UNIVERSITY OF MANITOBA

# Aircraft War with Kinect Sensor

Zhen ZHAO

Any questions and advices are welcome, email me today at zhen.now@gmail.com

Date: April 11, 2016

Electrical and Computer Engineering



# Outline

- **“Aircraft War” game**
- **Solution Overview**
- **Key Ideas & Techniques**
- **Source Code**
- **Run the demo**

# "Aircraft War" Game

- Show/hide Skeleton Window
- Virtual Switch
- Hero Controlling
  - Up, down
  - Left, right
  - Give up
  - Kill all (ultimate)
- Object Collision


From my YouTube Channel: *https://youtu.be/G7fK6GrSmB4*

# Solution

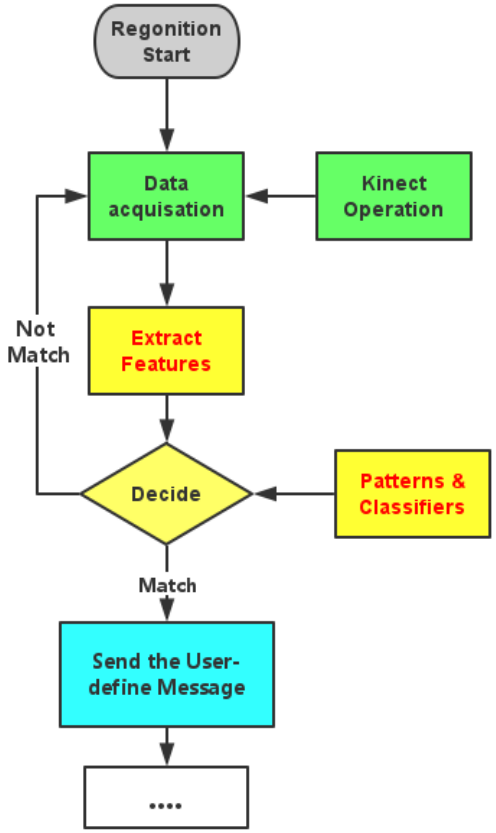
- **Platform:** Windows OS.
- **UI :** Win32 API
- **Gesture Recognition**
  - Static Movement:
    - Skeleton data from Kinect
    - Vector Angles in 3D space
  - Dynamic Movement:
    - multiple “static”
    - Timer

# Solution

- **Game Render:**
  - GDI double buffering for gaming view
  - OpenCV for Skeleton view
- **SDK/Libraries:**
  - Kinect for windows SDK v1.8
  - MCI libraries for background music
  - STL, Win API
- Programming Language: C++ (11)

# Solution

- Two major coding jobs
  - Game design
    - Game Object
    - UI render
    - Message Processing
  - Gesture recognition
    - Kinect-related design
    - Feature(Angle) design --- Classifiers
    - User-defined Message --- Actions




- Other jobs:
  - user score recording, play music, enemy generation

UNIVERSITY OF MANITOBA

# Key Ideas & Techniques

- Game Design
- Kinect Operations
- Gesture Recognition
- Motion Recognition
- Player Information

