



# Aircraft War with Kinect Sensor

Zhen ZHAO

Any questions and advices are welcome, email me today at [zhen.now@gmail.com](mailto:zhen.now@gmail.com)



# Outline

---

UNIVERSITY  
OF MANITOBA

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



# “Aircraft War” Game

UNIVERSITY  
OF MANITOBA

- 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

---

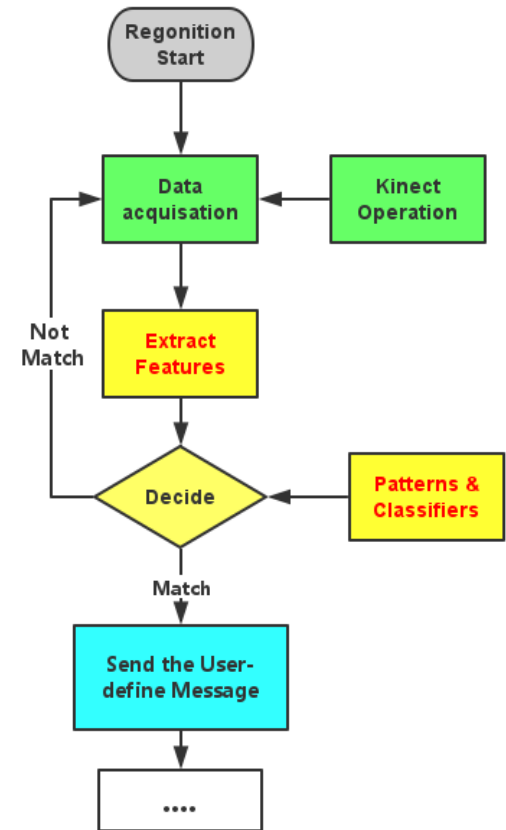
- **Game Render:**
  - GDI double buffering for gaming view
  - OpenCV for Skeleton view
- **SDK/Libraries:**
  - Kinect for windows SDK v1.8
  - MCI libraries for background music
  - STL, Win API
- Programming Language: C++ (11)



# Solution

UNIVERSITY  
OF MANITOBA

- 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





# Key Ideas & Techniques

---

UNIVERSITY  
OF MANITOBA

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



# Key Part 1: Game design

UNIVERSITY  
OF MANITOBA

- 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\_LEFTBUTTONDOWNXXX, WM\_CLOSE.
  - Keyboard & mouse Events.



