

**Nagoya Institute of Technology**

**Sato Laboratory**

**EFREI**

2011 - 2012

**3D Natural User Interface using Kinect**

**LUCCIN Karim – LUONG Kevin**



Table of contents

[I. Acknowledgements 4](#_Toc320024162)

[II. Introduction 5](#_Toc320024163)

[III. Nagoya Institute of Technology 6](#_Toc320024164)

[1. Nagoya 6](#_Toc320024165)

[2. NIT presentation 7](#_Toc320024166)

[3. Sato Laboratory 8](#_Toc320024167)

[IV. Motivations & Background 10](#_Toc320024168)

[1. Kinect Project 10](#_Toc320024169)

[2. Background on Kinect 10](#_Toc320024170)

[V. Objectives & Realizations 11](#_Toc320024171)

[1. Objectives 11](#_Toc320024172)

[2. Used Methodology, Technologies and Tools 11](#_Toc320024173)

[a. Development method 11](#_Toc320024174)

[b. GANTT chart 13](#_Toc320024175)

[c. Control 14](#_Toc320024176)

[3. Solution 14](#_Toc320024177)

[a. Description 14](#_Toc320024178)

[b. Possibilities offered by the Kinect 16](#_Toc320024179)

[c. Limitations of the Kinect 17](#_Toc320024180)

[4. System Architecture 18](#_Toc320024181)

[a. KinectBrowser 19](#_Toc320024182)

[b. Input.Kinect, Input.Mouse & Interaction 19](#_Toc320024183)

[c. ImageProcessing 19](#_Toc320024184)

[d. D3D 19](#_Toc320024185)

[5. Theory 20](#_Toc320024186)

[a. Algorithm to recognize hands 20](#_Toc320024187)

[b. CPU vs. GPU 21](#_Toc320024188)

[6. Technical 22](#_Toc320024189)

[a. Running in Real Time 22](#_Toc320024190)

[b. Implemented algorithm for the hand recognition 23](#_Toc320024191)

[7. Evaluations of the system 25](#_Toc320024192)

[a. Definition 25](#_Toc320024193)

[b. Protocol 25](#_Toc320024194)

[c. Results 26](#_Toc320024195)

[1. Accuracy evaluation 26](#_Toc320024196)

[2. Processing speed 30](#_Toc320024197)

[3. Word input speed 31](#_Toc320024198)

[VI. Results 34](#_Toc320024199)

[VII. Conclusion 35](#_Toc320024200)

[VIII. Glossary 36](#_Toc320024201)

[IX. References 37](#_Toc320024202)

[X. Appendix 38](#_Toc320024203)

[1. Screenshots of the system 38](#_Toc320024204)

[2. General Specifications Document 40](#_Toc320024205)

# Acknowledgements

We would like to acknowledge the following people for their support and assistance with my internship.

The special thanks go to M. SATO Jun, our helpful supervisor. He guided and helped us with the various problems encountered through this project.

We would also like to thank the students of the Sato Laboratory who helped us greatly during our daily lives in Japan.

# Introduction

This project report is submitted for the M2 Student Exchange in Nagoya Institute of Technology, Japan.

We were given the opportunity to do a Research & Development project in Japan for 6 months. So we have decided to work with the Microsoft Kinect and have joined the Sato Laboratory.

In this report, we will first of all present the environment (Nagoya Institute of Technology and Sato Laboratory) and the motivations as to why we have decided to work on the Kinect before detailing the used methodology, works and results of the project. We will then evaluate our system and then discuss what we have learnt from this past 6 months.

# Nagoya Institute of Technology

## Nagoya

For this Exchange Student project we joined the Nagoya Institute of Technology (or NIT). As its name implies, this school is located in Nagoya: the 4th biggest town in Japan.

Located on the Pacific coast in the Chūbu region on central Honshu, it is the capital of Aichi Prefecture and is one of Japan's major ports. It is also the center of Japan's third largest metropolitan region, known as the Chūkyō Metropolitan Area.

Despite being one of Japan’s major ports, Nagoya has a large number of state and private colleges and universities such as the Nagoya University, Toyota Technological Institute and Nagoya Institute of Technology.



Image Nagoya

## NIT presentation



Image 2 Entrance of NIT

The Nagoya Institute of Technology is a public highest-level educational institution of science and technology located in Nagoya, Japan. Nagoya Institute of Technology was founded on 1905 as Nagoya Higher Technical School, renamed Nagoya College of Technology in 1944 then merged under the new educational system with the Aichi Prefectural College of Technology to be refounded as Nagoya Institute of Technology in 1949. Finally in 2004 it was refounded as Nagoya University Corporation Nagoya Institute of Technology.

This university holds many departments and laboratories:

*Faculty of Engineering*

* Life and Materials Engineering
* Environmental and Materials Engineering
* Mechanical Engineering
* Electrical and Electronic Engineering
* Computer Science
* Architecture and Design
* Civil Engineering and Systems Management

*Graduate School of Engineering*

* Materials Science and Engineering
* Engineering Physics, Electronics and Mechanics
* Computer Science and Engineering
* Architecture, Civil Engineering and Industrial Management Engineering
* Techno-Business Administration
* Frontier Materials
* Scientific and Engineering Simulation

*Educational Research Centers*

* Quality Innovation Techno-Center
* Ceramics Research Laboratory
* Research Center for Nano-Device and System
* International Center for Automotive Research

## Sato Laboratory

For this project we joined the Sato Laboratory. This laboratory was created in 1998 and is specialized in Computer Vision and Graphics: in particular 3D reconstruction from camera images, multiple view geometry (which is geometry composed of multiple cameras images).

The Sato Laboratory has worked on many kinds of projects:

* ITS (Intelligent Transport System)

The ITS is a system that makes vehicles “intelligent”:

With the help of cameras, the driver uses image information to assist drive and reduce accidents. Multiple cameras are put on the car to prevent dead angles.

This system can also create virtual images to prevent rain on the view mirrors and create a clear image.

* Human Computer Interaction

Using the Kinect they make applications where the user moves the cursor on the screen by moving the eyes.

* Mixed reality, augmented reality

Mixed reality is a 3D reconstruction of the scene using projectors.

La méthode au NIT c’est projeter des lumières codées sur des objets et le résultat c’est que les couleurs dépendent de la profondeur des objets.

* Research on special camera that can work on 4D information:

Image and light direction. Instead of having images in the form of I(x, y) we have I(x, y, θ, ω) where θ and ω are angles to the light sources.

This laboratory offers all the materials necessary for the students to do their research. A student room and a computer room are accessible by everybody any day/time.

# Motivations & Background

## Kinect Project

For this project we both instantly decided to work on the Kinect as we are interested in new kinds of experiences for the user; it can go from games such as we can see on Xbox 360 or devices like touch devices such as Microsoft Surface.

