Wii Remote
Strategies and Algorithms

Steve Rabin
Principal Software Engineer
Software Development Support Group
Agenda

- Pointer functionality
- Accelerometers
  - Understanding accelerometers
  - Gesture recognition algorithms
    - Wii Sports case study
  - Steering
- Wii Balance Board
3D Pointing: Targeting

- Aiming or Choosing
  - Onscreen feedback required

- Hand Shakiness is an Issue
  - Use KPAD smoothing
  - KPADSetPosParam(chan, play, sensitivity);
    - <play> should be between 0 and 0.05 (full range [0,1])
    - Find ideal settings with "kpadsample" in SDK
    - Only adjust after KPADSetPosPlayMode has been decided (Tight vs Loose)
3D Pointing: Distance/Twisting/Gestures

• **Distance**
  - Absolute distance can be computed
    • But only use relative distance
  - Could use distance to zoom
  - Smooth with KPADSetDistParam()

• **Twisting**
  - Smooth with KPADSetHoriParam()
  - Could also use accelerometer

• **Gestures**
  - Drawing symbols for spell casting
  - Use directional flicks to augment actions
Accelerometers

+/-2.1G

+/-3.4G
Understanding Accelerometers

1. Gravity is a force
   - (an acceleration)

2. Start and stop sweep movement
   - x-axis: Acceleration followed by deceleration
   - y-axis: Only affected by gravity
   - z-axis: Arm imparts a centripetal force on remote

3. Simulated drum hit
   - x-axis: Not affected much
   - y-axis: Gravity + acceleration/deceleration
   - z-axis: Centripetal force
Accelerometer Lessons

- Acceleration ≠ velocity ≠ position
- Accelerometers always detect gravity
- Movement creates acceleration and deceleration
- Accelerometers detect *change* in velocity
  - Constant speed = no acceleration!
- Some rotations can't be detected by accelerometers
- Accelerometers are amazingly accurate & precise
  - Hand Shakiness needs to be dealt with
Accelerometer Applications

Gesturing

Steering
Accelerometers: Advice for Designing Gestures

• Don't wear out the player
  – Keep frequency/duration of vigorous gestures low

• Common issues
  – Missed recognition
    • Not sensitive enough
    • Player not holding controller correctly
  – Incorrect recognition
    • Gestures are too similar to each other
    • Use more context sensitive gestures
  – False positives
    • Expected gesture is too subtle or too similar to gravity
    • Use context sensitive gestures
Accelerometers:
Difficult to Track 3D Position

• Accelerometers measure acceleration
  – Not velocity or position
  – But, double integral of acceleration is position!
• Difficult to decouple gravity from movement
  – People hold controller differently
  – Orientation changes over duration of movement
  – Complicated algorithms can make educated guesses at the influence of gravity
  – Error makes this extremely difficult
• No known method to reliably track position only with accelerometers
Preprocess Signal to Estimate True Magnitude

- Wii Remote detects +/-3.4G
  - Easy to max out acceleration
Preprocess Signal to Estimate True Magnitude

- Wii Remote detects +/-3.4G
  - Easy to max out acceleration
Estimate True Magnitude: Spline Method

- Use spline to estimate actual magnitude
  - Hermite spline (C1 continuity)
  - Bezier spline (C2 continuity)
Estimate True Magnitude: Spline Method

- Use spline to estimate actual magnitude
  - Hermite spline (C1 continuity)
  - Bezier spline (C2 continuity)
Estimate True Magnitude: Spline Method

- Use a spline to estimate actual magnitude
  - Hermite spline (C1 continuity)
  - Bezier spline (C2 continuity)
Estimate True Magnitude: Spline Method

• Might need to estimate as data comes in
  – Option #1: Predict end control point
Estimate True Magnitude: Spline Method

• Might need to estimate as data comes in
  – Option #1: Predict end control point
Estimate True Magnitude: Spline Method

- Might need to estimate as data comes in
  - Option #1: Predict end control point
Estimate True Magnitude: Spline Method

• Might need to estimate as data comes in
  – Option #1: Predict end control point
    • Must guess at width... But how wide?
Estimate True Magnitude: Dampen Method

