Wii Remote Strategies and Algorithms

Steve Rabin
Senior Software Engineer
Software Development Support Group

Agenda

- Pointer functionality
- Accelerometers
  - Understanding accelerometers
  - Gesture recognition algorithms
    - Wii Sports case study
  - Steering

3D Pointing: Targeting

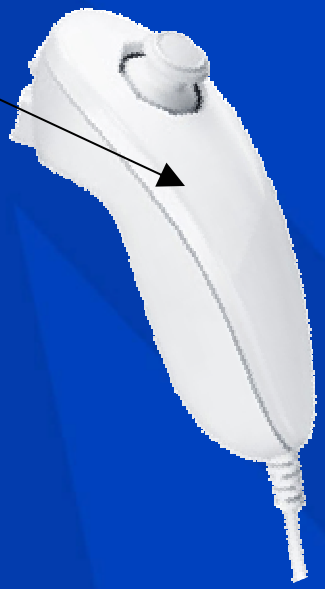
- Aiming or Choosing
  - Onscreen feedback required
- Hand Shakiness is an Issue
  - Use KPAD smoothing
  - Find ideal settings with "kpadsample" in SDK
  - KPADSetPosParam(chan, play, sensitivity);
    - <play> should be between 0 and 0.05 (full range [0,1])

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

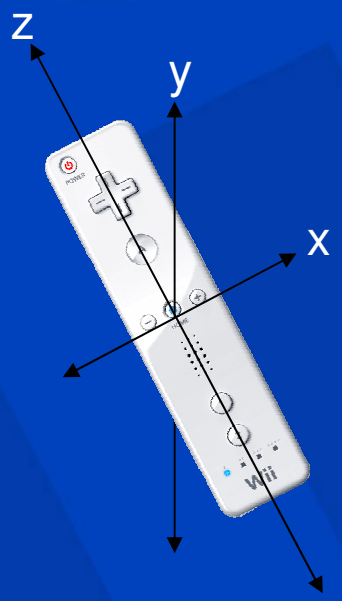


Nunchuk

+/-3.4G

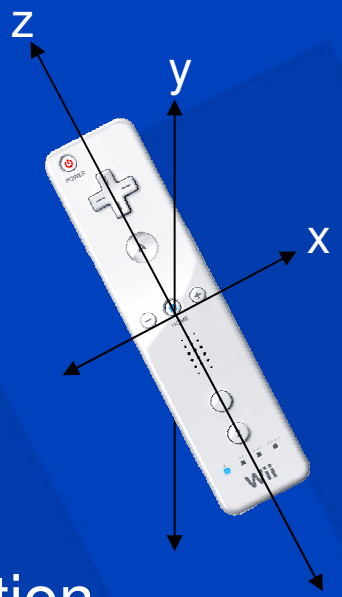
Wii Remote

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

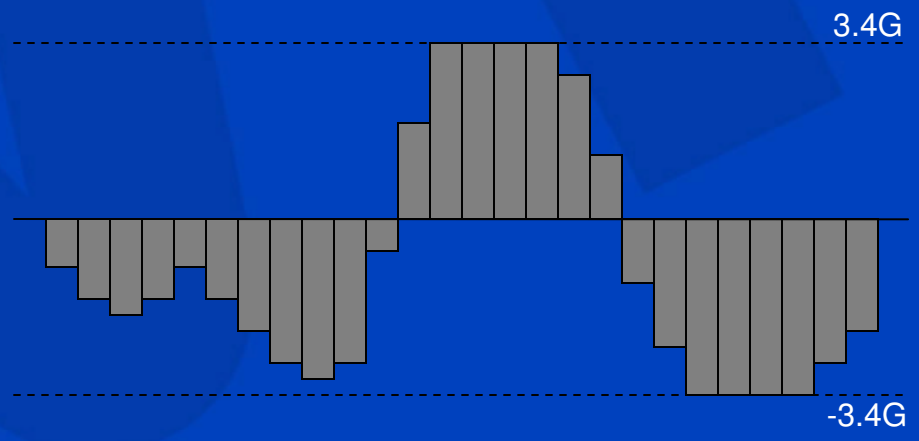
- Don't wear out the player
  - Keep frequency/duration of vigorous gestures low
- Common issues
  - Missed recognition
    - Not sensitive enough
    - Player not holding controller correctly
  - Incorrect recognition
    - Gestures are too similar to each other
    - Use more context sensitive gestures
  - 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

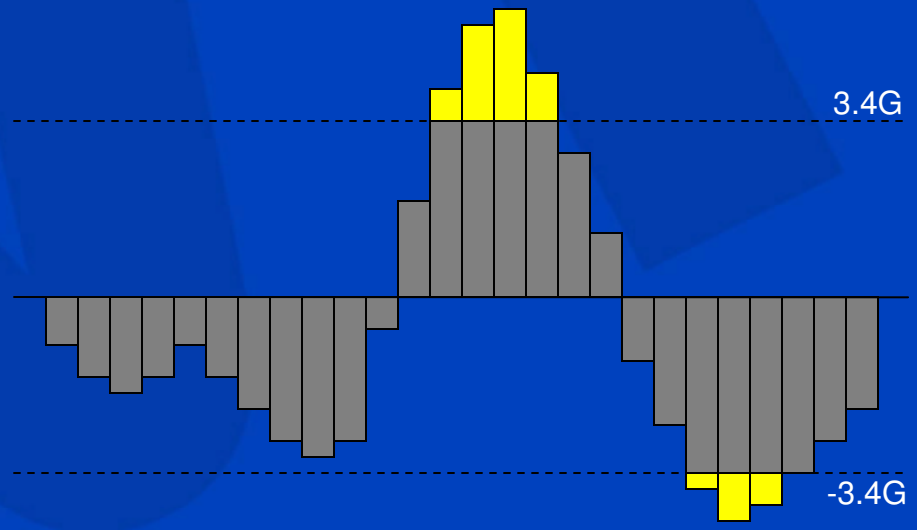
Preprocess Signal to Estimate True Magnitude

- Wii Remote detects +/-3.4G
  - Easy to max out acceleration



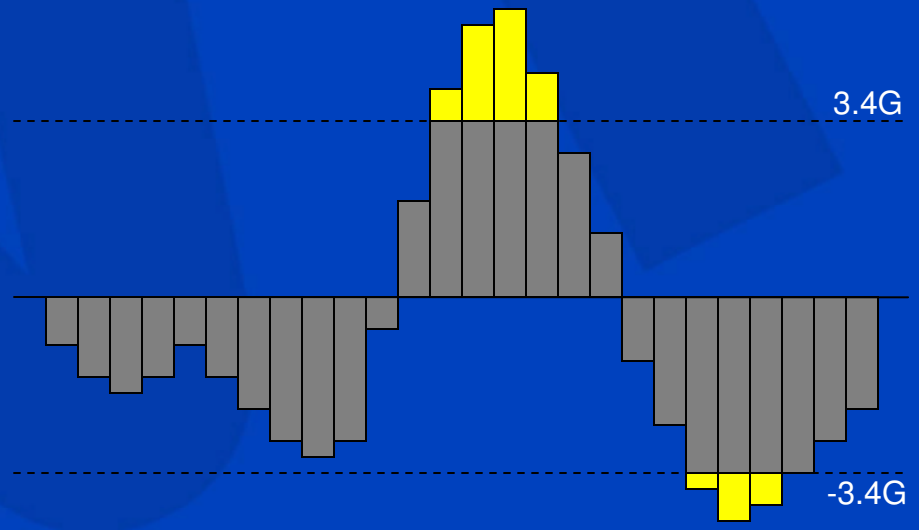
Preprocess Signal to Estimate True Magnitude