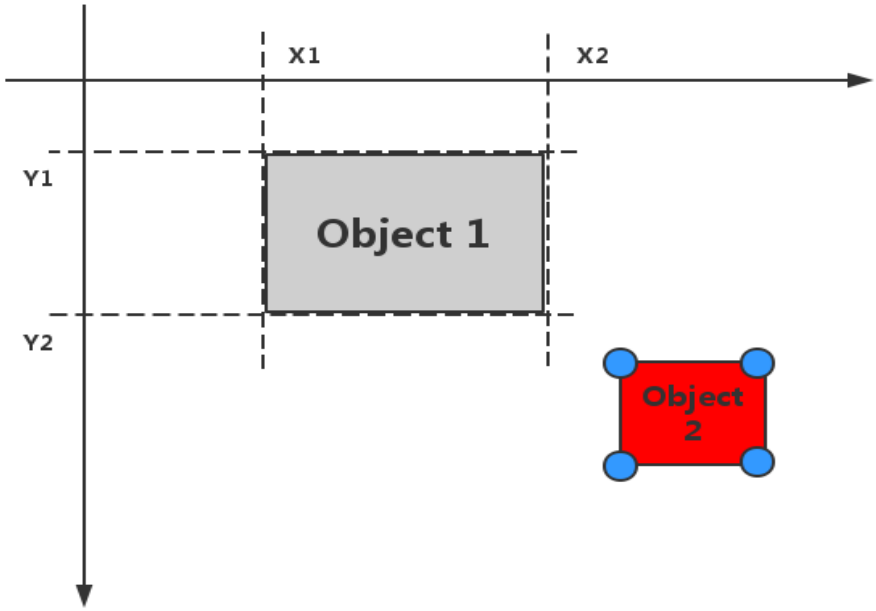
# Key Part 1: Game design

- Game State
  - GS_MENU, GS_PLAYING, GS_RESULT
- UI & UI Render
  - Create Window, Update Drawing
- Message Processing
  - Message Loop, Receiving and process the WM_KEYDOWN, WM_LEFTBUTTONXXX, WM_CLOSE.
  - Keyboard & mouse Events.