- Might need to estimate as data comes in
  - Option #2: Take delta, add to last, dampen
Estimate True Magnitude:
Dampen Method

• Might need to estimate as data comes in
  – Option #2: Take delta, add to last, dampen

\[ \text{Last} = \text{Last} + \Delta \times 0.85 \]
Estimate True Magnitude: Dampen Method

- Might need to estimate as data comes in
  - Option #2: Take delta, add, dampen

\[(\text{Last} + \text{Delta}) \times 0.85\]
Estimate True Magnitude: Dampen Method

- Might need to estimate as data comes in
  - Option #2: Take delta, add, dampen

\[
(\text{Last} + \Delta) \times 0.85
\]
Estimate True Magnitude: Dampen Method

- Might need to estimate as data comes in
  - Option #2: Take delta, add, dampen

\[(\text{Last} + \Delta) \times 0.85\]
Estimate True Magnitude: Dampen Method

- Might need to estimate as data comes in
  - Option #2: Choosing dampening value is difficult

Initial Delta = 2.5G
Dampening = 0.85
Estimate True Magnitude: Dampen Method

- Might need to estimate as data comes in
  - Option #2: Choosing dampening value is difficult

Initial Delta = 2.5G
Dampening = 0.90
Estimate True Magnitude: Dampen Method

• Might need to estimate as data comes in
  – Option #2: Choosing dampening value is difficult

Initial Delta = 2.5G
Dampening = 0.95
Estimate True Magnitude: Dampen Method

- Might need to estimate as data comes in
  - Option #2: Choosing dampening value is difficult
    (Initial delta and dampening determine period width)

Initial Delta = 2.5G
Dampening = 0.85
Estimate True Magnitude: Dampen Method

- Might need to estimate as data comes in
  - Option #2: Choosing dampening value is difficult
    (Initial delta and dampening determine period width)

Initial Delta = 2.5G
Dampening = 0.90
Estimate True Magnitude: Dampen Method

• Might need to estimate as data comes in
  – Option #2: Choosing dampening value is difficult
    (Initial delta and damping determine period width)

Initial Delta = 2.5G
Dampening = 0.95
Estimate True Magnitude: Dampen Method

- Might need to estimate as data comes in
  - Option #2: Choosing damping value is difficult (Initial delta and damping determine period width)

Initial Delta = 4.6G
Dampening = 0.95
Estimate True Magnitude: Dampen Method

• Might need to estimate as data comes in
  – Option #2: Choosing dampening value is difficult
    (Initial delta and dampening determine period width)

Initial Delta = 1.0G
Dampening = 0.95
## Width of Clamped Area

<table>
<thead>
<tr>
<th>Dampering</th>
<th>0.3G</th>
<th>0.5G</th>
<th>1.0G</th>
<th>1.5G</th>
<th>2.0G</th>
<th>2.5G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.995</td>
<td>25</td>
<td>31</td>
<td>37</td>
<td>39</td>
<td>40</td>
<td>41</td>
</tr>
<tr>
<td>0.99</td>
<td>13</td>
<td>18</td>
<td>24</td>
<td>26</td>
<td>27</td>
<td>28</td>
</tr>
<tr>
<td>0.98</td>
<td>6</td>
<td>10</td>
<td>15</td>
<td>17</td>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td>0.95</td>
<td>2</td>
<td>3</td>
<td>7</td>
<td>9</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>0.90</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>0.85</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>
Estimate True Magnitude: Predict Clamped Width

- Ultimately must predict clamped width in order to predict missing magnitude
  - For spline method or dampen method
  - Player/situation dependant
Estimate True Magnitude: Player Modeling

- AI statistical learning technique
- Track clamped length moving average on each axis
- Six moving averages to track
  - x-axis +, x-axis -, y-axis +, y-axis -, z-axis +, z-axis -
Detecting when Gestures Begin and End

- Player presses/releases button
  - Example: Drawing in the air

- Use centripetal force as a proxy
  - Moves cause centripetal force
    - Arm pivots at shoulder
    - Hand pivots at wrist
  - About 1.2G is a good threshold
    - Ignores non-gestures