- Wii Remote detects +/-3.4G
  - Easy to max out acceleration



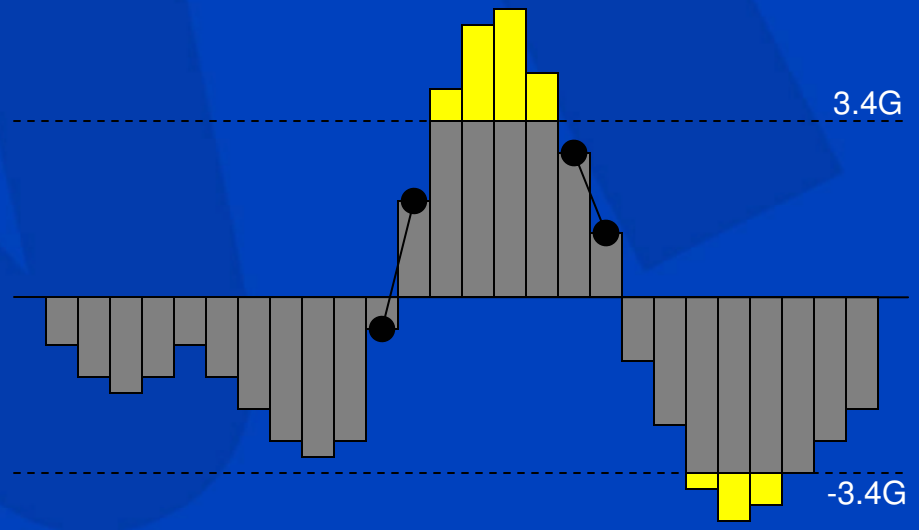
Preprocess Signal to Estimate True Magnitude

- Use spline to estimate actual magnitude
  - Hermite spline (C1 continuity)
  - Bezier spline (C2 continuity)



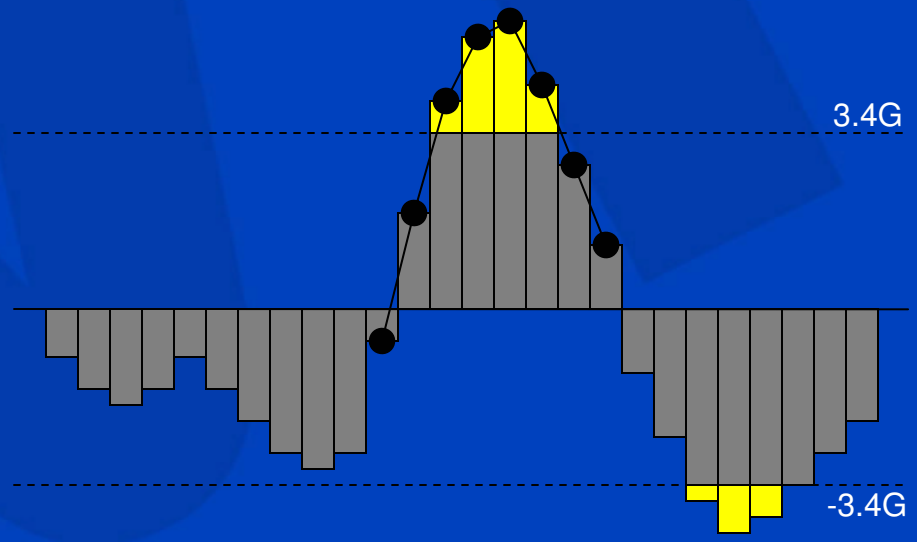
Preprocess Signal to Estimate True Magnitude

- Use spline to estimate actual magnitude
  - Hermite spline (C1 continuity)
  - Bezier spline (C2 continuity)



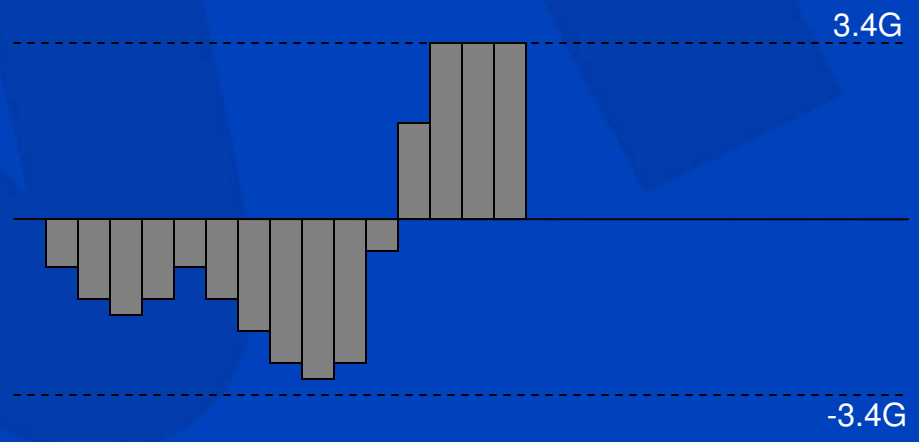
Preprocess Signal to Estimate True Magnitude

- Use spline to estimate actual magnitude
  - Hermite spline (C1 continuity)
  - Bezier spline (C2 continuity)



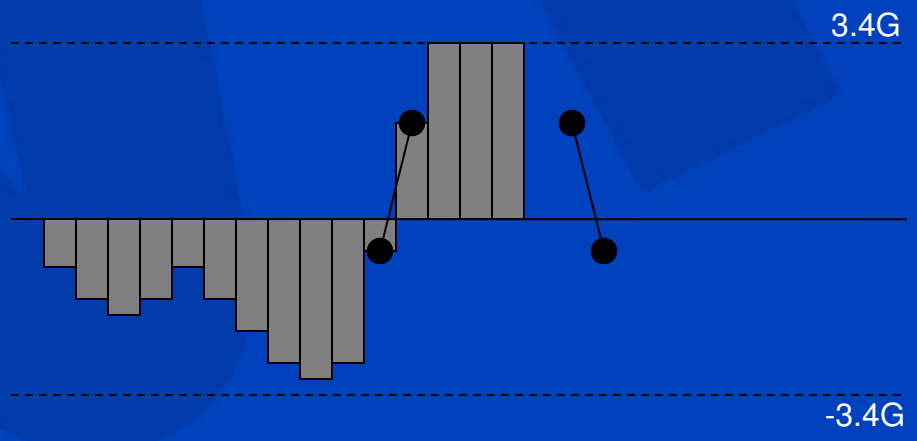
Preprocess Signal to Estimate True Magnitude

- Might need to estimate as data comes in
  - Requires predicting end control point



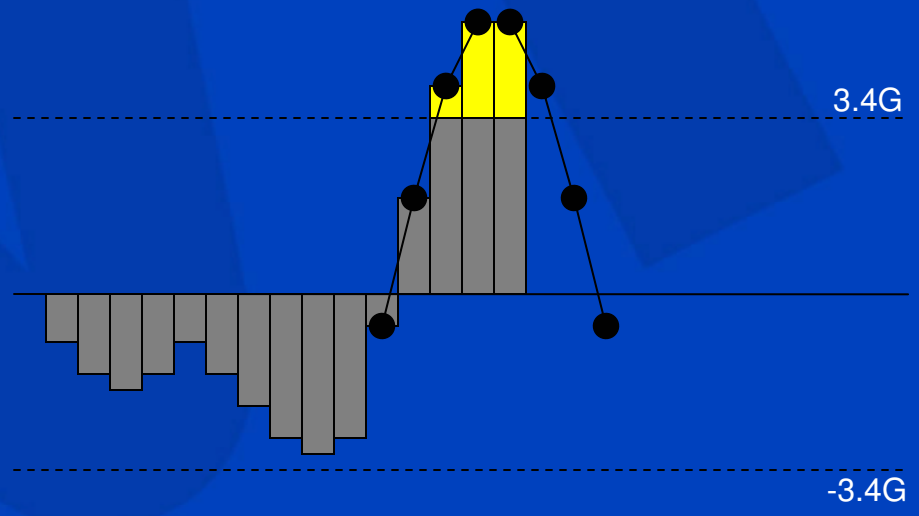
Preprocess Signal to Estimate True Magnitude