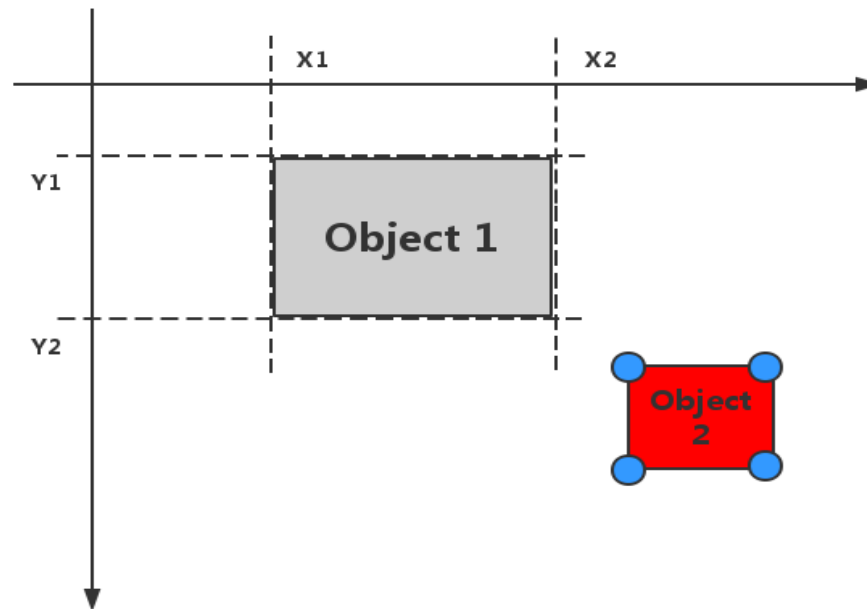


# Key Part 1: Game design

UNIVERSITY  
OF MANITOBA

## ■ Collision Detection

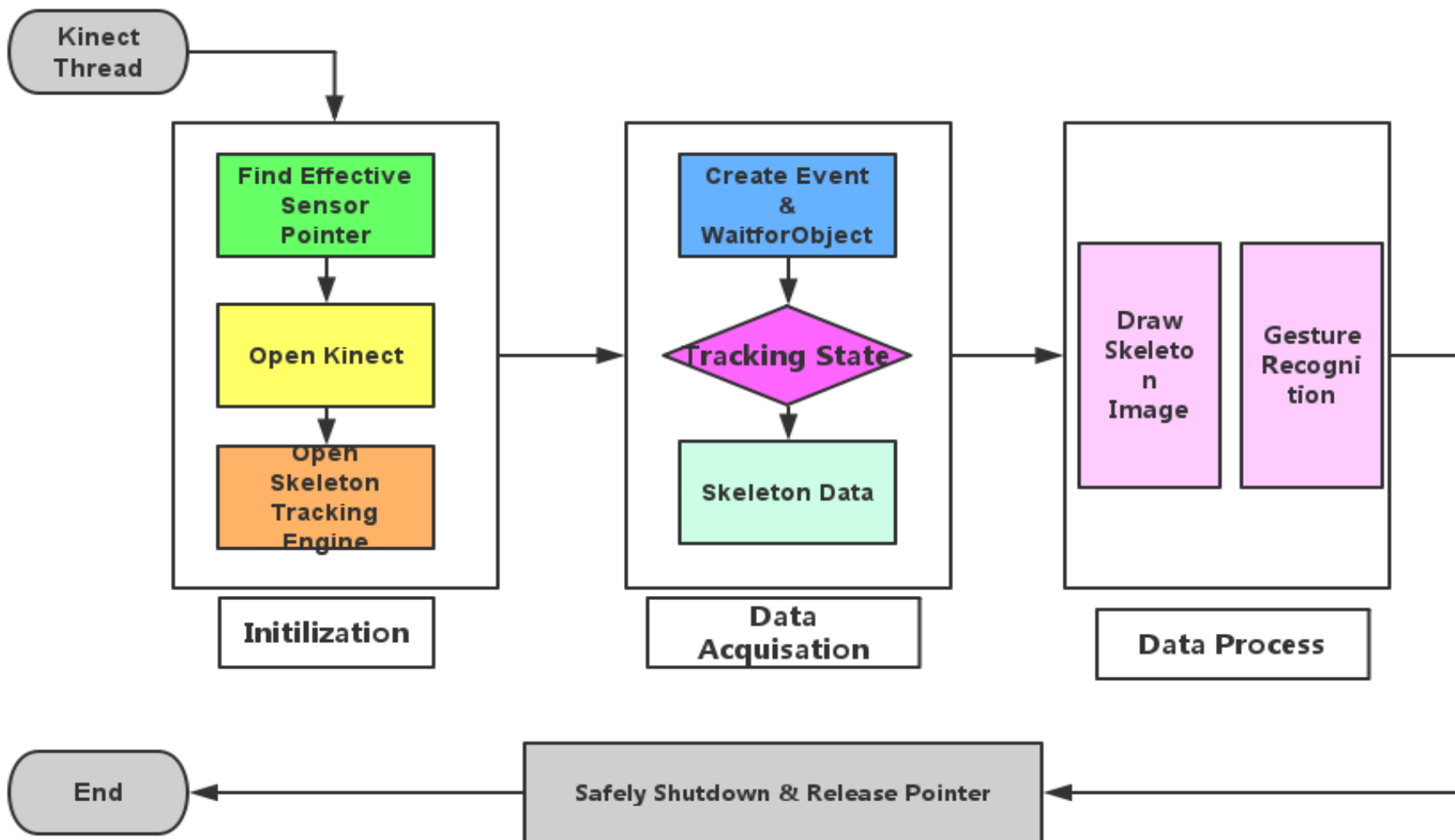
### 2-Dimension



- 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)