~1.2G Threshold
Accelerometer Gesture Recognition: Simple vs Complex
Accelerometer Gesture Recognition: Simple Motion

- Axis-aligned
- Short duration
- Easy to detect
Accelerometer Gesture Recognition: Complex Motion

- Multi-axis
- Longer duration
- Difficult to detect 100%
- Difficult to detect early
Gesture Recognition:
Simple Motion—Hits, Swipes, and Stabs

- These movements are axis-aligned
  - Easy to detect (using thresholds)
  - Natural player movement, simple to do
A Tale of Two Drums
Drum Hit—Case Study #1

• Two aspects
  – Detect moment of impact
  – Detect strength of impact
Drum Hit—Case Study #1

- Detect moment of impact
  - 0.5G "Prep" threshold will figuratively "cock trigger"
Drum Hit—Case Study #1

- Detect moment of impact
  - 0.5G "Prep" threshold will figuratively "cock trigger"
  - Once ready, -1.3G threshold represents moment of impact
Drum Hit—Case Study #1

- Detect strength of impact
  - Construct window between "prep" time and "impact" time
  - Within window, integrate positive acceleration on z-axis

![Diagram showing acceleration on different axes](image-url)
Drum Solo!

"It was sounding great, but, I could of used a little more cowbell"
Drum Hit—Case Study #2

- What if we want positional data?
  - Show drum stick going up/down in-sync with Wii Remote
  - Use actual on-screen motion/velocity to determine hit strength (loudness)
Drum Hit—Case Study #2

• Study up/down acceleration
Drum Hit—Case Study #2

• Study up/down acceleration
Drum Hit—Case Study #2

• Study up/down acceleration
Drum Hit—Case Study #2

- Study up/down acceleration

![Diagram showing upswing and downswing with acceleration vectors.](image-url)
Drum Hit—Case Study #2

- Study up/down acceleration
Drum Hit—Case Study #2

• Study up/down acceleration
Drum Hit—Case Study #2

- Adjust values for gravity
Drum Hit—Case Study #2

- Adjust values for gravity
Drum Hit—Case Study #2

- Cut out values near zero

[Diagram showing various stages of a drum hit with labels for Normal Gravity, Upswing Speeding Up, Upswing Slowing Down, Downswing Speeding Up, Downswing Slowing Down, and Noise, with y-axis values marked at ±0.2G]
Drum Hit—Case Study #2

- Collapse values toward zero
Drum Hit—Case Study #2

- Pretend this acceleration is position!
Drum Hit—Case Study #2

- Don't let drumstick go through drum
Drum Hit—Case Study #2

• Use derivative of position as velocity (loudness)
Drum Hit—Case Study #2

- Compare to original motion

![Diagram showing y-axis with various motion events: Normal Gravity, Upswing Slowing Down, Speeding Up, Downswing Slowing Down, and Noise.](image)
Drum Hit—Case Study #2

• What looks wrong about this?
  – Motion not smooth (in first derivative) – cartoonish
    • Upward/downward swing starts moving instantaneously
    • Abrupt stop at top (accelerometer limits)
  – Loudness is correlated with Wii Remote motion, but inaccurate (since actually derivative of acceleration)

Loudness = Slope
Drum Hit—Case Study #2

• Summary
  – Acceleration as position works in limited situations
  – Must constrain from going in the wrong direction
  – Works OK for drum hits and boxing (but cartoonish)
Complex Gesture Recognition: Five Techniques
Complex Gesture Recognition: Preprocessing the Signal

- Example from handwriting recognition
  - Normalize size
  - Normalize length/speed
Complex Gesture Recognition: First Step—Preprocessing

Massage input to look consistent/uniform
Complex Gesture Recognition:
First Step—Preprocessing

Massage input to look consistent/uniform

1. (optional) Remove gravity from all axes
   • Gravity problematic
   • Removes small movement noise
Complex Gesture Recognition: First Step—Preprocessing

Massage input to look consistent/uniform

1. (optional) Remove gravity from all axes
   - Gravity problematic
   - Removes small movement noise