- Might need to estimate as data comes in
  - Requires predicting end control point



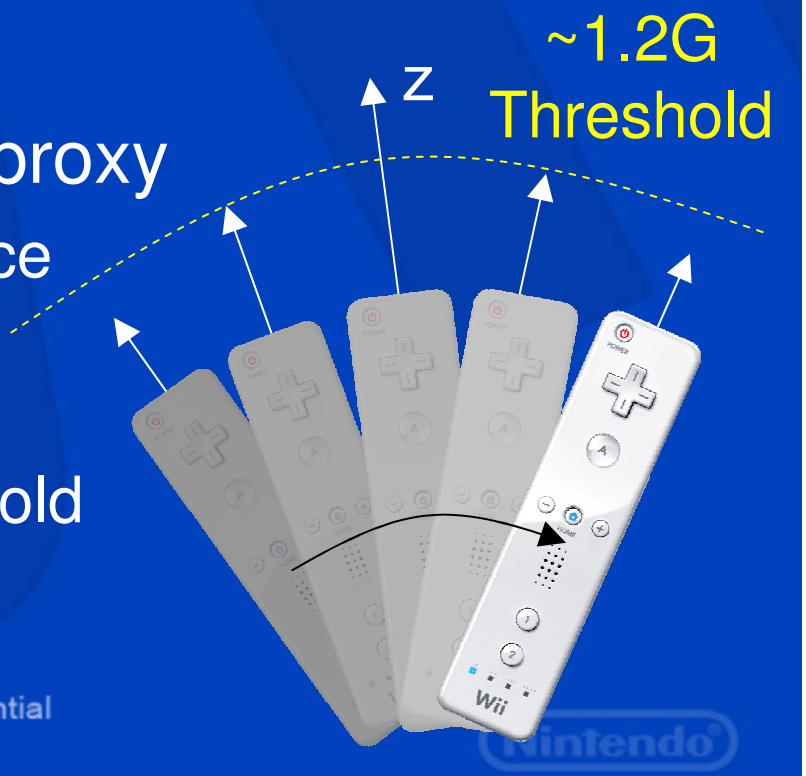
Preprocess Signal to Estimate True Magnitude

- Might need to estimate as data comes in
  - Requires predicting end control point

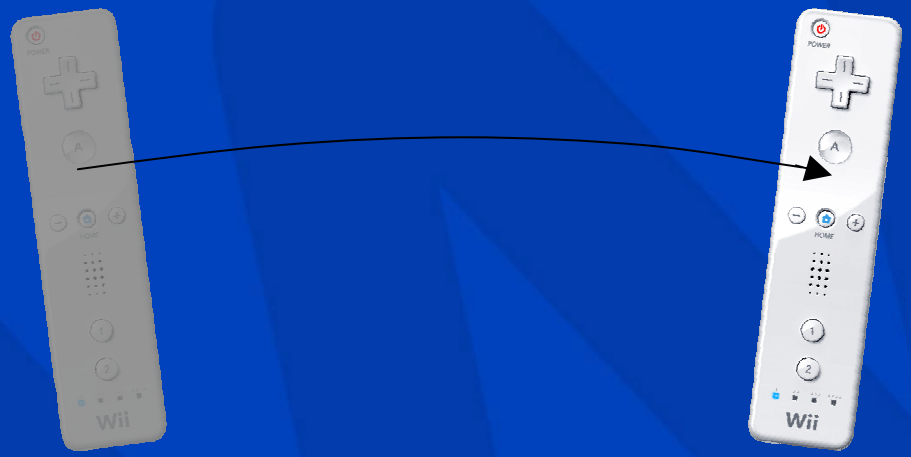


Detecting when Gestures Begin and End

- 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



Accelerometer Gesture Recognition: Simple vs Complex

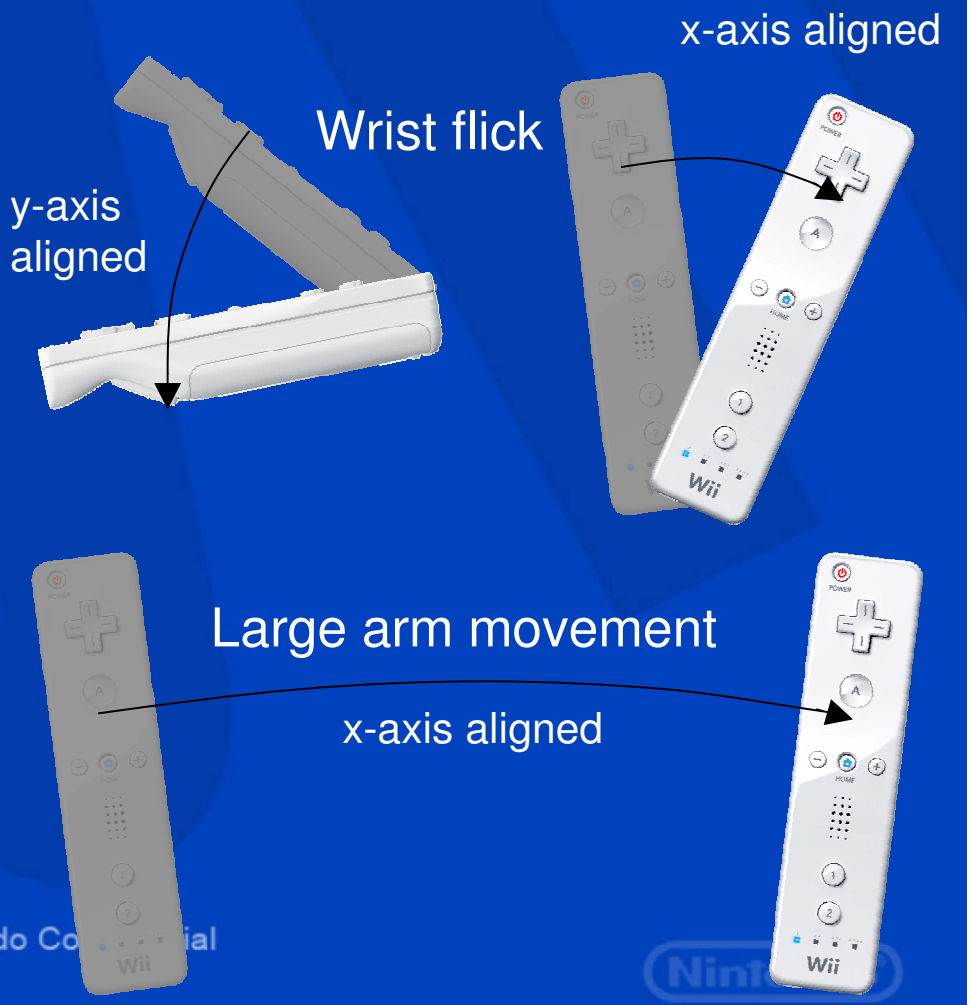


Axis-aligned

Multi-axis

Accelerometer Gesture Recognition: Simple Motion

- Axis-aligned
- Short duration
- Easy to detect



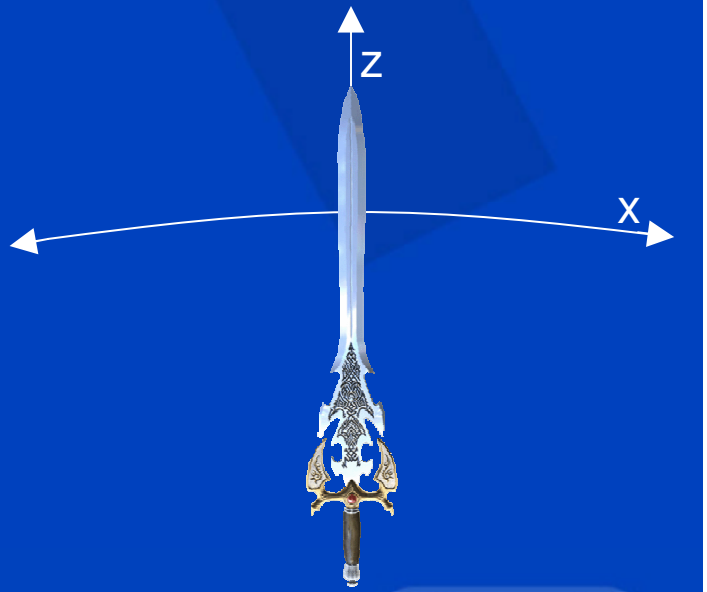
Accelerometer Gesture Recognition: Complex Motion