Another reason is that right now there is “boom” of touch devices. These devices are very popular such as smart phones or touch tables… Since the general public is keen on these devices it is the perfect moment to try and develop the next step to the touch user interface: the Natural User Interface or touch less devices.

## Background on Kinect

As I said earlier, we are right now in the “touch screen era”. Everybody has or has used touch devices such as smart phones, touch tablets… Companies produce more sophisticated touch devices.

People like to experience new ways to interact with computers or games. When the first “Kinect” project was revealed to the general public, which was called Project Natal at that time, people were amazed at the possibilities offered by this new gaming experience. That is why we turned our eyes to the Kinect as it was seen as the revolutionary gaming experience.

As of now the general public only sees the Kinect as a gaming device but more and more researchers do develop some new applications using the Kinect.

# Objectives & Realizations

## Objectives

We have set 2 objectives for this project:

* First we would like to see the various possibilities and limits offered by the Kinect and see how we can assemble the different devices to make a natural user interface.
* Secondly we would like to create some applications that use the potential of the Kinect such as a Web browser or a Media Center.

The goal of this project is to develop a natural user interface with Kinect. This means that the user will be able to interact with the computer via hand/arm motions. We will develop a web browser based on the design of Internet Explorer 10. This browser will be specifically adapted to Kinect using a Metro style. At the end the web browser will be able to understand the various input devices such as Kinect, the mouse and the keyboard.

## Used Methodology, Technologies and Tools

### Development method

The method used for this project is Agile. The main advantage of this method is to work by iterations. We first begin by working on the core of the system which has the least possibility of being modified through the project. At the end of iteration a meeting will be done to validate the solutions found.

The project is divided into 7 parts:

* Planning of the project

During this phase we first think of the functionalities offered by our system. By doing so we can then establish a Gantt chart and estimate the time needed for the development.

* Analysis

As it is a new technology to us we will have to study how the Kinect works and learn some new programming languages. During this phase we will also define the general and technical specifications that we will develop later.

The purpose of this phase is also to see the feasibility of the project by developing some prototypes or proof of concept.

* Conception

For this we will focus on the architecture of the system. From the prototypes done earlier we can see the different modules composing our system. We will use UML to draw the various diagrams such as class diagram, database diagram…

During this conception phase we will also sketch some rough screens of the user interface.

* Implementation

Now that we have decided the architecture of the system we will now implement the solution.

For the implementation phase it will be divided into 3 sprints:

* The first sprint will be on the data acquisition. By the end of this sprint we will have a functional module for gathering data from the input devices.
* The second sprint will focus on the algorithm of the movement recognition and some screens of the user interface. At the end of the sprint we will have some rough screens of the final user interface. By the end of this sprint each of us will work separately, one person will be in charge of the implementation of the movement recognition algorithm while the other will be working on the user interface.
* The last sprint will be the final version of our system. Everything should be implemented and functional at the end of the sprint.

### 

* Tests

For this phase we will test every functions of our system. The purpose is to make sure that there is no bugs anymore and that the project is fully working.

* Final check

In this last phase we check the documents that we have written for the project, clean the code, check that the application is fully functional…

### GANTT chartC:\Users\Kevin\Documents\Projet Japon\Gantt.png

Image GANTT chart

### Control

* Meetings

A weekly meeting with our teacher is done to see how the project is progressing. This meeting is also used to talk about the difficulties encountered in the project. During these meetings the teacher also indicates what could be improved on the current system and what would be a good idea to implement.

At the end of each phase/sprint, a little presentation is done to show what has been done so far.

* Bug tracking

Every bug encountered in this project will be written down as “not solved” with a small description of the bug and put in the Git versioning. Every time someone solves a bug, he has to push in Git the solution as well as a description on how he solved it.

## Solution

### Description

The project represents a Natural User Interface Web Browser using Microsoft Kinect. By using his hands as cursors the user could experience a new way to navigate on Internet.

But for this system to be fully enjoyable and useful for the user some requirements are necessary.

The system must be responsive; having time lags between each click would ruin the user’s experience. That is why we developed this project with the constraint of making the system run in Real Time (30 frames per second).

We originally planned to put the web browser into 2D Window System. But as it turned out quite bland, we have decided to switch to 3D Window System. Since the project is a Natural User Interface, it is very important to appeal to the user and having 3D Window System offers more possibilities in terms of User Interface than 2D Window System.