2. Remove parts with no acceleration

3. Normalize length

---

Raw       Step 1   Step 2,3   Step 4
Complex Gesture Recognition: First Step—Preprocessing

Massage input to look consistent/uniform

1. (optional) Remove gravity from all axes
   • Gravity problematic
   • Removes small movement noise
2. Remove parts with no acceleration
3. Normalize length
4. Normalize intensity

![Diagram showing steps](image)
Complex Gesture Recognition: Technique 1—Nearest Neighbor

- Compare player input to database of examples
Complex Gesture Recognition: Technique 1—Nearest Neighbor

- Compare player input to database of examples
Complex Gesture Recognition: Technique 1—Nearest Neighbor

• Compare player input to database of examples
Complex Gesture Recognition:
Technique 1—Nearest Neighbor

- Compare player input to database of examples
Complex Gesture Recognition: Technique 1—Nearest Neighbor

- Compare player input to database of examples
- **Lowest error is match**
Complex Gesture Recognition: Technique 1—Nearest Neighbor

- Compare player input to database of examples
- Lowest error is match

<table>
<thead>
<tr>
<th>Player Swing</th>
<th>Swing Left 1</th>
<th>Swing Left 2</th>
<th>Swing Left 3</th>
<th>Swing Right 1</th>
<th>Swing Right 2</th>
<th>Swing Right 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>
Complex Gesture Recognition: Technique 1—Nearest Neighbor

- Compare player input to database of examples
- **Lowest error is match**

![Player Swing]

- Swing Left 1
- Swing Left 2
- Swing Left 3
- Swing Right 1
- Swing Right 2
- Swing Right 3
Complex Gesture Recognition: Technique 1—Nearest Neighbor

- Compare player input to database of examples
- Lowest error is match
Complex Gesture Recognition: Technique 1—Nearest Neighbor

- Compare player input to database of examples
- **Lowest error is match**

<table>
<thead>
<tr>
<th>Swing Left 1</th>
<th>Swing Left 2</th>
<th>Swing Left 3</th>
<th>Swing Right 1</th>
<th>Swing Right 2</th>
<th>Swing Right 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error = 11</td>
<td>Error = 8</td>
<td>Error = 6</td>
<td>Error = 40</td>
<td>Error = 35</td>
<td>Error = 42</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>7</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7</td>
</tr>
</tbody>
</table>

Player Swing

Error = 11 Error = 8 Error = 6 Error = 40 Error = 35 Error = 42
Complex Gesture Recognition: Technique 1—Nearest Neighbor

- Compare player input to database of examples
- **Lowest error is match**

<table>
<thead>
<tr>
<th>Swing Left 1</th>
<th>Swing Left 2</th>
<th>Swing Left 3</th>
<th>Swing Right 1</th>
<th>Swing Right 2</th>
<th>Swing Right 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error = 11</td>
<td>Error = 8</td>
<td>Error = 6</td>
<td>Error = 40</td>
<td>Error = 35</td>
<td>Error = 42</td>
</tr>
<tr>
<td>3 0 2 2</td>
<td>1 0 1 1</td>
<td>1 0 1 0</td>
<td>7 5 4 3</td>
<td>0 4 3 2</td>
<td>8 7 6 4</td>
</tr>
</tbody>
</table>

**Player Swing**
Complex Gesture Recognition: Technique 1—Nearest Neighbor

- Compare player input to database of examples
- Lowest error is match (ROOT MEAN SQUARE!)
Complex Gesture Recognition: Technique 1—Nearest Neighbor

- Compare player input to database of examples
- Lowest error is match *(ROOT MEAN SQUARE!)*