- Multi-axis
- Longer duration
- Difficult to detect 100%

Multi-axis

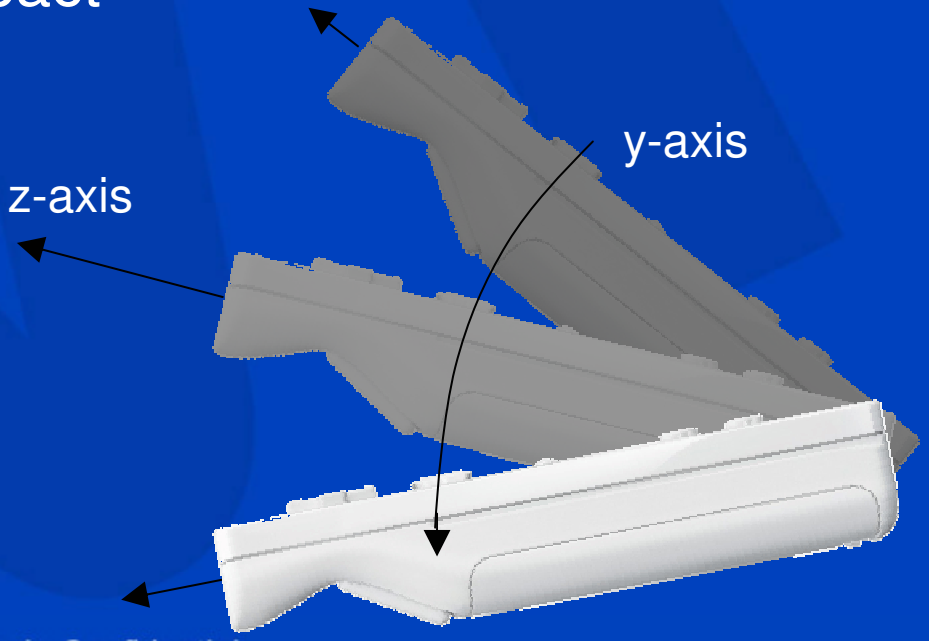
Gesture Recognition:
Simple Motion—Hits, Swipes, and Stabs

- These movements are axis-aligned
  - Easy to detect (using thresholds)
  - Natural player movement, simple to do



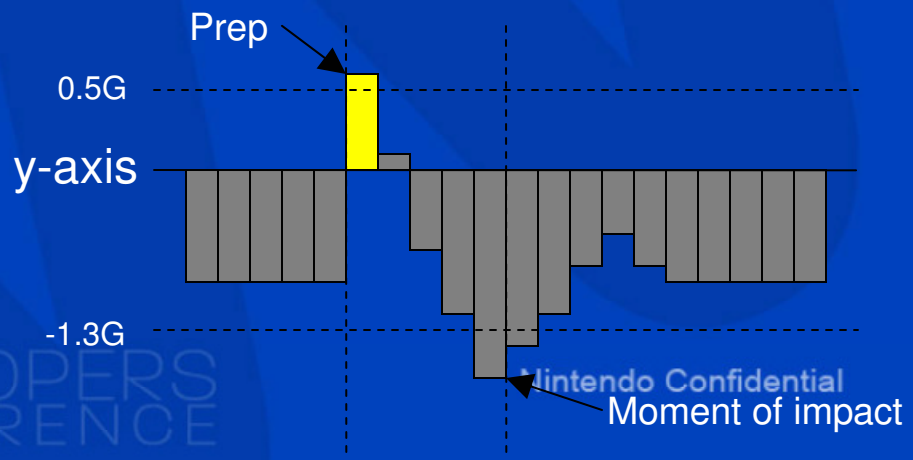
Gesture Recognition:
Simple Motion—Drum Hit Case Study

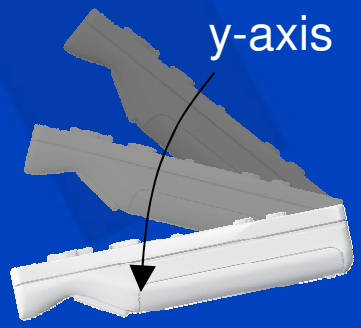
- Two aspects
  - Detect moment of impact
  - Detect strength of impact



Gesture Recognition:
Simple Motion—Drum Hit Case Study

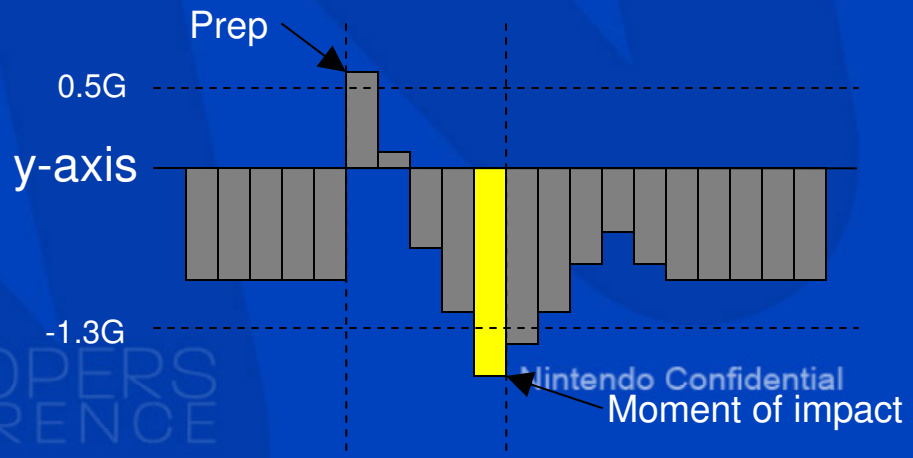
- Detect moment of impact
  - 0.5G "Prep" threshold will figuratively "cock trigger"