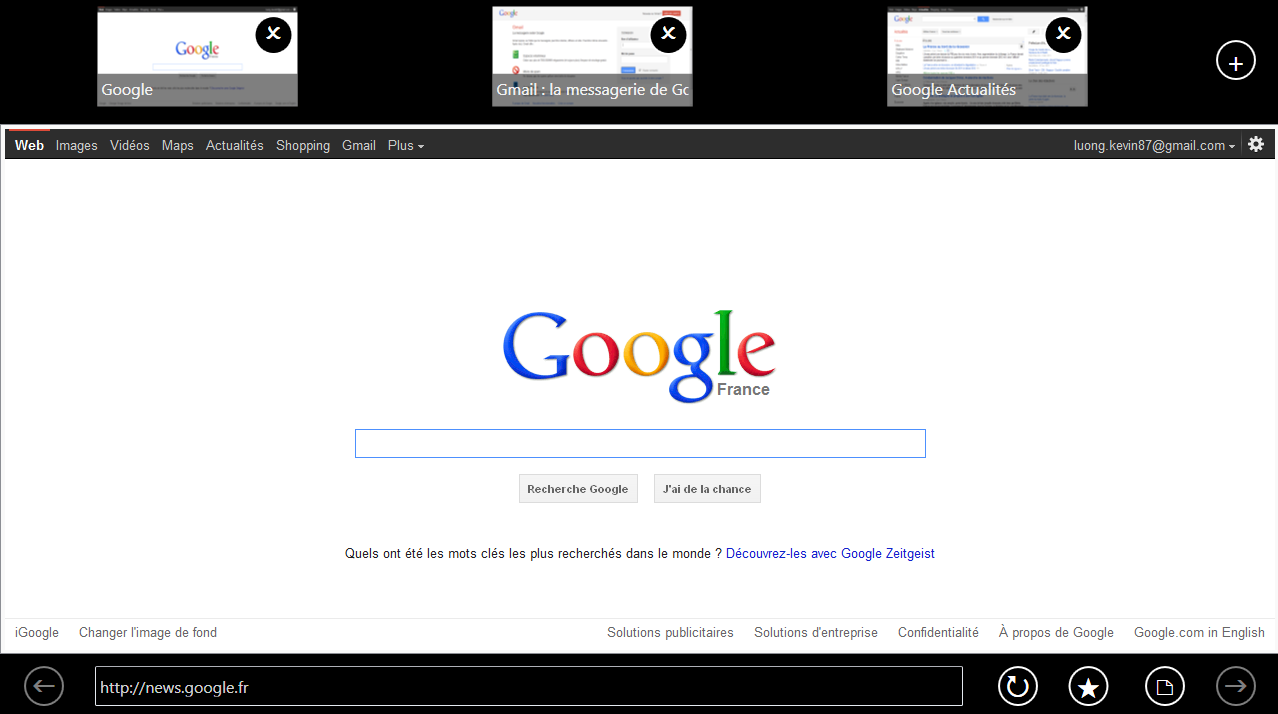


Image 4 2D Window System

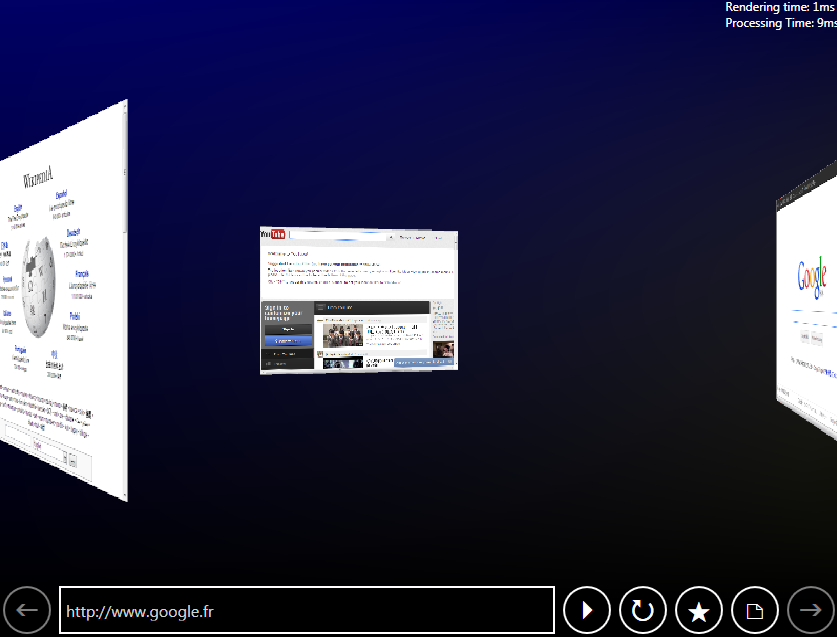


Image 5 3D Window System

### Possibilities offered by the Kinect

The Kinect is a motion sensing input device by Microsoft. Based on the design of a web cam, it enables the user to interact with applications using gestures or spoken commands. It includes 3D depth sensors, RGB camera and a multi-array microphone.

In 2011, Microsoft released a Software Development Kit (SDK) to allow developers to write Kinect Applications. This SDK allows us to access raw streams of data captured by the cameras of the Kinect in the form of 4 data features:

* Depth data
* Color data
* Skeleton recognition
* Voice recognition

### Limitations of the Kinect

During our first phase of our project we had to identify the various limitations that the Kinect has. From this phase we have identified 3 main limitations:

* Distance: Kinect only recognizes shapes and objects that are at least 1 meter away from the camera.
* Accuracy: Some noises appear on the data output which makes the accuracy a little instable.
* Recognition: you have to be standing up first for the Kinect to recognize your skeleton. When the user is sitting down the Kinect won’t be able to output your skeleton correctly.

Another problem with the recognition is the ambient lighting. Because the lighting differ from each place and moment the Color data given by the Kinect can be false. It might not recognize some elements because it is too bright or too dark.

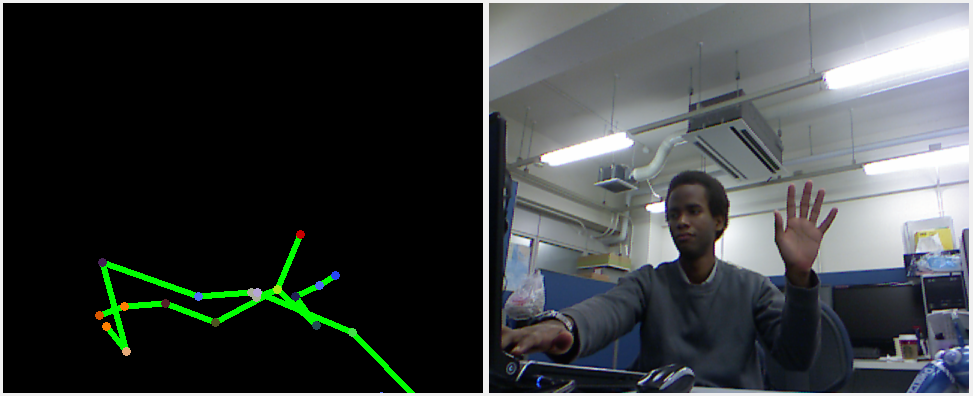


Image 6 Skeleton recognition problem

## System Architecture

Image Class Diagram

### KinectBrowser

KinectBrowser is the name of our system. This part of the system is the main core: it initializes all the elements necessary for the system such as the Web Browser, the main window and the keyboard.

### Input.Kinect, Input.Mouse & Interaction

Since we wanted our project to both utilize the Kinect and the Mouse it is natural to have our system detect these two entry devices. As its name implies Input.Kinect gets raw data from the Kinect whereas Input.Mouse gets the data from the mouse device.

At this point Input.Kinect gives all the data: Depth, Color and Skeleton data. These data are then filtered and treated by the Interaction.dll and the ImageProcessing.dll. From the limitations we have found earlier we have decided to only use the Depth data.

The Interaction.dll is used to recognize the hands of the user as well as the gestures done by the user.

### ImageProcessing

ImageProcessing.dll is used to treat data from the Kinect on the graphical card. Since our objective is to make a system that runs in Real Time this processing on the graphical card is mandatory.

When the system gets the raw Depth data from the Kinect, we couldn’t determine what part of the data represents the hands of the user. ImageProcessing.dll tracks the regions (or blobs) on the screen and determines where the hands of the user are. This part will be explained in the “Technical” part of the report (page 22).

### D3D

D3D.dll or Direct3D.dll is part of Microsoft’s Application Programming Interface (API). It is used to render 3D graphics in applications, allow applications to run in full screen, apply shaders…

## Theory

### Algorithm to recognize hands

From what we have said earlier about the different features offered by the Kinect we have decided to only use the Depth data as it was reliable under any conditions (poorly light room, bright room, user sat down…).

Our system recognizes the hands of the user and uses them to control cursors. But having only the Depth data means we would have to execute an algorithm to recognize the hands and isolate them from the rest of the body.

Since the user needs to move his hands in order to control the cursors means that the closest parts of the body to the Kinect are the hands. They would typically be in front of the user as he is using our system. Applying this logic to the Depth data from the Kinect we would filter the pixels based on their distance from the sensor of the Kinect, the closest pixel from the sensor being the tip of the hand.

Once the pixels are filtered we would then group these pixels by blobs because the closest blobs are more likely to be the hands of the user.

When all the blobs are done the system would track only the 4 biggest blobs on the screen and label them as potential hands candidates. We have decided to track the 4 biggest blobs because usually the Kinect detects both hands and the head. The last blob is added for safety because sometimes there can be some noise from the Kinect’s sensor.