<table>
<thead>
<tr>
<th></th>
<th>Swing Left 1</th>
<th>Swing Left 2</th>
<th>Swing Left 3</th>
<th>Swing Right 1</th>
<th>Swing Right 2</th>
<th>Swing Right 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Error = 23</strong></td>
<td>9 0 4 4 1 0</td>
<td>1 0 1 1 0 1</td>
<td>1 0 1 1 0 1</td>
<td>16 36 4 25 16 49</td>
<td>49 36 16 9 0 16</td>
<td>4 49 36 16 49 9</td>
</tr>
<tr>
<td><strong>Error = 10</strong></td>
<td>1 0 1 1 1 4</td>
<td>4 1 0 1 1 4</td>
<td>9 10 1 9 10 1</td>
<td>49 49 16 16 16 16</td>
<td>49 36 16 9 0 16</td>
<td>4 49 36 16 49 9</td>
</tr>
<tr>
<td><strong>Error = 12</strong></td>
<td>0 0 1 0 1 1</td>
<td>0 0 1 0 1 1</td>
<td>0 0 1 0 1 1</td>
<td>25 25 16 16 16 16</td>
<td>25 25 16 16 9 0</td>
<td>0 0 1 0 1 1</td>
</tr>
<tr>
<td><strong>Error = 195</strong></td>
<td>49 49 16 16 16 16</td>
<td>49 49 16 16 16 16</td>
<td>49 49 16 16 16 16</td>
<td>49 49 16 16 16 16</td>
<td>49 49 16 16 16 16</td>
<td>49 49 16 16 16 16</td>
</tr>
<tr>
<td><strong>Error = 191</strong></td>
<td>49 36 16 9 0 16</td>
<td>49 36 16 9 0 16</td>
<td>49 36 16 9 0 16</td>
<td>49 36 16 9 0 16</td>
<td>49 36 16 9 0 16</td>
<td>49 36 16 9 0 16</td>
</tr>
<tr>
<td><strong>Error = 203</strong></td>
<td>4 49 36 16 49 9</td>
<td>4 49 36 16 49 9</td>
<td>4 49 36 16 49 9</td>
<td>4 49 36 16 49 9</td>
<td>4 49 36 16 49 9</td>
<td>4 49 36 16 49 9</td>
</tr>
</tbody>
</table>

Error calculation: Root Mean Square
Complex Gesture Recognition: Technique 1—Nearest Neighbor

- Compare player input to database of examples
- Lowest error is match
Complex Gesture Recognition: Technique 1—Nearest Neighbor

- Compare player input to database of examples
- Lowest error is match
Complex Gesture Recognition: Technique 1—Nearest Neighbor

- Compare player input to database of examples
- Lowest error is match

Player Swing

<table>
<thead>
<tr>
<th>Swing Left 1</th>
<th>Swing Left 2</th>
<th>Swing Left 3</th>
<th>Swing Right 1</th>
<th>Swing Right 2</th>
<th>Swing Right 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error = 70</td>
<td>Error = 59</td>
<td>Error = 35</td>
<td>Error = 97</td>
<td>Error = 112</td>
<td>Error = 153</td>
</tr>
<tr>
<td>0 0 4 16</td>
<td>4 4 1 16</td>
<td>4 4 1 16</td>
<td>4 16 25 1</td>
<td>4 16 25 1</td>
<td>4 16 25 1</td>
</tr>
<tr>
<td>4 25 4</td>
<td>4 1 25</td>
<td>1 25 4</td>
<td>0 1 16 4</td>
<td>0 1 16 4</td>
<td>0 1 16 4</td>
</tr>
<tr>
<td>4 1 16 4</td>
<td>4 1 25 4</td>
<td>1 25 4</td>
<td>0 1 16 4</td>
<td>0 1 16 4</td>
<td>0 1 16 4</td>
</tr>
<tr>
<td>0 0 4 16</td>
<td>4 25 4</td>
<td>4 1 25 4</td>
<td>0 1 16 4</td>
<td>0 1 16 4</td>
<td>0 1 16 4</td>
</tr>
</tbody>
</table>
Complex Gesture Recognition: Technique 1—Nearest Neighbor

• Compare player input to database of examples
• Lowest error is match
• Large error = no match

<table>
<thead>
<tr>
<th>Player Swing</th>
<th>Error = 70</th>
<th>Error = 59</th>
<th>Error = 35</th>
<th>Error = 97</th>
<th>Error = 112</th>
<th>Error = 153</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swing Left 1</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Swing Left 2</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Swing Left 3</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Swing Right 1</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Swing Right 2</td>
<td>4</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Swing Right 3</td>
<td>4</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