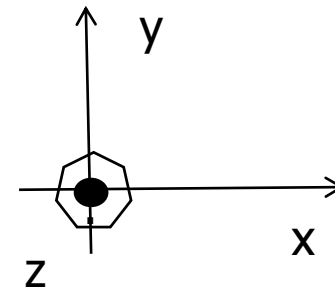
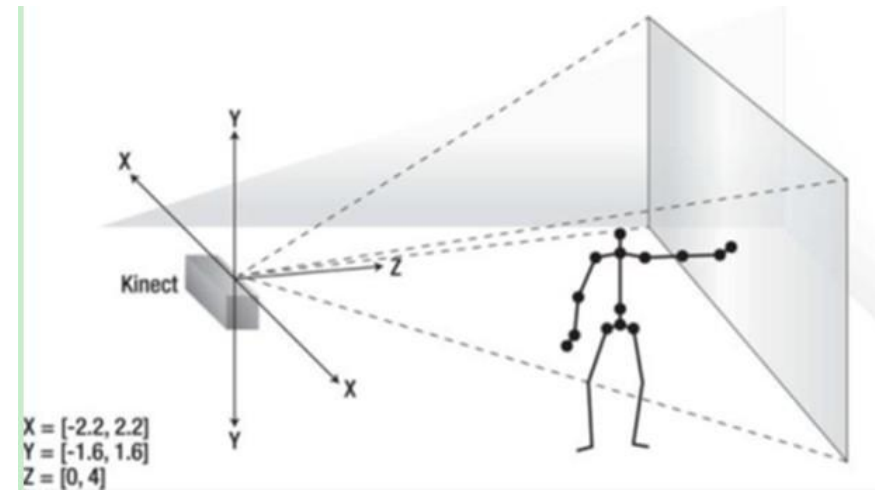
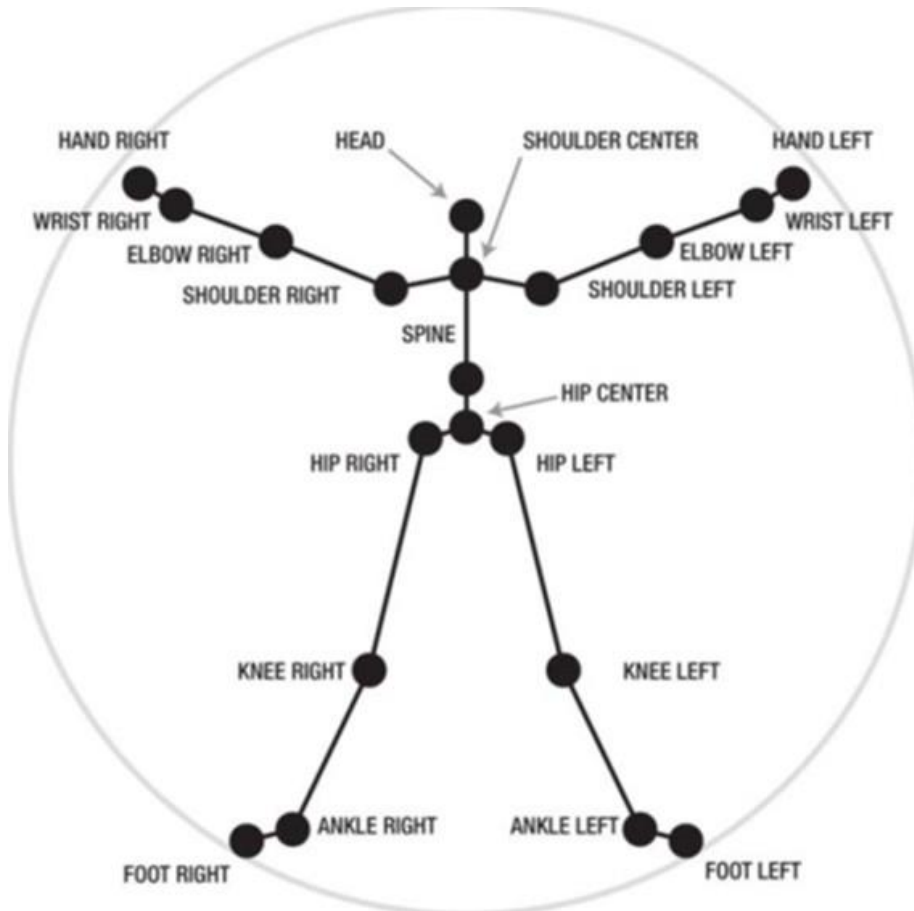
# Key Part 2: Kinect Operation