# Key Part 1: Game design

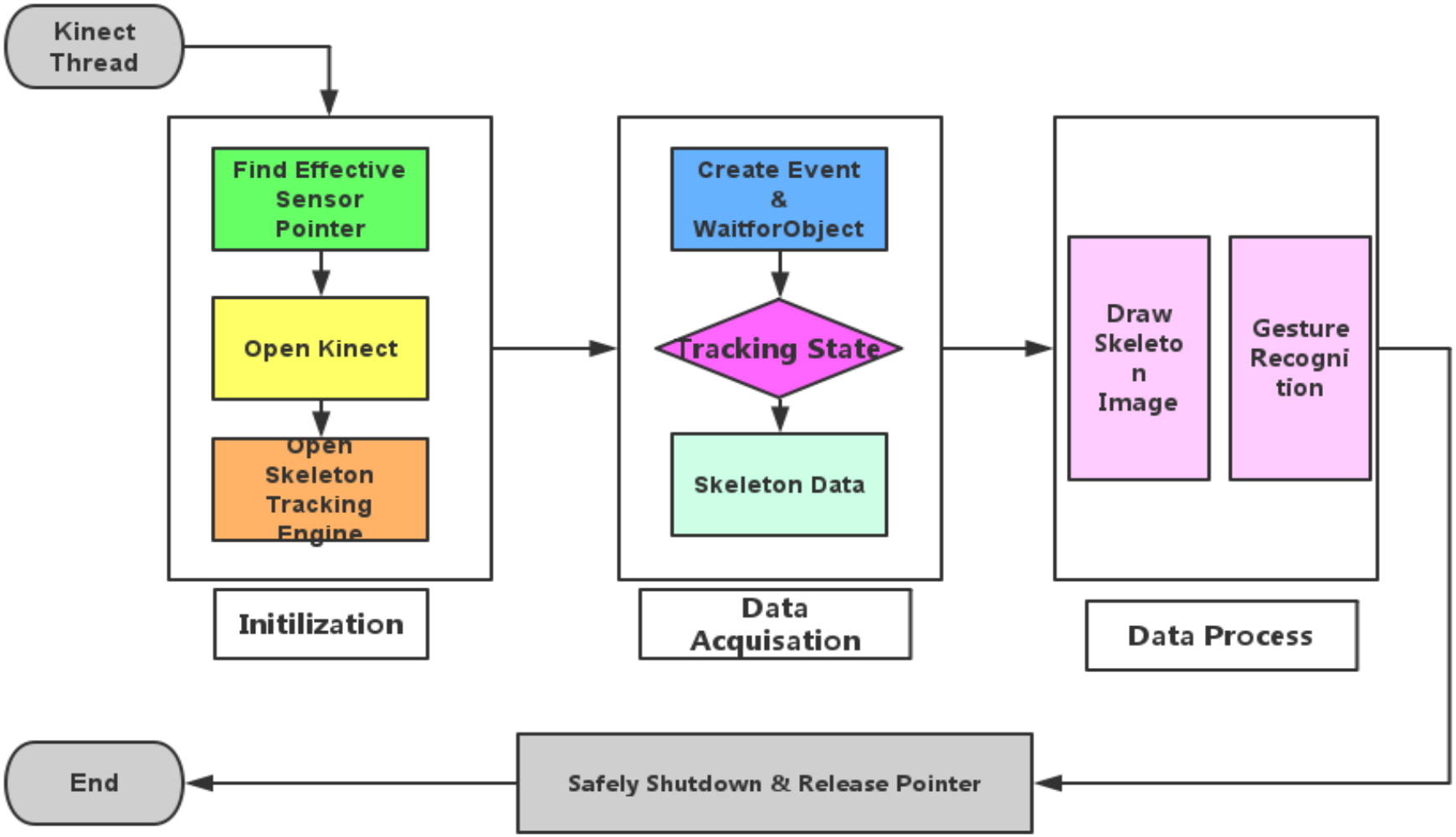
- Collision Detection

2-Dimension

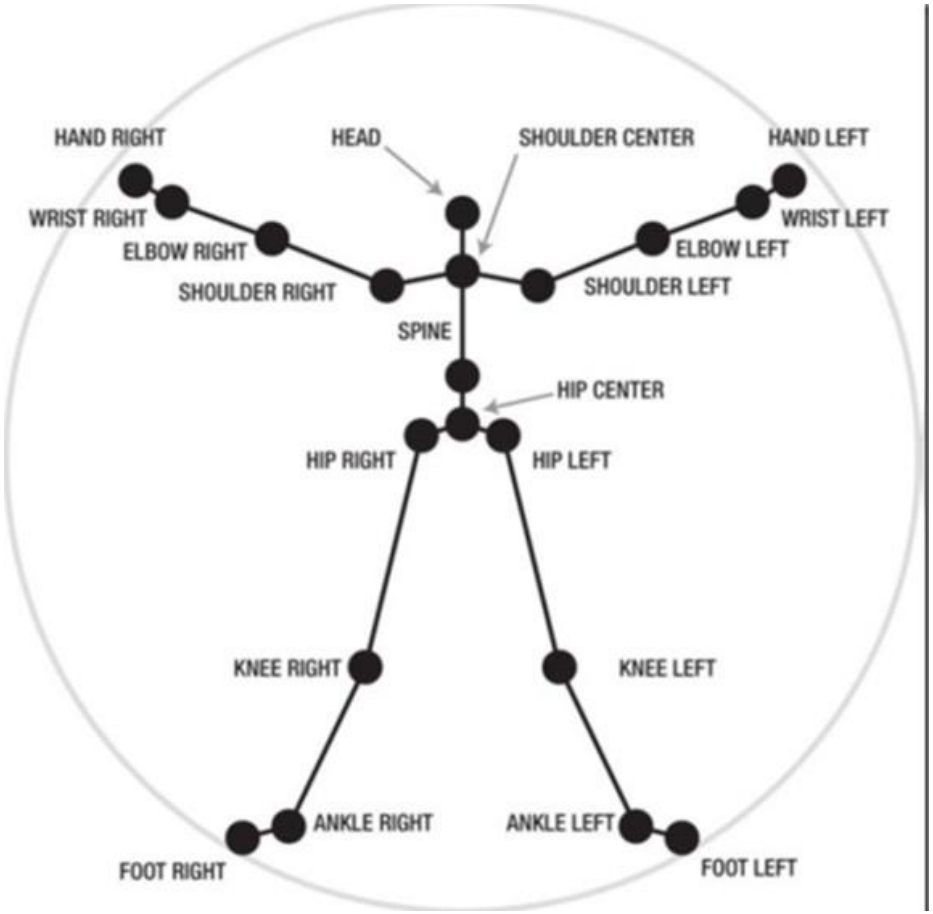


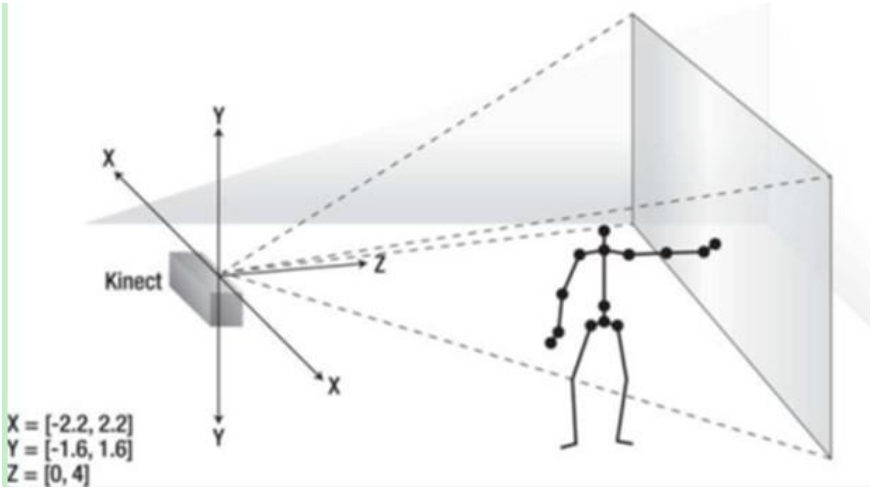

- User:
  - UE(user experience)
- Designer/Developer:
  - Art Design (video, image, sound)
  - Game Scripts (story)
  - Performance (memory, Multithread)
  - Algorithm (drawing, light, AI (path finding, Behavior Tree…))
  - Physical Property (Collision Detection, Gravity effect)