### CPU vs. GPU

The objective of our system is to run in Real Time. So during the Analysis phase of the project we have concluded some experiments on how to improve the speed processing of our system. One way to improve it was to execute the image processing directly into the graphic card instead of the CPU.

Pixels

Time in sec

Used range

Image 8 Speed processing on the GPU and the CPU

As you can see on this comparison image between the GPU and the CPU there is a huge gap between the speed processing done on the GPU and the CPU.

The CPU tends to be faster as the image gets smaller whereas the GPU is constant and always very fast.

## Technical

### Running in Real Time

As we said earlier, one of the objectives of this project is to make this system run in Real Time. The Kinect runs at 30 frames per seconds, this means that our system must approximately run everything under 33 milliseconds.

During the whole project we tried to optimize the algorithms implemented and we have found 2 algorithms that greatly improved the speed processing of the system:

* Processing done on the GPU

We have demonstrated earlier that the GPU is faster than the CPU when it comes to image processing. So we tried to use the graphical card when possible, only the small tasks are done on the CPU.

* 3D scene rendering

The main window of our system represents a 3D scene. At first when we implemented this scene we did everything on the CPU and saw that the whole scene was quite slow. But then when we tried to render the 3D scene and compose it on the GPU everything was smooth.

From theses algorithms we have determined that it is very important to fully utilize the capacity of our computers. Dividing the tasks into the CPU and the GPU at the same time can greatly improve the speed processing.

### Implemented algorithm for the hand recognition

The hand recognition algorithm is composed of 7 steps:

* Kinect depth acquisition (CPU – driver)

The depth image is provided by the Kinect as a 320x240 pixels image. Each pixel is 32bits wide and contains the depth as a 24bits integer and the player index on the last 8 bits. Only the depth is used, ranging from 800mm to 4000mm. This depth image is available at an average of 30Hz.

* Depth filtering and scaling (GPU)

The depth information is processed in order to remove the noise from the background.

* Region extracting (GPU)

Starting from the depth of the closest pixel, the space is divided in order to keep only the closest pixels on the zone viewed by the camera. In the defined usage scenario, these blobs will match the hands of the user.

* Borders gradient (GPU)

The gradients at the borders of the region extracted are computed to get a discrete direction along the 8 principal axes using a Sobel operator.

* Blobs counting and directions approximation (CPU)

Each separate region is counted as a blob having several parameters such as its center, its number of pixels, and its orientation (using the discrete direction at the borders).

* Tracking of the 4 main blobs (CPU)

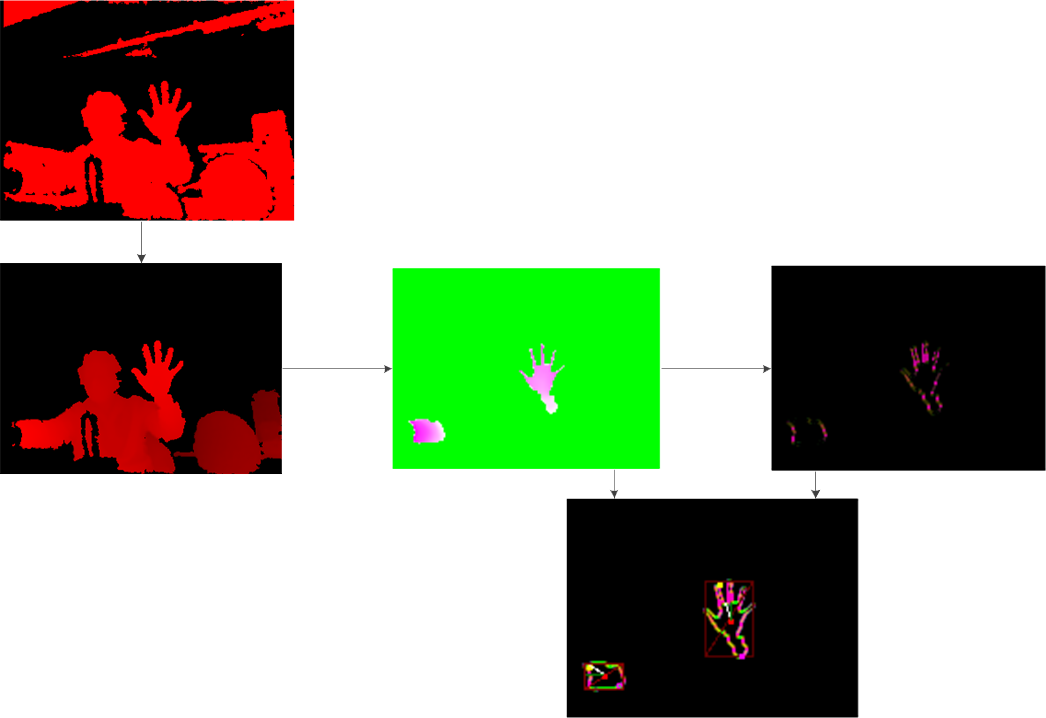
For software then tracks the 4 biggest blobs based on their attributes to propagate the information for the next frame. For each blob an average cursor position is computed.

* Gesture recognition and web pages rendering (CPU)

Each average cursor position is processed through the gesture recognition software, and the Web Page is rendered through a custom port of Chromium into a texture.

* Final composition for the UI (GPU)

The Web Browser Texture is used in a 3D scene and composed back into the WPF User Interface.



Final data : blobs + borders + orientation

Borders gradient

Depth based regions extraction

Filtered depth

Original Kinect Depth Image

Image 9 Hand recognition

## Evaluations of the system

Now that our system is complete we need to evaluate our work done.

### Definition

* Hand recognition system

This is our project which is based on blob tracking from the Kinect depth data.

* Kinect Skeleton

This is the native skeleton data available when the user is recognized.

### Protocol

For this we have decided to check certain criteria that are important to a NUI system:

* Accuracy

This evaluation is used to compare the accuracy between the 3 possible input devices in our system: mouse, Kinect Skeleton and finally our Kinect hand recognition.

For this test we have decided to try out the Kinect Skeleton recognition and implement it in our project. We wanted to see how our hand recognition fares against the “ready-to-use” skeleton recognition and the mouse.

* Processing speed

The speed processing evaluation is used to check how fast our system works on different computers. As we said earlier we want our system to work in Real Time so we want to know the minimal PC requirements for our system to work fully.

* Word input speed

For this project we have implemented 2 types of virtual keyboard: a “circular keyboard” where the letters are displayed around a textbox and a “T9 keyboard” which represents the same type of keyboards that we find on phones.