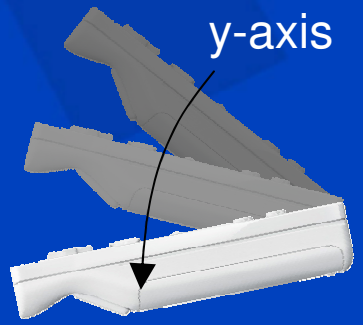




Gesture Recognition:
Simple Motion—Drum Hit Case Study

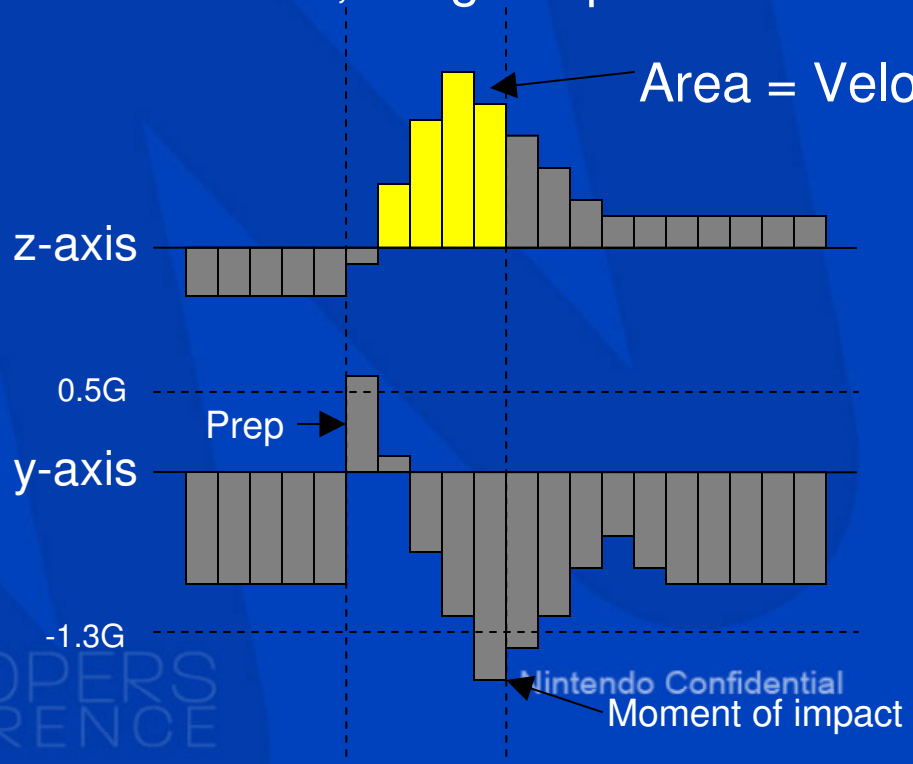
- Detect moment of impact
  - 0.5G "Prep" threshold will figuratively "cock trigger"
  - Once ready, -1.3G threshold represents moment of impact

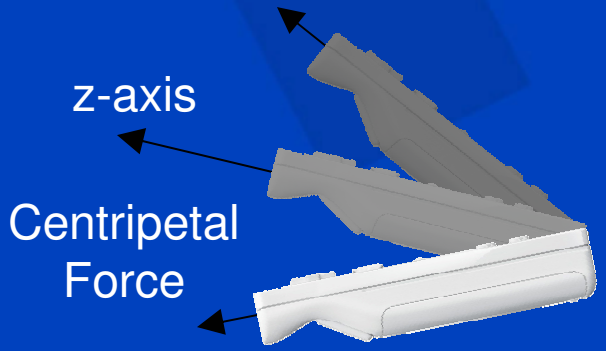




Gesture Recognition:
Simple Motion—Drum Hit Case Study

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

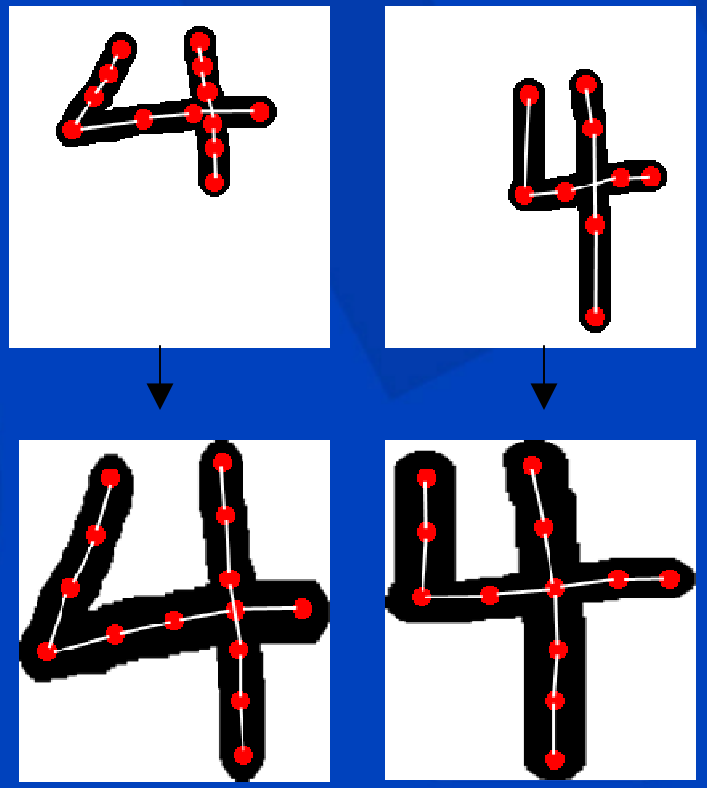




Complex Gesture Recognition: Five Techniques

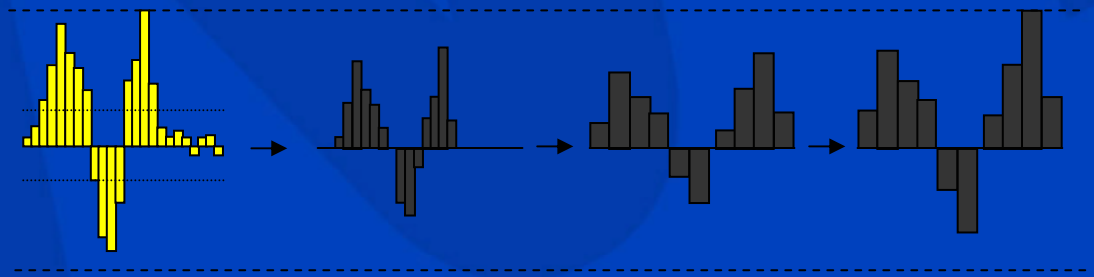
Complex Gesture Recognition: Preprocessing the Signal

- Example from handwriting recognition
  - Normalize size
  - Normalize length/speed



Complex Gesture Recognition: First Step—Preprocessing

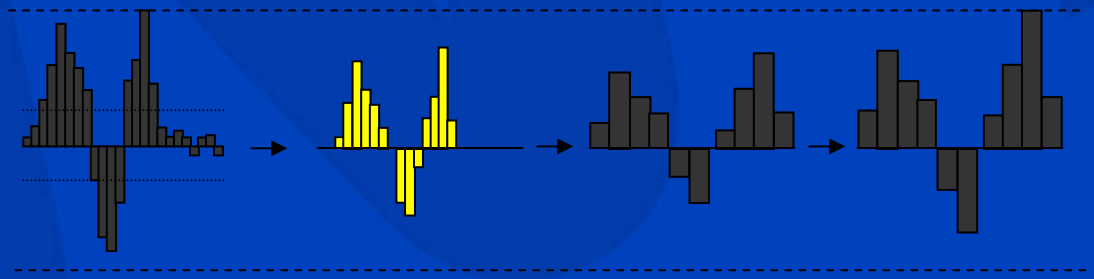
Massage input to look consistent/uniform