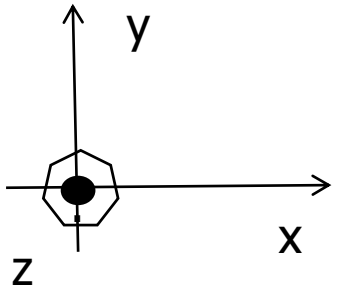
# Key Part 2: Kinect Operation



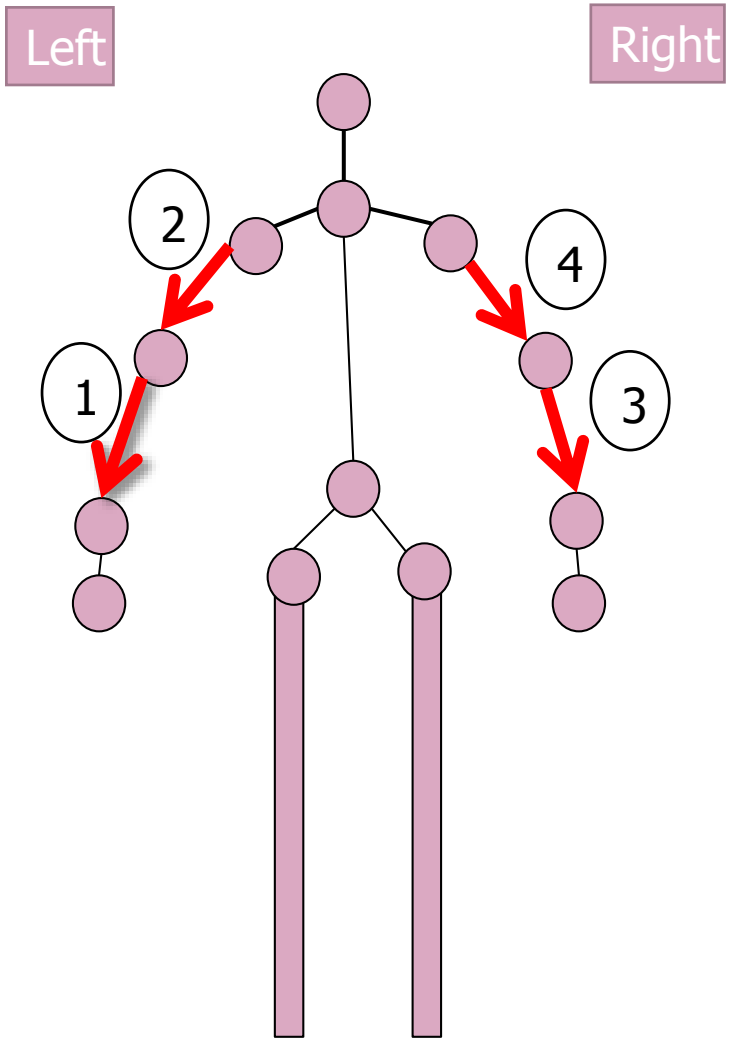
# Key Part 3: Gesture Recognition



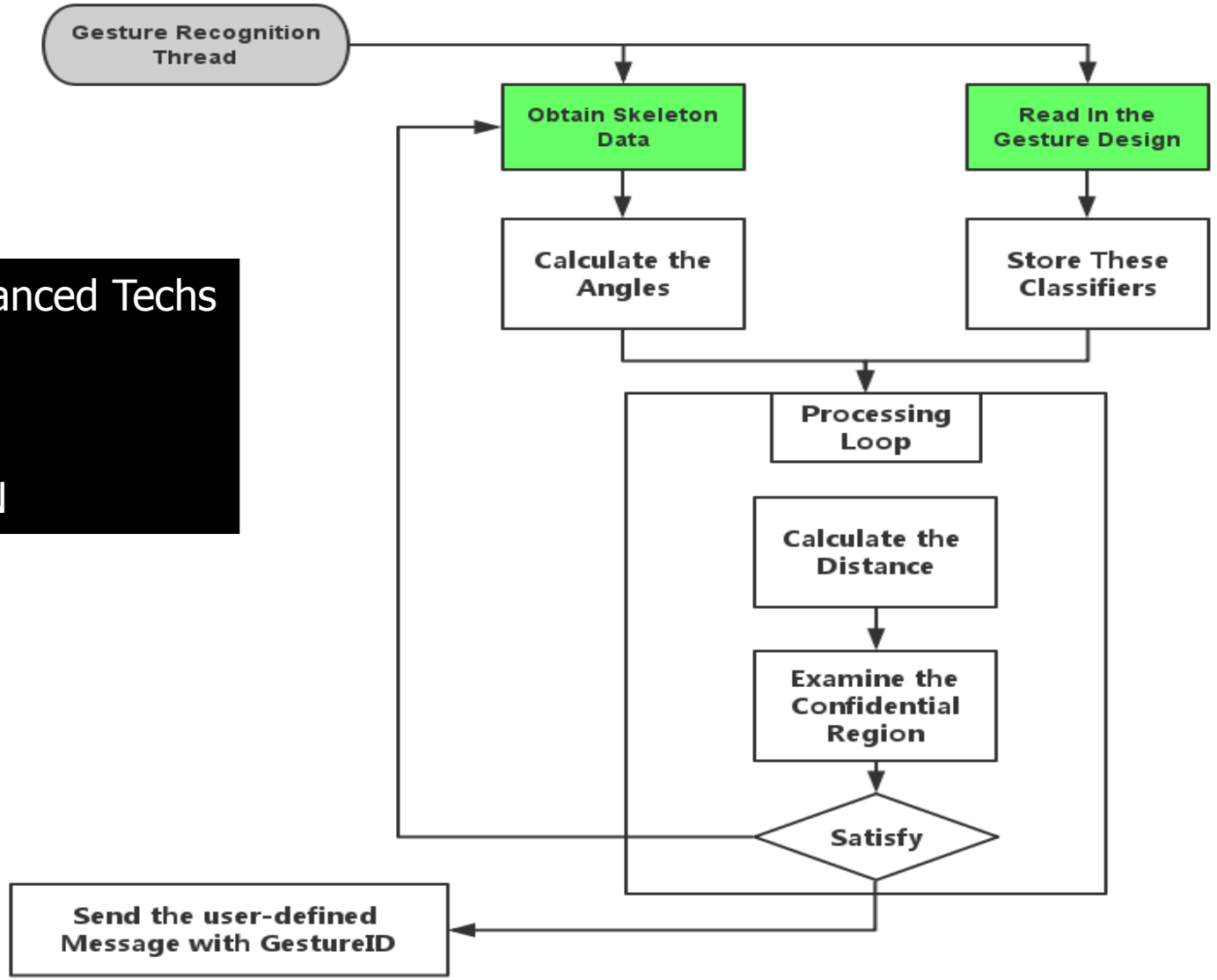
# Key Part 3:Gesture Recognition




| Features(Angles) | Tolerance Errors |
|---|---|
| Angle 1: <vec1, X> | (10,20) |
| Angle 2: <vec2, X> | (10,20) |
| Angle 3: <vec3, X> | (10,20) |
| Angle 4: <vec4, X> | (10,20) |
| Angle 5: <vec1, Z> | (10,20) |
| Angle 6: <vec2, Z> | (10,20) |
| Angle 7: <vec3, Z> | (10,20) |
| Angle 8: <vec4, Z> | (10,20) |