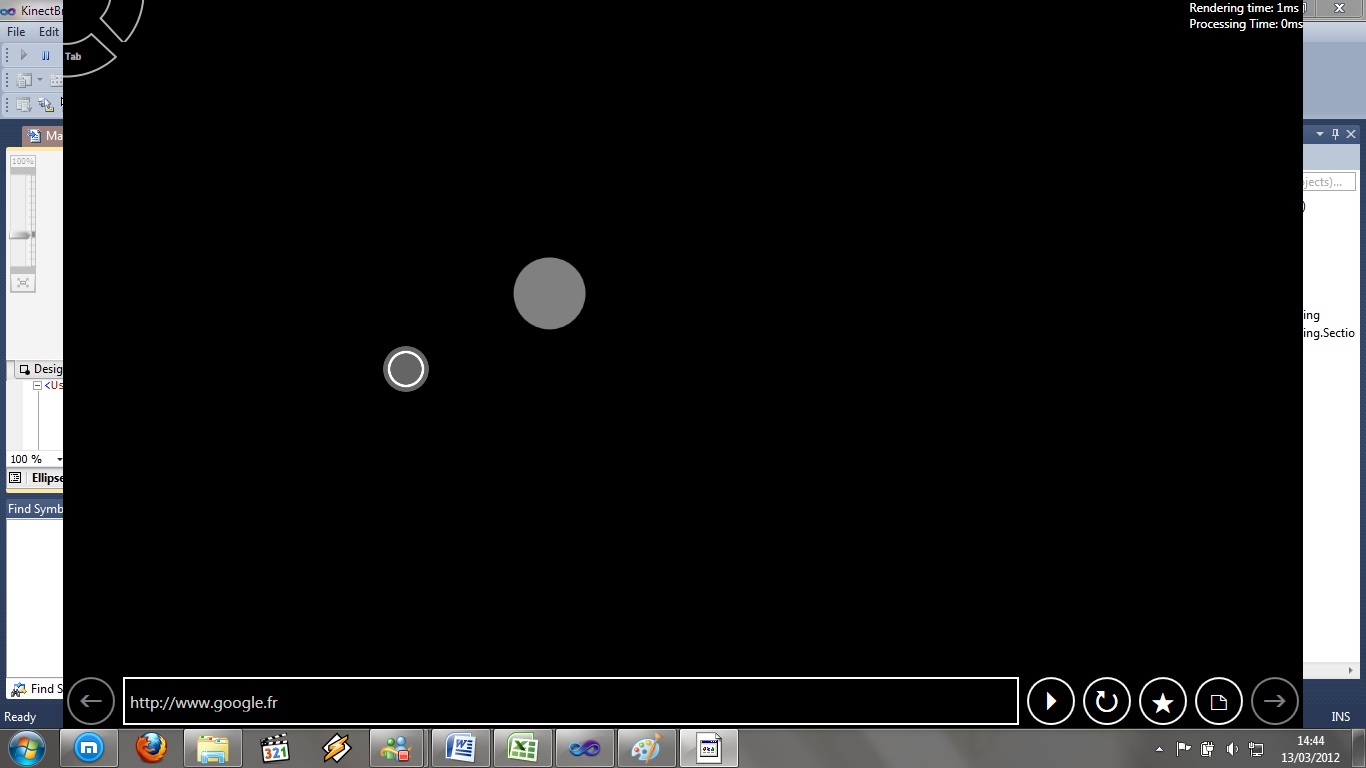
The goal of this test is to see how fast a user can write a word using these keyboards.

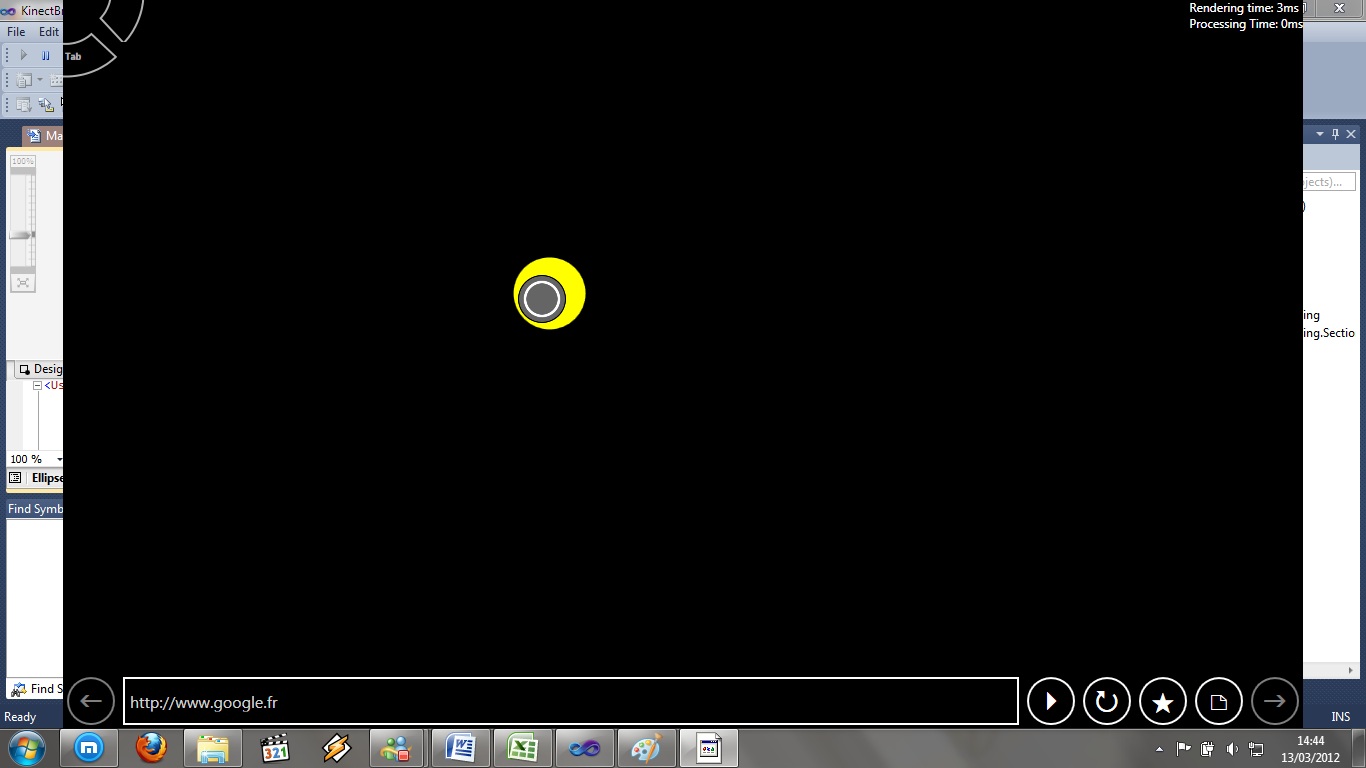
### Results

### Accuracy evaluation

The goal of this test is for the user to point at and to stay in different targets on the screen.

Once the user validates a target he has to wait 2 seconds inside the target and then a new one appears on the screen. We have tested different size targets to see how precise our system works on small or large targets.