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

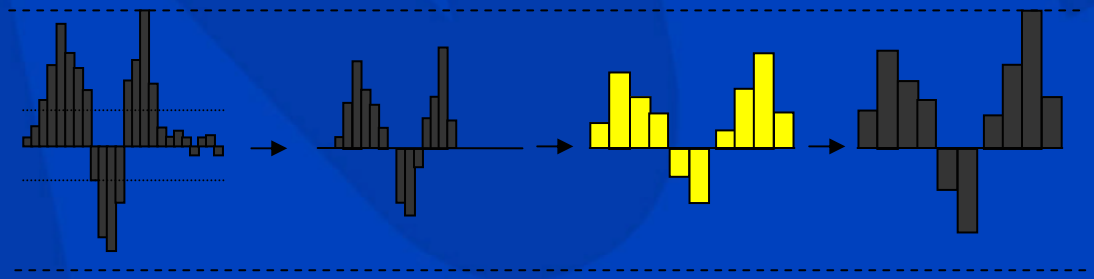


Raw Step 1 Step 2,3 Step 4

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

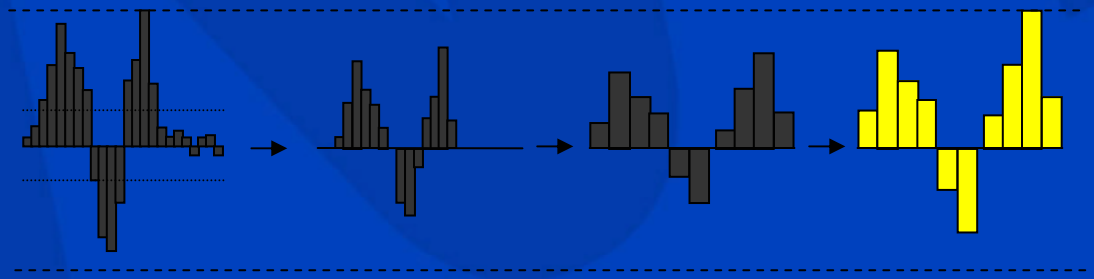


Raw Step 1 Step 2,3 Step 4

Complex Gesture Recognition: First Step—Preprocessing

Massage input to look consistent/uniform

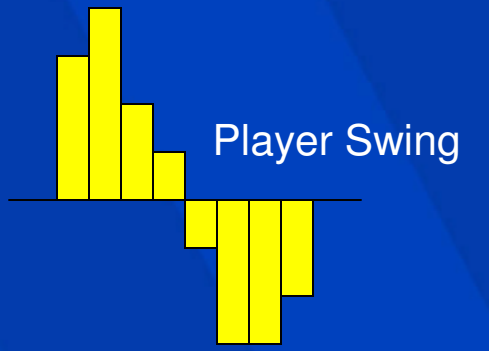
1. (optional) Remove gravity from all axes
   - Gravity problematic
   - Removes small movement noise
2. Remove parts with no acceleration
3. Normalize length
4. Normalize intensity

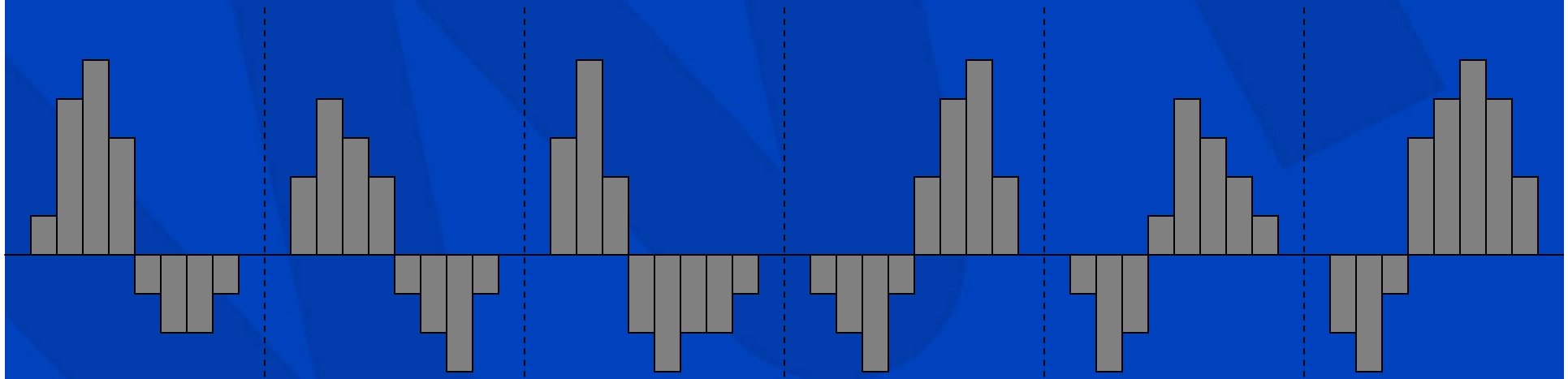


Raw Step 1 Step 2,3 Step 4

Complex Gesture Recognition: Technique 1—Nearest Neighbor

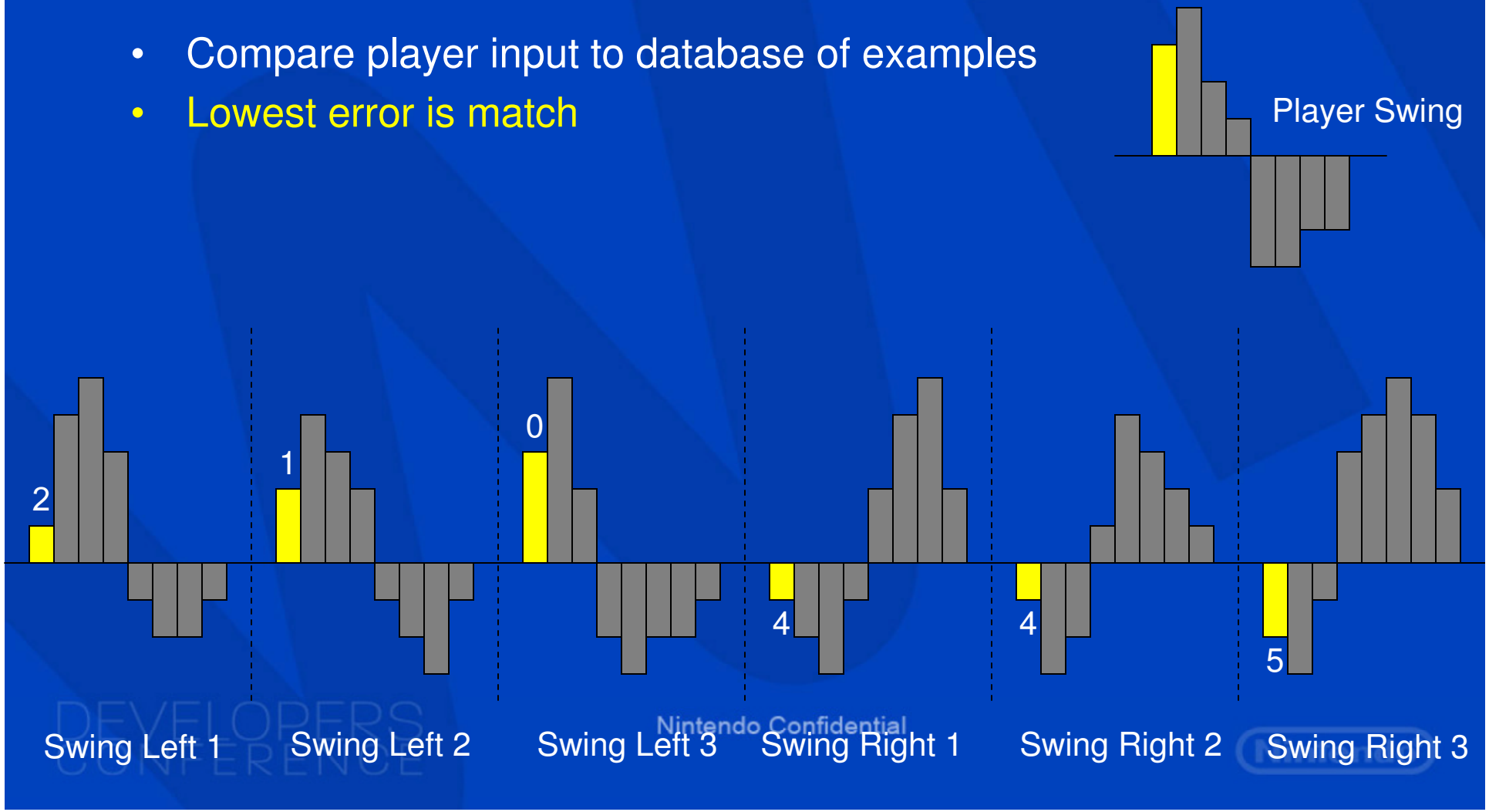
- Compare player input to database of examples