Error values indicate the difference between the player's swing and the closest match in the database.
Complex Gesture Recognition: Technique 1—Nearest Neighbor

- General algorithm to match against database
  - Not many examples needed
  - Preprocess data for best matching
- Can constantly monitor input stream
- Player could supply examples
Complex Gesture Recognition: Technique 2—Neural Network

• Black box that tells you the answer
• You train it with 100s or 1000s of examples
  – Network generalizes to examples
Complex Gesture Recognition: Technique 3—Cheat

- Adapt a complex gesture into a series of simple gestures
- Sequences of axis-aligned movements
  - Easier to detect
  - Train the player
Complex Gesture Recognition: Technique 4—LiveMove Middleware

At runtime, your LiveMove recognizer understands any move your players make with a motion sensitive controller such as the Wii™ controller.

LiveMove can recognize:
- A tennis back-hand shot
- An overhead lasso
- Drawing a revolver
- A sword thrust or parry
- Letters and numbers
- Anything you can think of!

At development time, LiveMove lets designers create a LiveMove recognizer in minutes just by performing motions. No code. No scripts.

www.ailive.net
support@ailive.net
Complex Gesture Recognition: Technique 5—Use your Brain

1. Study the move(s) you want to detect
2. Identify its features
   • Is there a single feature that is unique?
   • Is it consistent no matter who does the gesture?
3. Write custom detection code for the single gesture
   • Various threshold tests in sequence
   • Threshold triggering relative to other axes
4. Discern the differences between two gestures
   • In cases where it's one or the other
Complex Gesture Recognition: Wii Sports Tennis Case Study

- Recognize any swing
- Recognize left or right swing
- Recognize topspin, backspin, no spin
- Recognize underhand or overhand
- Recognize hard or soft hit
Complex Gesture Recognition: Recognize Swing

• Threshold on z-axis
  – Something like 1.2G to 1.5G
Complex Gesture Recognition:
Left or Right Swing

Left Swing (counterclockwise)

Right Swing (clockwise)
Complex Gesture Recognition: Left or Right Swing

- Orientation of controller doesn't matter!
- Increase recognition:
  - Predict correct swing
  - Make incorrect swings require larger threshold
    - Avoids mistaking "prep" as swing
Complex Gesture Recognition: Topspin, No Spin, or Backspin

Topspin

No Spin

Backspin
Complex Gesture Recognition: Underhand or Overhand

- Look at z-axis before swing
  - Negative = Overhand
  - Positive = Underhand
Complex Gesture Recognition: Hard Hit or Soft Hit

Hard Hit

Soft Hit
Complex Gesture Recognition:
Hard Hit or Soft Hit

- Position
  - Change in Position
- Velocity
  - Change in Velocity
- Acceleration
  - Change in Acceleration?
Complex Gesture Recognition: Hard Hit or Soft Hit

- Position
  - Change in Position
- Velocity
  - Change in Velocity
- Acceleration
  - Change in Acceleration
- Jerk
Complex Gesture Recognition: Wii Sports Tennis Timeline

- Sequence of events during a swing and hit
Complex Gesture Recognition:
Wii Sports Tennis Timeline

- Time A: Swing started by player
Complex Gesture Recognition: Wii Sports Tennis Timeline

- Time B: Detect left or right swing
Complex Gesture Recognition:
Wii Sports Tennis Timeline

- Time B: Detect underhand or overhand
Complex Gesture Recognition: Wii Sports Tennis Timeline

- Time B: Start animation (left/right, over/under)
Complex Gesture Recognition: Wii Sports Tennis Timeline

- Time C: Racket collides with ball
Complex Gesture Recognition: Wii Sports Tennis Timeline

- Time D: Velocity and spin recognized
Complex Gesture Recognition: Wii Sports Tennis Timeline

- High velocity and backspin
Complex Gesture Recognition: Wii Sports Tennis Timeline

- Average speed with no spin
Complex Gesture Recognition: Wii Sports Tennis Timeline