# Key Part 3:Gesture Recognition

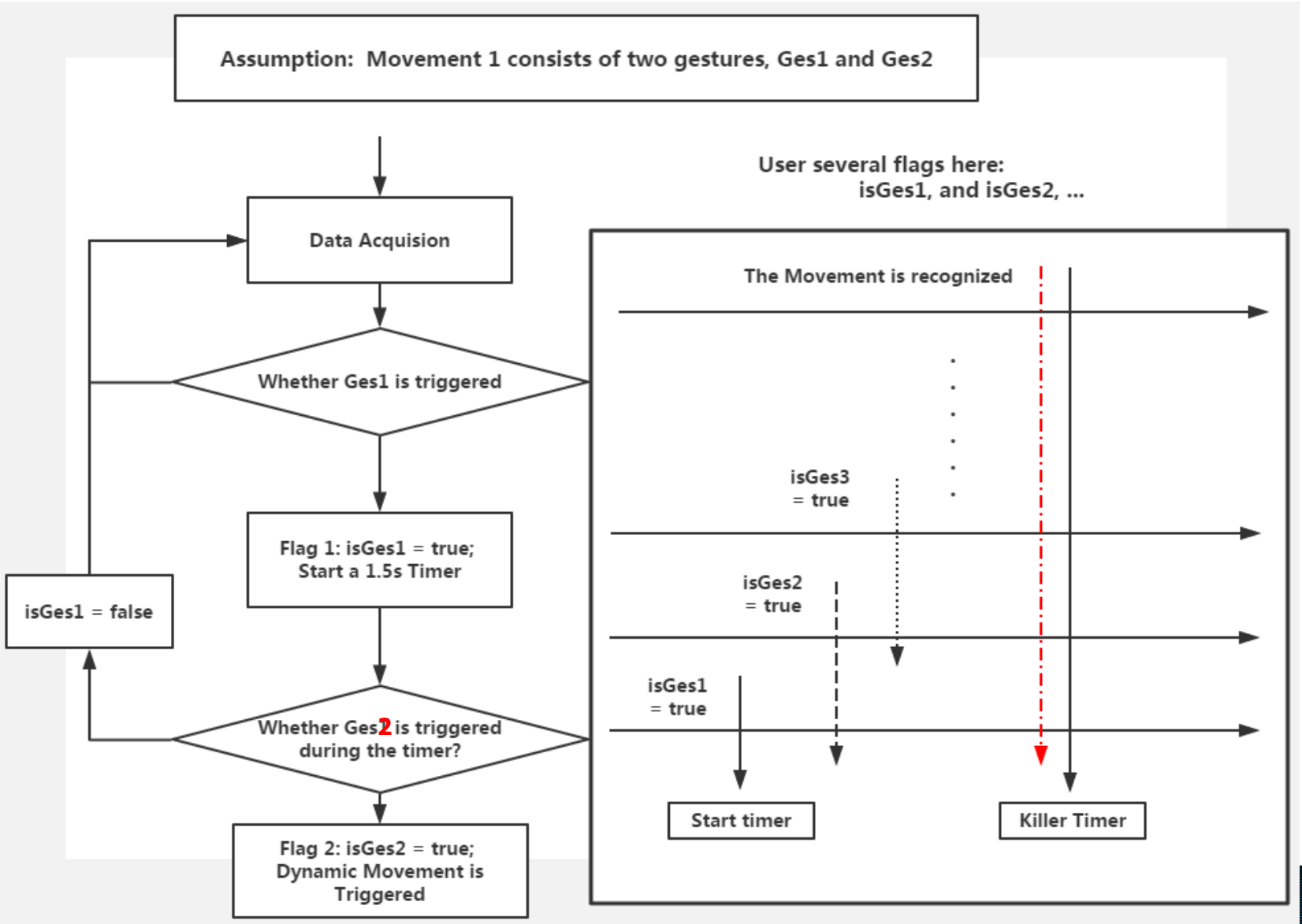
- More Advanced Techs
  - DTW
  - HMM
  - LPP
  - TDNN