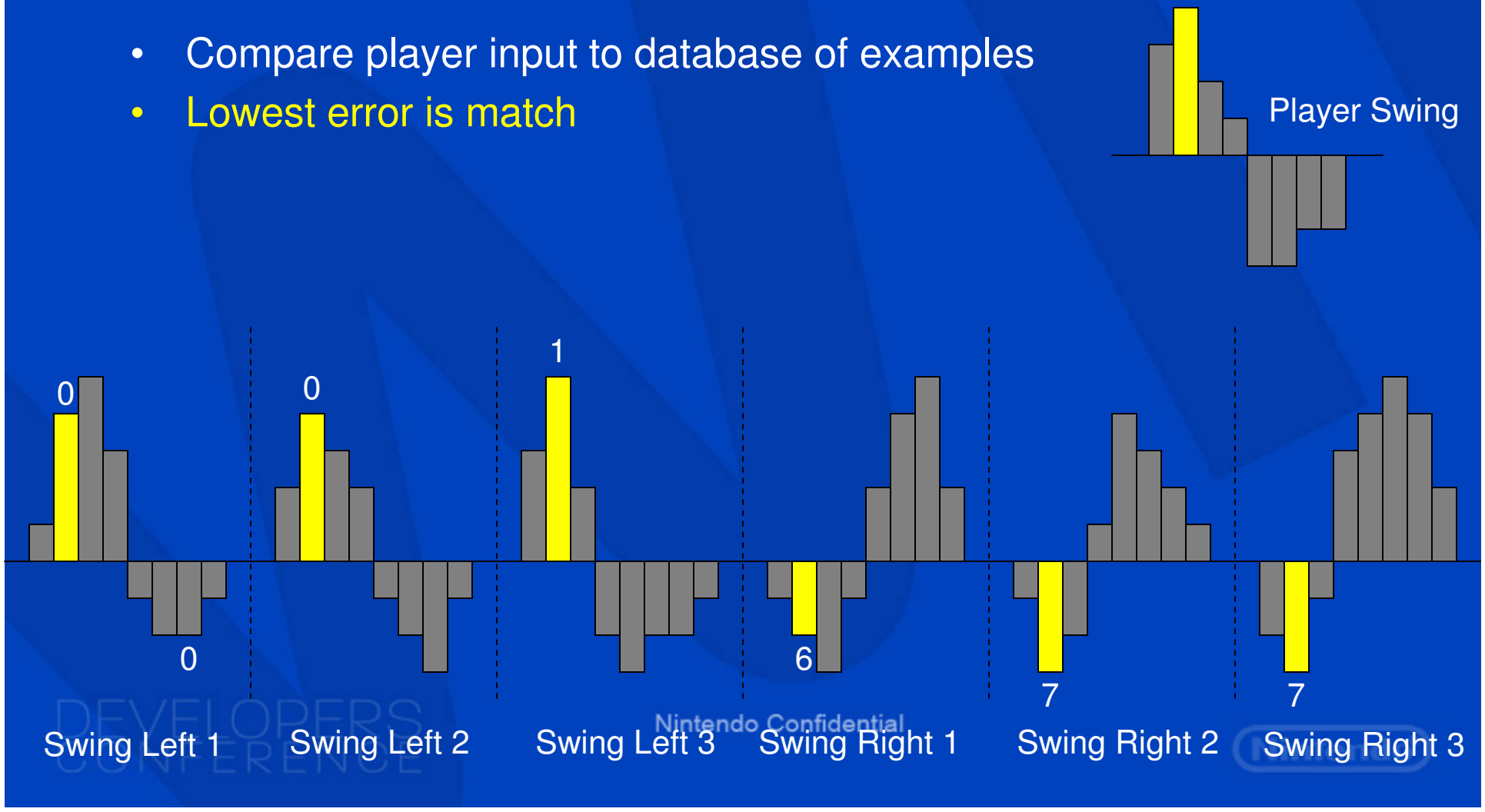
Complex Gesture Recognition: Technique 1—Nearest Neighbor

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



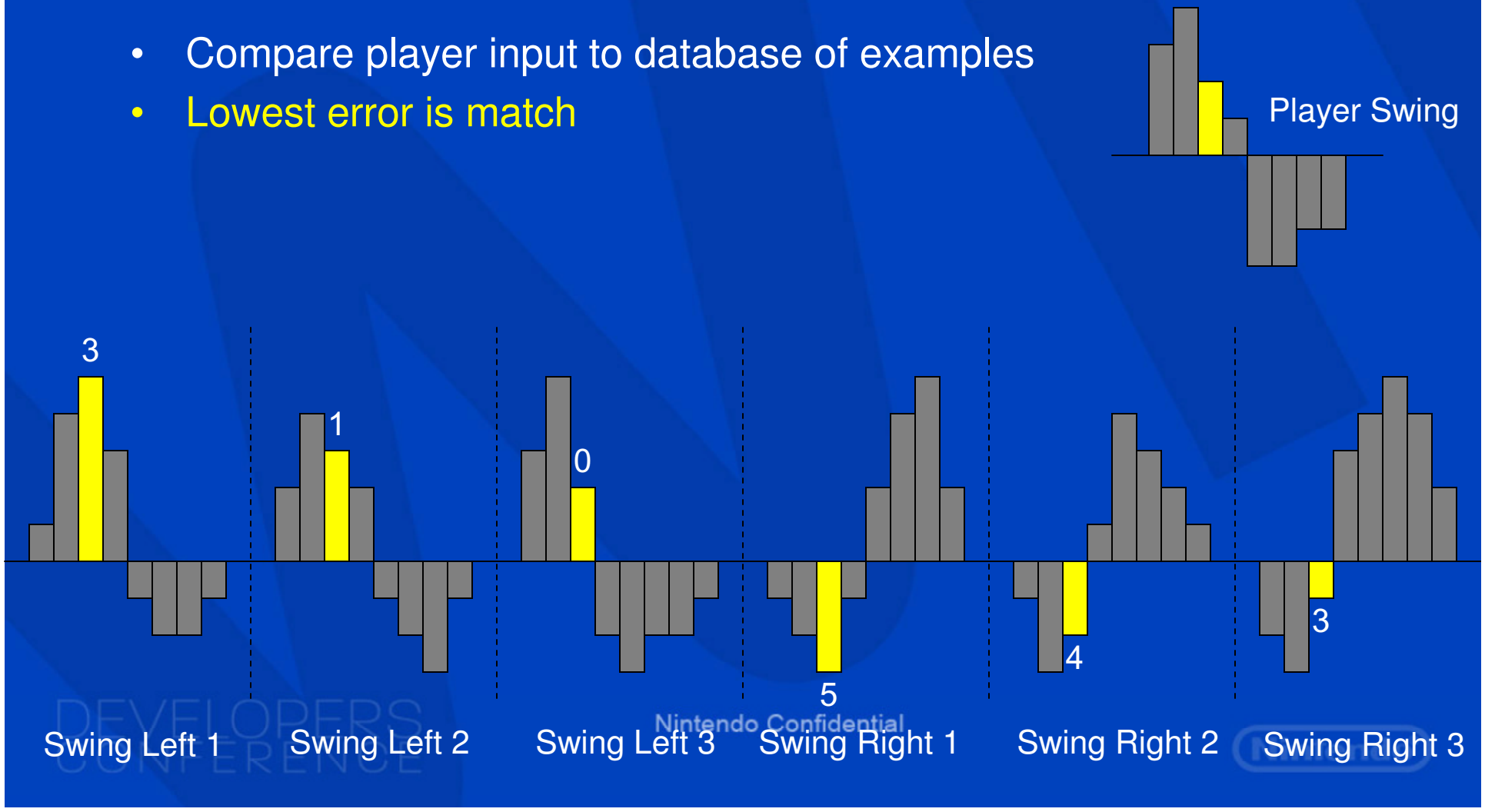
Complex Gesture Recognition: Technique 1—Nearest Neighbor

