perceptual computing SDK, Creative Camera) inherent limitation study and remedy.

At the end of last semester, the lion puppet we drawn is a composition of several 3D or 2D primitives; the lion puppet's move is restricted to certain degree and the movement and transformation we made is trivial, which makes the move simulation is awkward and lacking of reality.

Now, when it comes to end of this semester, we have already constructed a highly realistic string lion puppet model with texture. Although the shape is not perfect, the physics part is quite reasonable. After we add physics engine into our string lion puppet model, not only puppet control becomes more flexible also simulation result is much more close to real string lion puppet.

When first start this project in the beginning of this academic year, we are totally new comers of computer graphic and perceptual computing world. We began our study from the very primary level, until now we can simulate a simple string lion puppet on the screen; we went through a long journey of studying and improving.

In the study process, we have encountered lots of difficulties, including the lack of familiarity with development tools, inherent defects in SDK, confusion on topic decision, deficiency of relevant knowledge. Faced with those problems, we search for solutions patiently and aggressively, start from the very beginning and progress step by step. As for Intel Perceptual Computing SDK's inherent defect, we tried to figure out suitable algorithm to reduce the inconvenience caused by them in programming.

It is a great experience for us to use Intel Perceptual Computing SDK and Creative Camera to do application development. Creative Camera is a small and exquisite device with powerful functionality. Intel Perceptual Computing SDK offered us useful functions for raw data processing. Although both of them have certain degree's weakness, there are still young and promising hardware and software tools. If they developed well, there should be many relevant application in the future.

Chapter6: Acknowledgement

We would like to thank the supervisor of our project, Professor Michael Lyu, who gives us suggestions for our project and advice on how to write a good report and make a good presentation. Professor Lyu arranges a one-hour meeting every week for us to do progress feedback and give us further suggestion.

We would like to thank technical staff Mr. Edward Yao in VIEW Lab who provides us with facilities and technical support. He spent a lot of time helping us to do testing and gives us lots of valuable advices.

Chapter7 References

[1] Perceptual computing - Wikipedia, the free encyclopedia:

http://en.wikipedia.org/wiki/Perceptual_computing

[2] Natural user interface - Wikipedia, the free encyclopedia:

http://en.wikipedia.org/wiki/Natural_user_interface

[3] Unity (game engine) - Wikipedia, the free encyclopedia:

http://en.wikipedia.org/wiki/Unity_(game_engine)

[4] Creative* Interactive Gesture Camera Developer Kit:

http://click.intel.com/intelsdk/Creative_Interactive_Gesture_Camera_Developer_Kit-P2061.aspx

[5] Creative Senz3D - Web Cameras - Creative Technology (Hong Kong):

http://hk.creative.com/p/web-cameras/creative-senz3d

[6] William T. Freeman and Michal Roth: "Orientation Histograms for Hand Gesture Recognition", MITSUBISHI ELECTRIC RESEARCH LABORATORIES CAMBRIDGE RESEARCH CENTER:

http://ece.ucsb.edu/~parhami/rsrch_paper_gdlns.htm

[7] James L. Crowley: "Finger Tracking as an Input Device for Augmented Reality":

http://venus.inrialpes.fr/Prima/Homepages/jlc/papers/FG95.pdf

[8] Ross Cutler and Matthew Turk: "View-based Interpretation of Real-time Optical Flow for Gesture Recognition": http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=670984

Figure1:

http://www.letsgodigital.org/images/artikelen/816/touch-screen-photo-frame.jpg

http://www.blogcdn.com/www.engadget.com/media/2010/07/smartpad-tablet.jpg

http://www.onbile.com/info/wp-content/uploads/2013/08/mobile-touch-screen-keypad.jpg

http://www.pagegadget.com/wp-content/uploads/2012/11/top-best-touch-screen-computer-2012-1024x768.jpg

http://blog.eoffice.net/wp-content/uploads/2011/07/touch-screen-tv-2.jpg

Figure2:

http://cdn.physorg.com/newman/gfx/news/2013/intelspercep.jpg

Figure3:

http://oaklandaegis.com/wp-content/uploads/2012/04/lion-dance-dragon-1024x971.jpg

http://www.chinasprout.com/store/media/AND009L01.jpg

Figure4:

http://www.crystalinks.com/pinocchio.jpg

http://us.123rf.com/400wm/400/400/anyka/anyka1103/anyka110300008/9019737-hand-cutting-the-strings-of-a-puppet-giving-it-freedom.jpg

http://goparallel.sourceforge.net/wp-content/uploads/2013/02/Untitled.png http://i01.i.aliimg.com/wsphoto/v0/1281132665/laptop-computer-15-inch-font-b-Lenovo-b-font-ThinkPad-font-b-W530-b-font-2438.jpg

Figure5:

http://bpa.atech.edu/team3/pictures/kinect-sports-.jpg

https://www.mln.com.au/img/uploads/images/products/xbox-kinect_angle_view.jpg

Figure12: http://download-software.intel.com/sites/default/files/article/325946/secured-perc-productbrief-q1-2013-327943-003us.pdf

Figure16: http://click.intel.com/intelsdk/Images/Product_Images/CreativeCamera_4625fd50-8c95-41e9-acfa-79b1381a4393.png

Figure17:

http://www.everythingusb.com/images/list/creative_senz3d_news.jpg

http://g-ecx.images-amazon.com/images/G/02/uk-videogames/2010/Xbox/kinectfront-lg.jpg

http://hk.creative.com/p/web-cameras/creative-senz3d

http://msdn.microsoft.com/en-us/library/jj131033.aspx

Figure26:

http://software.intel.com/sites/landingpage/perceptual_computing/documentation/html/worldcoordinatesystem.png

Figure28:

http://software.intel.com/sites/landingpage/perceptual_computing/documentation/html/

Figure29, 30, 31, 32:

http://eofdreams.com/data_images/dreams/hand/hand-01.jpg

Table3:

http://download-software.intel.com/sites/default/files/article/325946/secured-perc-productbrief-q1-2013-327943-003us.pdf

Table4,5,6: http://stimulant.io/depth-sensor-shootout/