# Key Part 4: Motion Recognition

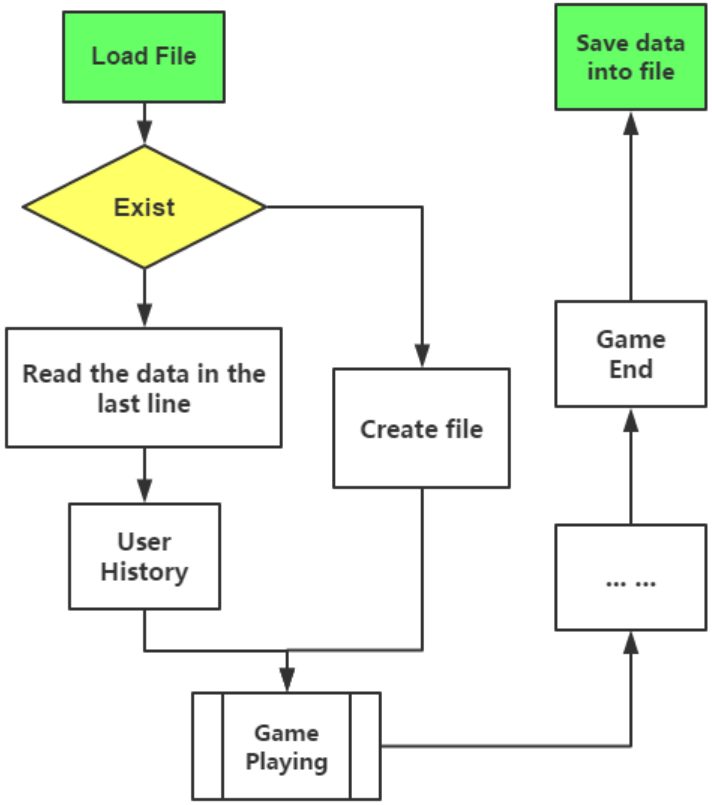
UNIVERSITY OF MANITOBA

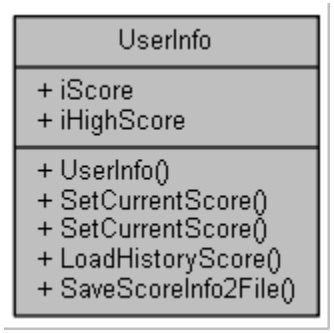
Video = multiple picture (changing fast)
Dynamic Movement = several static gestures during a short time



# Key Part 5: Player Information

- Server
- Local data storage
  - Structured way: lightweight Database (Sqlite), XML
  - Unstructured way: plain-text files