- Velocity and spin are detected late
Complex Gesture Recognition: Wii Sports Tennis Timeline

- Interpolate ball to desired trajectory
Accelerometer Applications: Steering
Steering and Rotating

- Robust and reliable
- Various orientations
  - Sideways / Wii Wheel
  - Paper airplane
  - Flight stick
Steering and Rotating: Desired Angles

- 0°
- 45°
- 90°
- -45°
- -90°
Steering and Rotating: Desired Angles

-0°
-45°
-90°
0°
45°
90°
Steering and Rotating: Desired Angles

-90°  0°  90°
-45°  45°
Steering and Rotating: Desired Angles

0°
-90°
-45°
45°
90°
Steering and Rotating: Angle Conversion

- Wrong way
  - Multiply z-axis by 90 degrees
Steering and Rotating: Angle Conversion

- Wrong way (multiply z-axis by 90 degrees)
  - Close, but causes "swerving" near zero degrees
Steering and Rotating: Angle Conversion

• Correct Way
  – Use trigonometry (sin or cos)
Steering and Rotating: Trigonometry Visualization

$G$ (gravity)
Steering and Rotating: Trigonometry Visualization

y-axis

z-axis

y-axis
Steering and Rotating: Trigonometry Visualization

y-axis

z-axis

G (gravity)
Steering and Rotating: Trigonometry Visualization

\[ G = \sqrt{y\text{Axis}\text{Acceleration}^2 + z\text{Axis}\text{Acceleration}^2} \]
Steering and Rotating: Trigonometry Visualization

\[ G = \sqrt{y\text{AxisAcceleration}^2 + z\text{AxisAcceleration}^2} \]

G (gravity) ≠ 1.0!
Steering and Rotating: Trigonometry Visualization

y-axis

z-axis

\( G \) (gravity)

\( \theta \)
Steering and Rotating: Trigonometry Visualization

\[ \sin(\theta) = \frac{\text{opposite}}{\text{hypotenuse}} = \frac{\text{zAxisAcceleration}}{\sqrt{\text{yAxisAcceleration}^2 + \text{zAxisAcceleration}^2}} \]
Steering and Rotating: Trigonometry Visualization

\[ \theta = \arcsin \left( \frac{z\text{AxisAcceleration}}{\sqrt{y\text{AxisAcceleration}^2 + z\text{AxisAcceleration}^2}} \right) \]
Avoiding Jitter in Steering

• Player's hands are shaky
  – Smooth out accelerometer data
  – KPADSetAccParam(chan, play, sensitivity);
    • <play> should be between 0 and 0.05
    • Note that these are smoothed independently for each axis
WPAD vs KPAD

- **WPAD**
  - Low level
  - y-axis is forwards
  - No smoothing

- **KPAD**
  - High level
  - z-axis is forwards
  - Offers smoothing
Pointing Summary

- Perfect for aiming or selecting
- Capable of
  - 2D position
  - Distance
  - Twisting
- Use KPAD library to smooth
  - 2D position
  - Horizontal (twisting)
  - Distance
Accelerometer Summary: Gesture Recognition

- Simple vs Complex
  - Complex takes more development effort and tuning
  - Complex harder to achieve 100% accuracy
  - Try to discern between two options – use your brain!
- Adapt game design to make gesture recognition robust
- Make use of velocity
Accelerometer Summary: Steering

• Remember to use trigonometry
  – Swerving could mean it was implemented wrong
• Use KPAD to smooth values
Wii Balance Board
Wii Balance Board

• Four "balance sensors"
  – Top left, Top right, Bottom left, Bottom right
  – Measures amount of change in pressure
  – Must be set to "zero point", like a typical scale

• Simple function WBCRead() returns total weight measurement from combined sensors
  – Use WBCGetTGCWeight() to correct for temperature and gravitational acceleration
Wii Balance Board

• Download the Wii Balance Board package from WarioWorld.com

• Won't be sold separately
  – Sold only with Wii Fit
  – Your game must work with or without the Wii Balance Board
Questions?

Ask me during the reception/breaks
Or e-mail support@noa.com