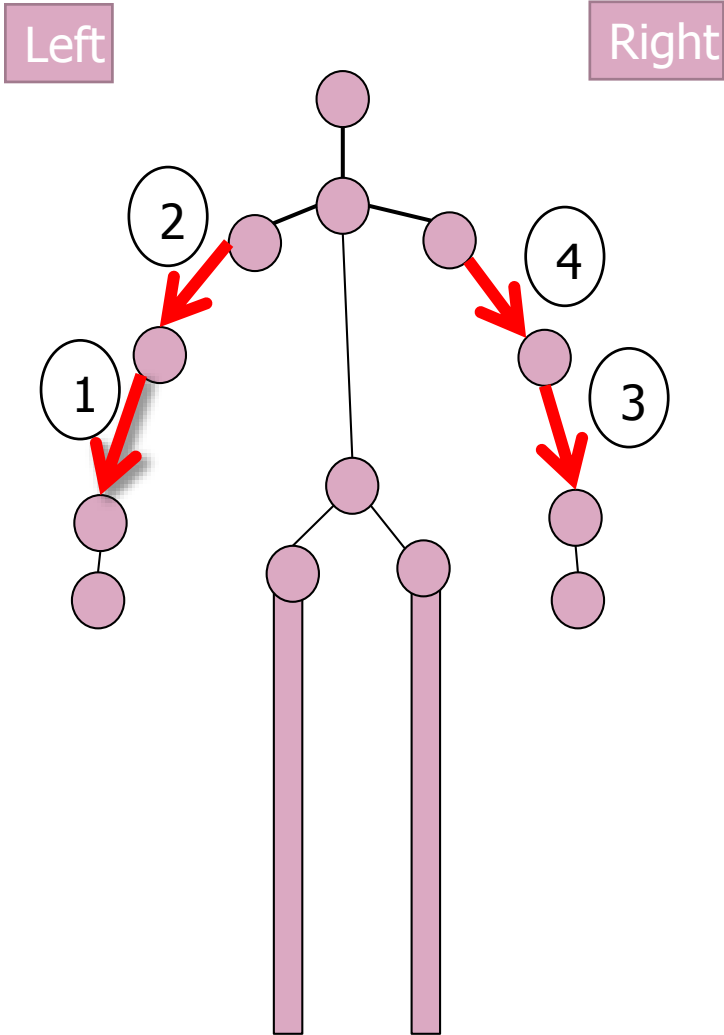
# Key Part 3: Gesture Recognition

UNIVERSITY  
OF MANITOBA





# Key Part 3: Gesture Recognition



## Features(Angles) Tolerance Errors

Angle 1:  $\langle \text{vec1}, X \rangle$  (10,20)

Angle 2:  $\langle \text{vec2}, X \rangle$  (10,20)

Angle 3:  $\langle \text{vec3}, X \rangle$  (10,20)

Angle 4:  $\langle \text{vec4}, X \rangle$  (10,20)

Angle 5:  $\langle \text{vec1}, Z \rangle$  (10,20)

Angle 6:  $\langle \text{vec2}, Z \rangle$  (10,20)