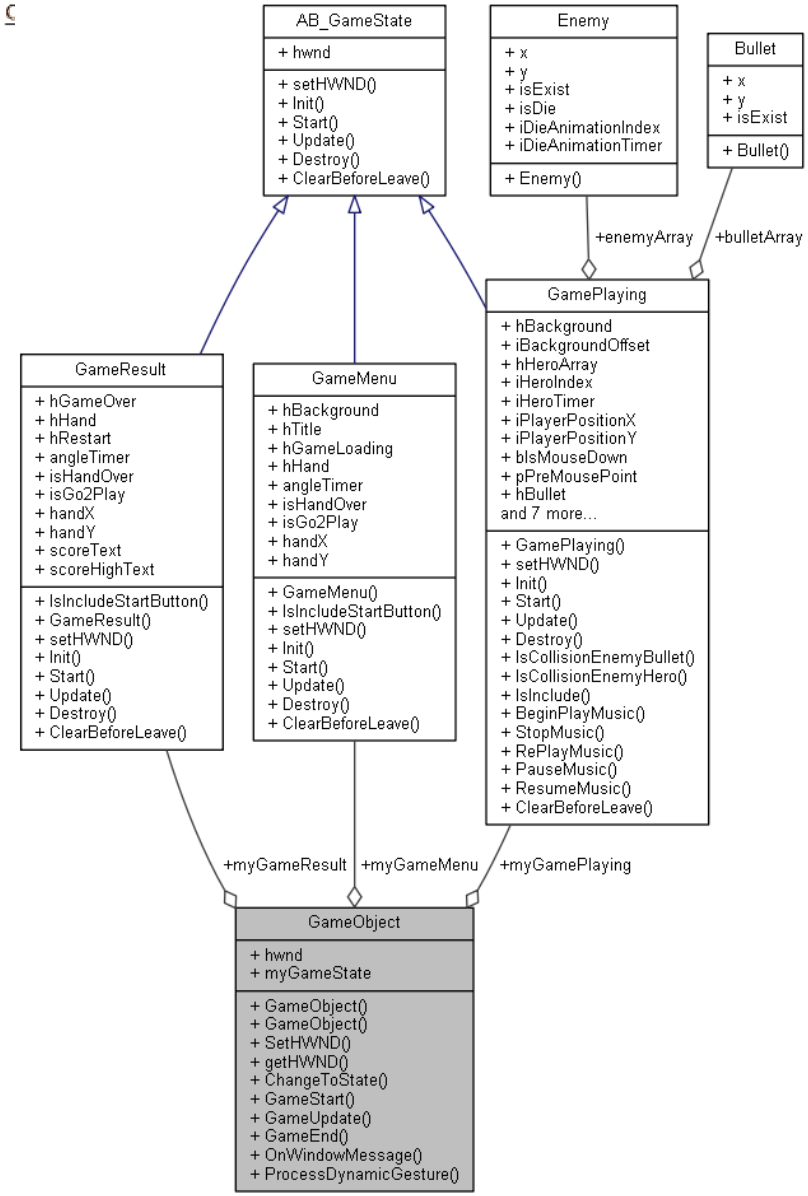
UNIVERSITY OF MANITOBA

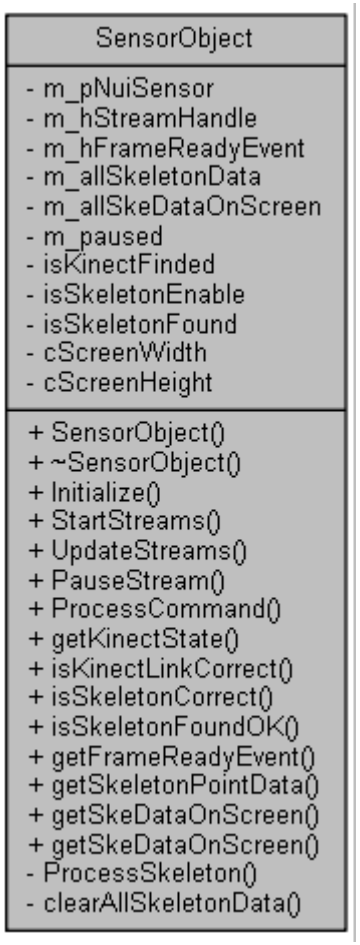
# Source Code

- **UML Diagram**
  - **(9 diagram: sequence, class, use case, activity … )**
  - **Class Diagram (Doxygen)**
- **Source Code in VS2010**

UNIVERSITY

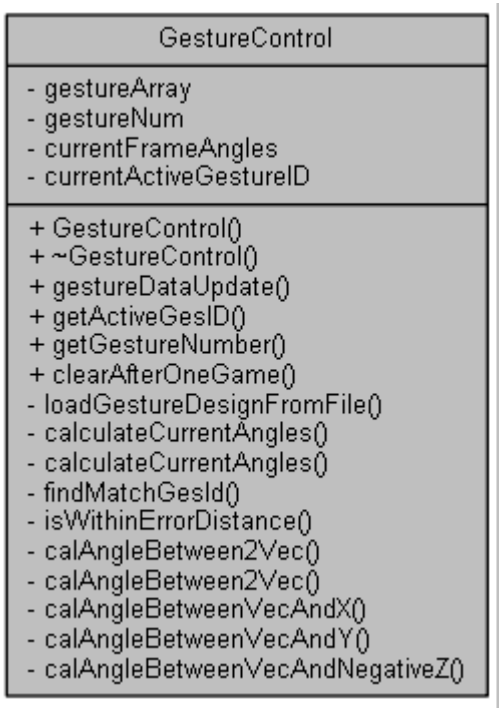
# Class Diagram




Link To Code Analysis

Link To Visual Studio



UNIVERSITY OF MANITOBA

# Run & Test the Demo

**Q&A**