For each of the input devices we have 4 criteria:

* Arrival duration: how fast we can point at the target
* Average error: how far we are from the center of the target when it is validated
* Percentage time in the target zone: the % time we stay in the target zone
* Distance moved when the target is validated: how stable the cursor is when the target is validated

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Mouse |  |  |  |  |  | |
| ZoneSize (px) | Arrival Duration (s) | Avg Error (px) | % time in zone | Distance in rest | Distance (processed) | |
| 24 | 0,765 | 4,62 | 99,88 | 73,9 | 7,39 | |
| 32 | 0,818 | 4,71 | 100 | 43,1 | 4,31 | |
| 64 | 0,728 | 6,84 | 100 | 28,31 | 2,831 | |
| 72 | 0,611 | 11,97 | 100 | 54,54 | 5,454 | |
| 92 | 0,571 | 26,04 | 100 | 67,32 | 6,732 | |
| 128 | 0,464 | 50,11 | 100 | 39,63 | 3,963 | |
|  |  |  |  |  |  | |
| Kinect Hands |  |  |  |  |  | |
| ZoneSize | Arrival Duration (s) | Avg Error (px) | % time in zone | Distance in rest | | Distance (processed) |
| 24 | 3,08 | 15,07 | 74,86 | 488,39 | 48,839 | |
| 32 | 2,92 | 21,16 | 94,91 | 492,85 | 49,285 | |
| 64 | 2,45 | 16,94 | 95,34 | 375,31 | 37,531 | |
| 72 | 3,03 | 24,99 | 94,5 | 495,51 | 49,551 | |
| 92 | 2,23 | 31,99 | 100 | 443,38 | 44,338 | |
| 128 | 2,14 | 57,39 | 100 | 574,17 | 57,417 | |
|  |  |  |  |  |  | |
| Kinect Skeleton | |  |  |  |  | |
| ZoneSize | Arrival Duration (s) | Avg Error (px) | % time in zone | Distance | Distance (processed) | |
| 24 | 3,226 | 20,75 | 64,09 | 321,28 | 32,128 | |
| 32 | 3,223 | 17,27 | 72,1 | 334,88 | 33,488 | |
| 64 | 2,289 | 19,26 | 97,26 | 318,44 | 31,844 | |
| 72 | 2,351 | 26,03 | 99,49 | 340,85 | 34,085 | |
| 92 | 2,084 | 31,81 | 100 | 315,1 | 31,51 | |
| 128 | 1,775 | 57,44 | 100 | 332,2 | 33,22 | |

Graph Arrival duration

Diameter of the target

Time in sec

Diameter of the target

Accumulated distance from center of the target in pixel

Graph Average error

Percentage

Diameter of the target

Graph 3 Percentage time in zone

Accumulated perturbations from the desired position in pixels

Diameter of the target

Graph 4 Stability

From these tests we can see that our hand recognition is equivalent to the Kinect Skeleton. Even though the Kinect Skeleton is a little more stable than our hand recognition system, the main difference is that our hand recognition works when the user is it at least 1 meter away from the Kinect and sat down which is not possible with the Kinect Skeleton.

We can also see that starting from targets of 64 or 72 pixels in size it is easier for the user to point at targets. The bigger the target is the faster the user can point at it.

### Processing speed

The goal of this evaluation is to see the minimal PC requirements needed for our system to work at 30 frames per second.

The Kinect is used in the following configuration:

* Kinect Depth Camera : 320x240 px
* Image Processing : 320x240 px
* 3D Rendering : 1024x768 px

For this evaluation we have tested how our system works on 2 different computers:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| * PC1 (Karim) | Core i7 2 Ghz | Radeon HD 4650 |  |  |
| * PC 2 (Kevin) | Core 2 Duo 2,13 Ghz | Radeon HD 4530 |  |  |
|  | Proc Time (ms) | Render Time (ms) | Total | Max FPS |
| PC1 Normal | 18 | 3 | 21 | 47,6190476 |
| PC1 Eco | 30 | 52 | 82 | 12,195122 |
| PC2 Normal | 30 | 20 | 50 | 20 |
| PC2 Eco | 50 | 40 | 90 | 11,1111111 |

Time in ms

Kinect 30 FPS

Graph 5 Processing and Rendering time

From this evaluation we can see that we need at least a PC1 configuration to make our system fully functional. For the PC2 Normal configuration, the system is still usable but it is not a smooth as the PC1 normal configuration.

### Word input speed

For this project we have decided to implement 2 types of keyboard:



Image Circular Keyboard



Image 11 T9 Keyboard

For each of these keyboards he have tested how long it takes for the user to type a word and in how many clicks he can do it.

|  |  |  |  |
| --- | --- | --- | --- |
|  | words | time (s) | nb clicks |
| Circular Keyboard | yes | 17,26 | 4 |
|  | zero | 20,2 | 5 |
|  | apple | 25,88 | 5 |
|  | morning | 22,11 | 6 |
|  | establishment | 53,64 | 11 |
|  |  |  |  |
| T9 Keyboard | yes | 14,54 | 4 |
|  | zero | 15,42 | 5 |
|  | apple | 17,86 | 6 |
|  | morning | 16,56 | 5 |
|  | establishment | 17,48 | 7 |
|  |  |  |  |

Time in seconds

Graph Word input time

Number of clicks

Graph 7 Number of clicks to input word

From these tests we can see that the T9 keyboard seems more stable as it usually takes around 17 seconds to input words and in around 5 to 6 clicks.

# Results

The system is fully functional and respects the objectives fixed at the beginning of the project. A web browser implemented inside a 3D scene was developed as well as all the interactions (click, zoom, scroll, new tab…).

The total execution time is from 10 to 18 ms which is good compared to the 33 ms limit.

The minimal functional distance is 1 meter. This is a limitation of the Kinect: right now it can only offer data when the user is at least 1 meter away from the sensors.

The only problem right now is that this system can only track a single user. It can only detect 2 hands on the screen so if somebody else passes at the back of the user or in front the system might lose the tracking of the user’s hands.

From the evaluations we have done earlier our hand recognition system works very well and is on par with the Kinect Skeleton. The main difference with the Kinect Skeleton is that our hand recognition system can be used even when the user is sat down. The cursor is stable enough to let the user points at either small or large targets even though it is easier for the user to have large icons.

Although the mouse is still more precise and faster, our hand recognition system is fully usable when it is used with a web browser.

# Conclusion

From this project we have learned a lot about how the Kinect works:

* 4 types of features offered
  + Depth data
  + Color data
  + Skeleton recognition
  + Voice
* Limitations:
  + Minimal distance of 1 meter
  + Accuracy lost when the user is not standing up

After developing this system we can say that the Kinect is an “easy-to-use” camera as it offers all the data necessary for the developer as well as a SDK.

We have also learned from this project the importance of the optimization of algorithms and how to divide the tasks on the CPU/GPU. As the speed processing was one of the key points of this project we had to determine the best ways to implement the algorithms.