Angle 7:  $\langle \text{vec3}, Z \rangle$  (10,20)

Angle 8:  $\langle \text{vec4}, Z \rangle$  (10,20)



# Key Part 3: Gesture Recognition

Gesture Recognition Thread

Obtain Skeleton Data

Read In the Gesture Design

Calculate the Angles

Store These Classifiers

Processing Loop

Calculate the Distance

Examine the Confidential Region

Satisfy

Send the user-defined Message with GestureID

## ❖ More Advanced Techs

- ❖ DTW
- ❖ HMM
- ❖ LPP
- ❖ TDNN

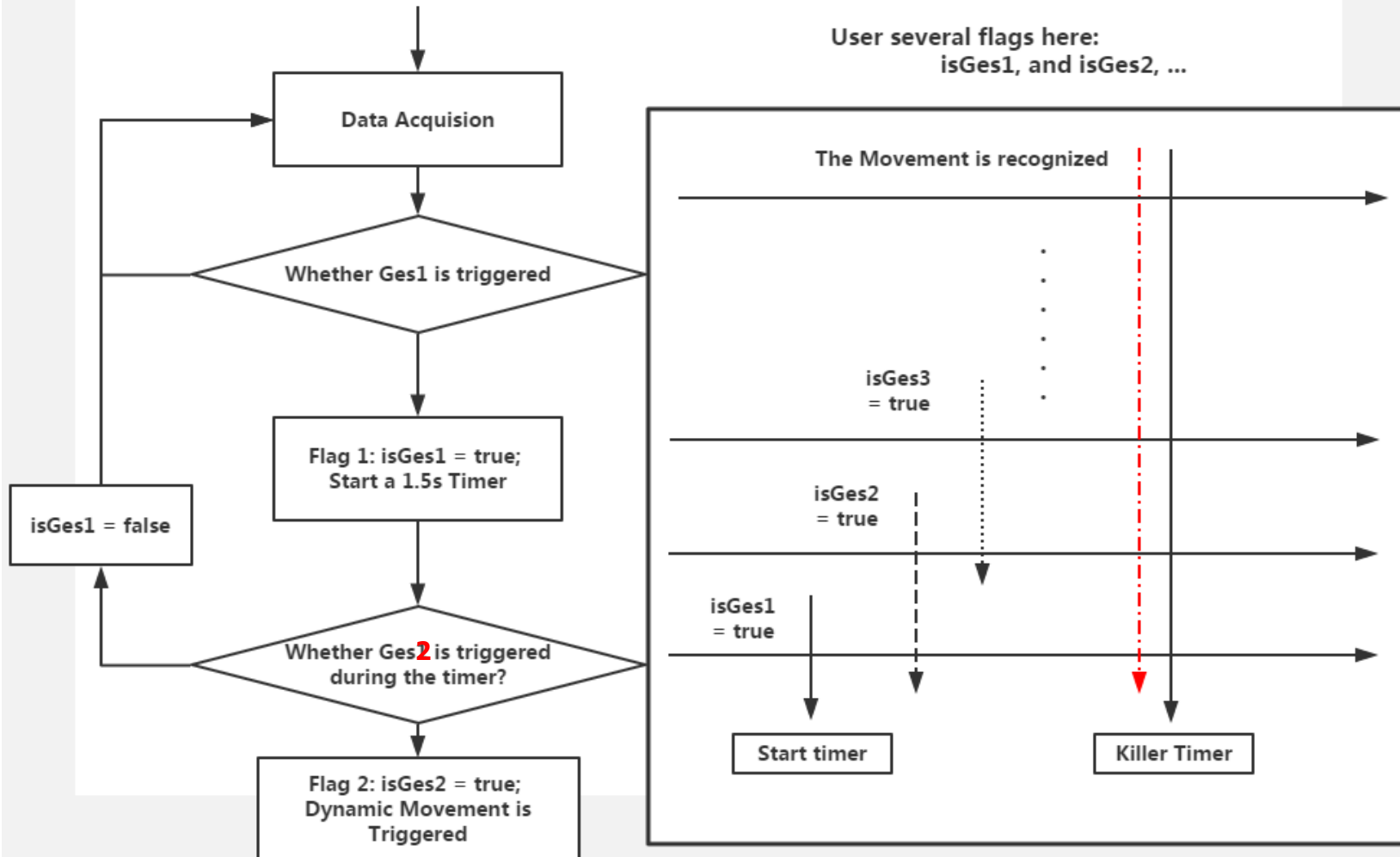


# Key Part 4: Motion Recognition

UNIVERSITY  
OF MANITOBA

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

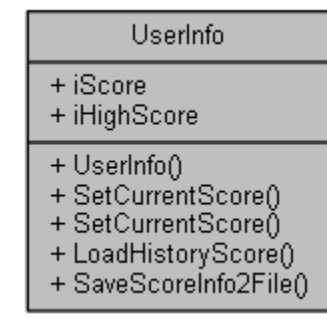
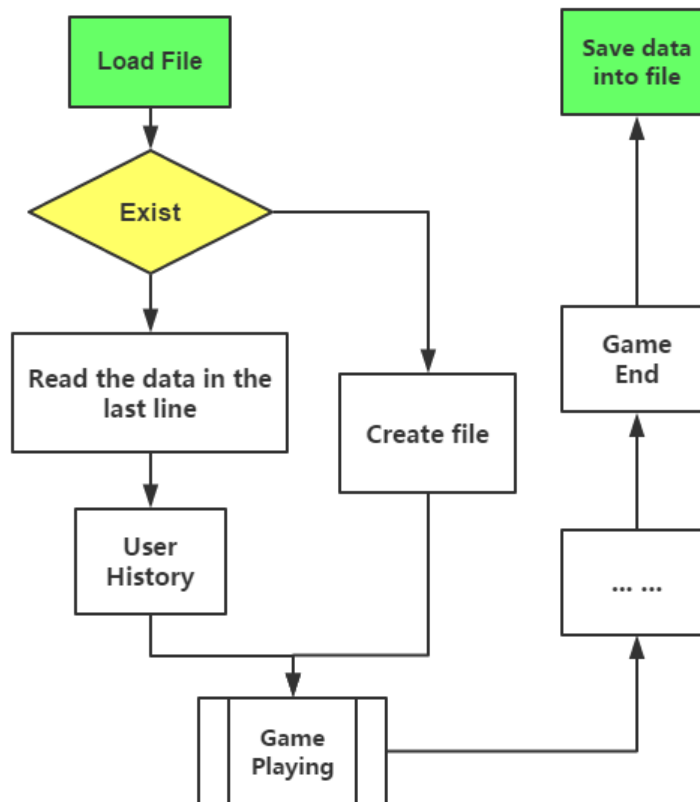
Assumption: Movement 1 consists of two gestures, Ges1 and Ges2





# Key Part 5: Player Information

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





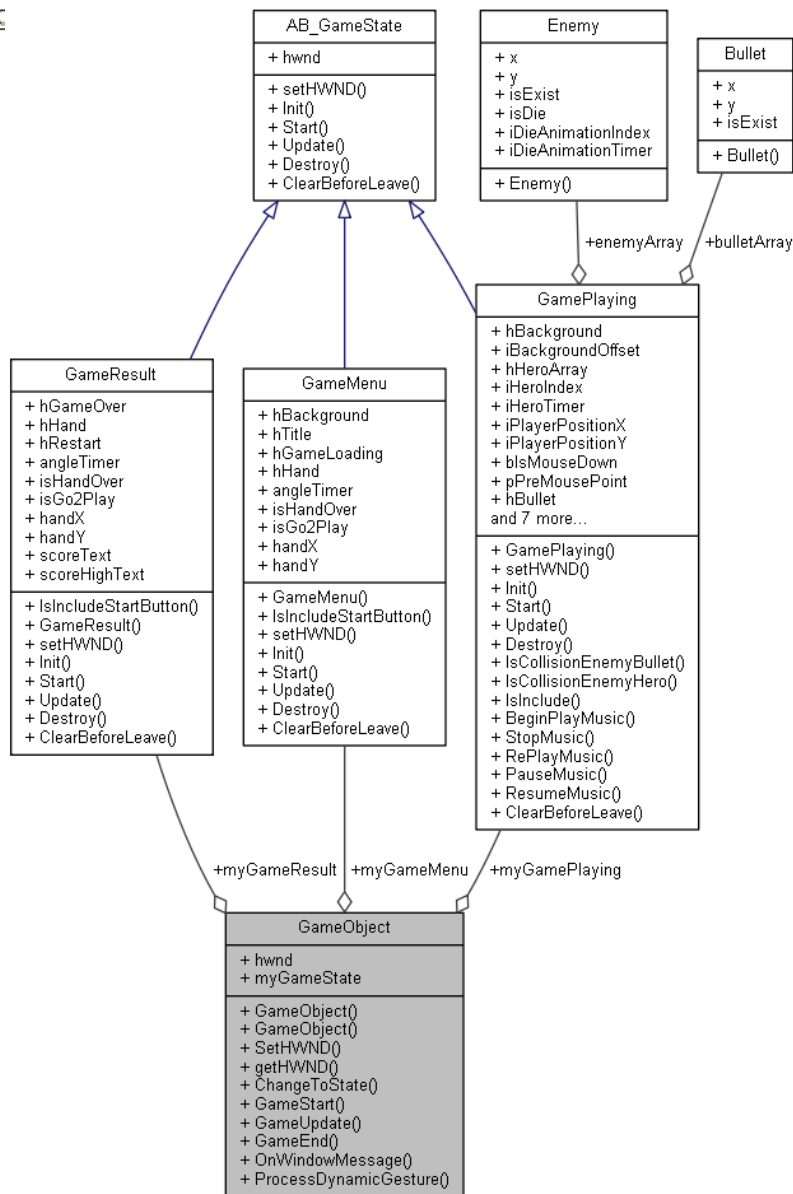
- **UML Diagram**
  - (9 diagram: sequence, class, use case, activity ... )
  - **Class Diagram (Doxygen)**
  
- **Source Code in VS2010**





# Class Diagram

UNIVERSITY



[Link To Code Analysis](#)

[Link To Visual Studio](#)

```

class SensorObject {
- m_pNuiSensor
- m_hStreamHandle
- m_hFrameReadyEvent
- m_allSkeletonData
- m_allSkeDataOnScreen
- m_paused
- isKinectFined
- isSkeletonEnable
- isSkeletonFound
- cScreenWidth
- cScreenHeight

+ SensorObject()
+ ~SensorObject()
+ Initialize()
+ StartStreams()
+ UpdateStreams()
+ PauseStream()
+ ProcessCommand()
+ getKinectState()
+ isKinectLinkCorrect()
+ isSkeletonCorrect()
+ isSkeletonFoundOK()
+ getFrameReadyEvent()
+ getSkeletonPointData()
+ getSkeDataOnScreen()
+ getSkeDataOnScreen()
- ProcessSkeleton()
- clearAllSkeletonData()
}
  
```

```

class GestureControl {
- gestureArray
- gestureNum
- currentFrameAngles
- currentActiveGestureID

+ GestureControl()
+ ~GestureControl()
+ gestureDataUpdate()
+ getActiveGesID()
+ getGestureNumber()
+ clearAfterOneGame()
- loadGestureDesignFromFile()
- calculateCurrentAngles()
- calculateCurrentAngles()
- findMatchGesId()
- isWithinErrorDistance()
- calAngleBetween2Vec()
- calAngleBetween2Vec()
- calAngleBetweenVecAndX()
- calAngleBetweenVecAndY()
- calAngleBetweenVecAndNegativeZ()
}
  
```



# Run & Test the Demo

---

UNIVERSITY  
OF MANITOBA

# Q&A