A Kinect 2 was developed by Microsoft and released by Microsoft at the end of our project. We didn’t have time to test it but from what we have read it improved the minimal distance to 50 cm and also generally improved the accuracy of the sensors.

The new one has a better resolution (640x480 px) whereas the older Kinect was only at 320x240 px.

The Kinect 2 also adds a new “Near mode” which enables the Kinect to work from 50 to 300 cm, the far mode working from 80 to 400 cm.

# Glossary

***NUI***: “Natural User Interface” is a user interface that is effectively invisible, or becomes invisible with successive learned interactions, to its users, and is based on natural interactions.

***SDK***: “Software Development Kit” set of programs and libraries to help develop applications.

***Kinect***: Microsoft camera used to capture gestures done by the user.

***User interface***: Visible part of the program for the user where he can interact with the application.

***Metro design***: Design for user interface which is appropriate for touch devices (large buttons, wide text…).

***CPU***: “Central Processing Unit” is the portion of a computer system that carries out the instructions of a computer program, to perform the basic arithmetical, logical, and input/output operations of the system.

***GPU***: “Graphical Processing Unit” is designed to rapidly manipulate and alter memory in such a way so as to accelerate the building of images in a frame buffer intended for output to a display.

# References

Kinect: <http://en.wikipedia.org/wiki/Kinect>

WPF: <http://www.kirupa.com/blend_wpf/index.htm>

Expression Blend: <http://codeflow49.blogspot.com/search/label/Expression%20Blend>

Metro design: <http://www.jeff.wilcox.name/2011/03/metro-design-guide-v1/>

MSDN: <http://msdn.microsoft.com/en-us/ms348103>

XNA: <http://msdn.microsoft.com/en-us/aa937791>

NVidia: <http://developer.download.nvidia.com/shaderlibrary/webpages/shader_library.html>

# Appendix

## Screenshots of the system

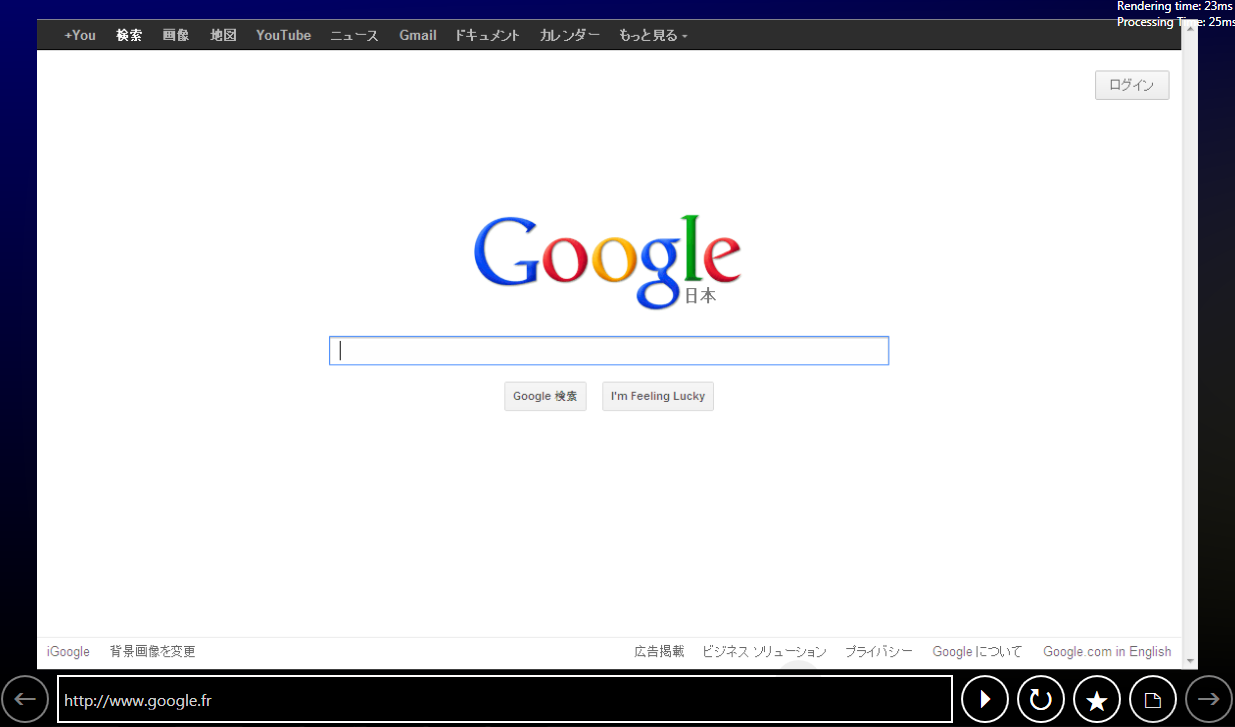


Image 12 Main window



Image Circular menu



Image Circular Keyboard



Image T9 Keyboard

## General Specifications Document

Summary

[I. Introduction 42](#_Toc306269327)

[1. Purpose of the document 42](#_Toc306269328)

[2. Glossary 42](#_Toc306269329)

[II. Natural User Interface 43](#_Toc306269330)

[1. Purpose of the project 43](#_Toc306269331)

[2. Global description of the system 43](#_Toc306269332)

[3. Composition of the system 43](#_Toc306269333)

[4. How the system will work 44](#_Toc306269334)

[a. User interface 44](#_Toc306269335)

[b. Movement recognition 44](#_Toc306269336)

[III. Planning 45](#_Toc306269337)

[1. Calendar 45](#_Toc306269338)

[2. Gantt chart 47](#_Toc306269339)

[IV. Requirements 48](#_Toc306269340)

[1. Environment 48](#_Toc306269341)

[2. Demonstration 48](#_Toc306269342)

[3. Conception 48](#_Toc306269343)

[4. Implementation 48](#_Toc306269344)

[V. Quality assurance 49](#_Toc306269345)

[1. Development method 49](#_Toc306269346)

[a. Planning of the project](#_Toc306269347) 49

[b. Analysis 49](#_Toc306269348)

[c. Conception 49](#_Toc306269349)

[d. Implementation 50](#_Toc306269350)

[e. Tests 50](#_Toc306269351)

[f. Final check 50](#_Toc306269352)

[2. Control 51](#_Toc306269353)

[a. Meetings 51](#_Toc306269354)

[b. Bug tracking 51](#_Toc306269355)

[c. Project report 51](#_Toc306269356)

[3. Quality check 51](#_Toc306269357)

# Introduction

## Purpose of the document

The purpose of this document is to present our project with Kinect: Implementation of a Natural user interface. This document will present in details the environment and what we will do in the project.

## Glossary

*SDK*: “Software Development Kit” set of programs and libraries to help develop applications.

*Kinect*: Microsoft camera used to capture gestures done by the user.

*User interface*: Visible part of the program for the user where he can interact with the application.

*Metro design*: Design for user interface which is appropriate for touch devices (large buttons, wide text…).

# Natural User Interface

## Purpose of the project

The goal of this project is to develop a natural user interface with Kinect. This means that the user will be able to interact with the computer via hand/arm motions.

The Kinect program will be developed with the Kinect SDK that came out last April.

For this project we will also develop a web browser based on the design of Internet Explorer 10. This browser will be specifically adapted to Kinect using a Metro style. At the end the web browser will be able to understand the various input devices such as Kinect, the mouse and the keyboard.

## Global description of the system

The system will be composed of 3 parts:

* Data acquisition: the system will be able to acquire data from the Kinect, the mouse and the keyboard
* Movement recognition: the system will have to recognize and understand the various gestures done via the Kinect. The gestures recognized will be analyzed using a database built with learning algorithms.
* User interface: a metro style UI will be developed to let the user interact with the Kinect.

## Composition of the system

User interface :

Metro style

Input devices :

Kinect, mouse, Keyboard

## How the system will work

### User interface

The user will have to first initialize the system. When it is done he will arrive in the front page of the program where he will choose to launch: a web browser, his mail account or the desktop.

The user will choose the different options using the Kinect or the mouse. By doing gestures the system will recognize and interpret them. From this application the user could zoom in/out, click, and move the page up or down…

### Movement recognition

From the data acquisition done via the Kinect, our system will have to analyze and recognize the gestures. From a database containing different patterns, the system will have to match the gestures the user has done with the patterns.

Each gesture will be composed of successive simple movements. If a complex gesture is done by the user, for example moving his hand right and up then the system will recognize first the movement “going right” and then the movement “going up”. The purpose of this solution is to simplify every gesture done by the user: these movements are independent from the time and the distance. So it will not matter if the user has done a very slow gesture or a very small one.

# Planning

## Calendar

Planning of the project: 2 weeks

* Define the project
  + Purpose of the project
  + Use cases
* Gantt chart
* Launch

Analysis: 5 weeks

* Data gathering of the various tools (camera/Kinect)
* Learning programming languages (Xaml, C#)
* Do prototypes to see feasibility and to see how it works
* Define functional and technical specifications
* Put priority on specifications

Conception: 3 weeks

* Conceive architecture of the system
* Conceive and integrate the database
* Conceive prototypes of the user interface
* Conceive prototypes of the system interface

Implementation: 13 weeks

* Data acquisition
  + Data gathering from Kinect
  + Data gathering from mouse/Keyboard
  + Data gathering from vocal
* Movement recognition + database management
  + Algorithm for the movement recognition + implementation
* User interface
  + Web browser
  + Mail
  + Photo/video
  + Vocal data management
* Integration

Tests: 2 weeks

* Functional tests

Final check: 1 week

* Check documents
* Clean code

## C:\Users\Kevin\Documents\Projet Japon\Gantt.pngGantt chart

# Requirements

## Environment

The Kinect must be connected to the computer. The user must be at least 2 meters away from the Kinect.

## Demonstration

For the presentation of our system, the Kinect must be connected to the computer. Then the user will execute the application. From this point on he will be using the program by doing gestures, using the mouse/keyboard, or using vocal commands.

## Conception

The application done for this system will be object oriented.

We will use UML language to draw the different diagrams as it is well known by most developers.

## Implementation

The core system will be developed in C# on Windows 7 since the Kinect makes use of the Windows environment. We will also be using the Kinect SDK released earlier as it offers many libraries to work with the Kinect. For the user interface it will be developed in Xaml and C#.

The database used for this project will be MySQL.

We will use Git for our versioning as it will improve our organization. Moreover this versioning system is not centralized: each user keeps in locale the modifications they have done on the project; so each user can work on different branch of the project.

# Quality assurance

The system will must be of quality to satisfy the user. We will especially focus on the aesthetic of the user interface. It has to be both easy to use and visually attractive.

## Development method

The method used for this project is Agile. The main advantage of this method is to work by iterations. We first begin by working on the core of the system which has the least possibility of being modified through the project. At the end of iteration a meeting will be done to validate the solutions found.

The project is divided into 7 parts:

### Planning of the project

During this phase we first think of the functionalities offered by our system. By doing so we can then establish a Gantt chart and estimate the time needed for the development.

### Analysis

As it is a new technology to us we will have to study how the Kinect works and learn some new programming languages. During this phase we will also define the general and technical specifications that we will develop later.

The purpose of this phase is also to see the feasibility of the project by developing some prototypes or proof of concept.

### Conception

For this we will focus on the architecture of the system. From the prototypes done earlier we can see the different modules composing our system. We will use UML to draw the various diagrams such as class diagram, database diagram…

During this conception phase we will also sketch some rough screens of the user interface.

### Implementation

Now that we have decided the architecture of the system we will now implement the solution.

For the implementation phase it will be divided into 3 sprints:

* The first sprint will be on the data acquisition. By the end of this sprint we will have a functional module for gathering data from the input devices.
* The second sprint will focus on the algorithm of the movement recognition and some screens of the user interface. At the end of the sprint we will have some rough screens of the final user interface. By the end of this sprint each of us will work separately, one person will be in charge of the implementation of the movement recognition algorithm while the other will be working on the user interface.
* The last sprint will be the final version of our system. Everything should be implemented and functional at the end of the sprint.

### Tests

For this phase we will test every functions of our system. The purpose is to make sure that there is no bugs anymore and that the project is fully working.

### Final check

In this last phase we check the documents that we have written for the project, clean the code, check that the application is fully functional…

## Control

### Meetings

A weekly meeting with our teacher is done to see how the project is progressing. This meeting is also used to talk about the difficulties encountered in the project.

At the end of each phase/sprint, a little presentation is done to show what has been done so far.

### Bug tracking

Every bug encountered in this project will be written down as “not solved” with a small description of the bug and kept in an Excel file. If someone is working on a specific bug he has to leave his name next to the bug and change the bug’s status to “testing” so the team won’t try to solve the same bug. Once the person finds a solution he will have to write down the solution and finally update the status to “solved”.

### Project report

Each week a meeting is done to see how the project is progressing and what is left to be done for the next weeks. From these meetings, the project report is slowly done; the goal is to have most of the project report done by the end of the project.

## Quality check

|  |  |  |  |
| --- | --- | --- | --- |
| Iteration | Activity | Quality check | On… |
| Analysis | Analysis of the requirements | Document review | GRS |
| Sprint 1 | Whole iteration | Project review | Release Definition |
| Development | Code review | Code |
| Tests | Test review | Sprint Test List |
| Sprint 2 | Whole iteration | Project review | Release Definition |
| Development | Code review | Code |
| Tests | Test review | Sprint Test List |
| Sprint 3 | Functional tests | Test review | Project Report |
| Whole iteration | Project review | Project Report |
| System | Code review | Code |