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



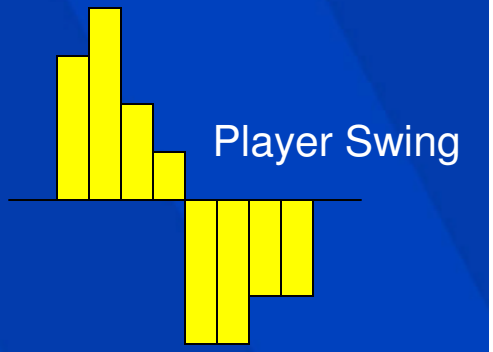
Complex Gesture Recognition: Technique 1—Nearest Neighbor

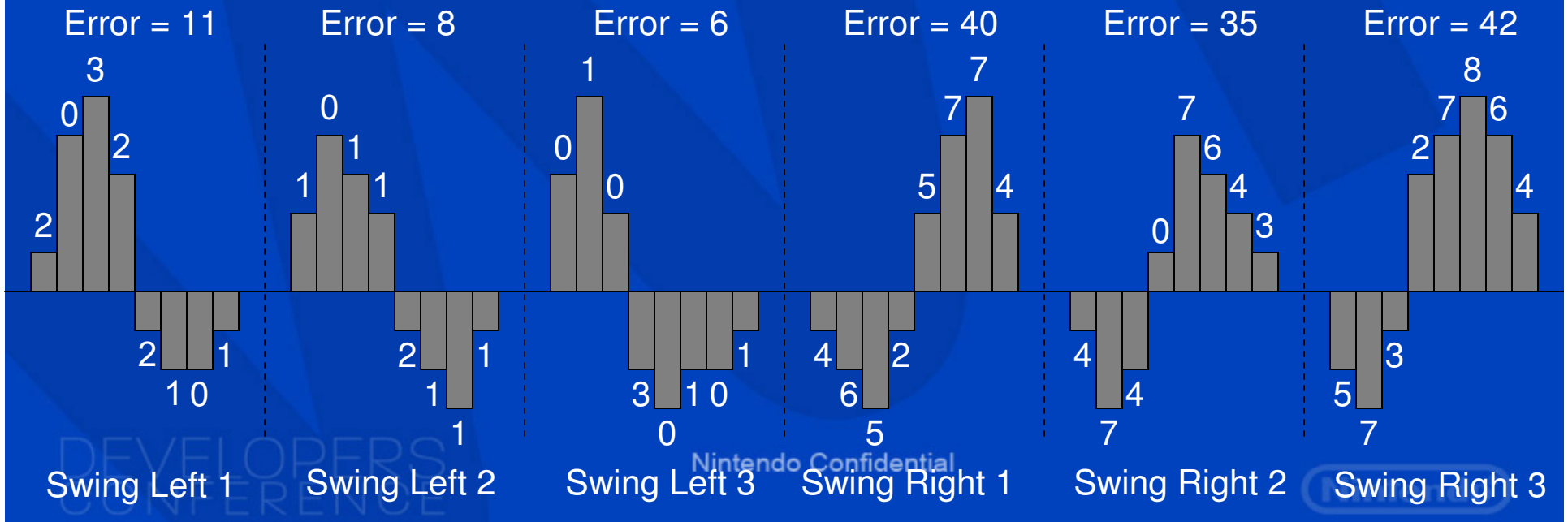
- Compare player input to database of examples
- Lowest error is match



Complex Gesture Recognition: Technique 1—Nearest Neighbor

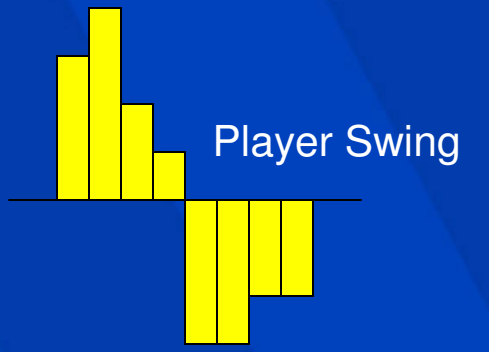
- Compare player input to database of examples
- Lowest error is match

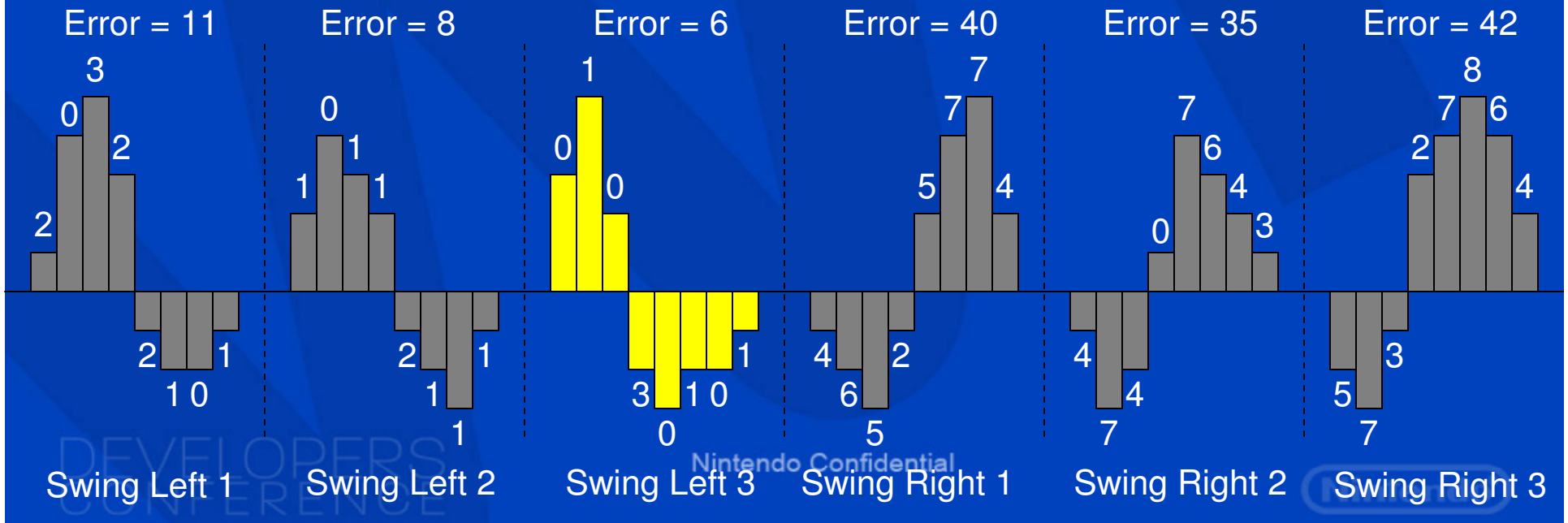




Complex Gesture Recognition: Technique 1—Nearest Neighbor

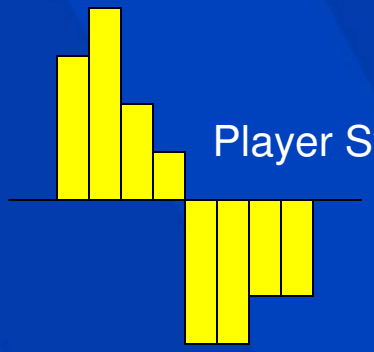
- Compare player input to database of examples
- Lowest error is match

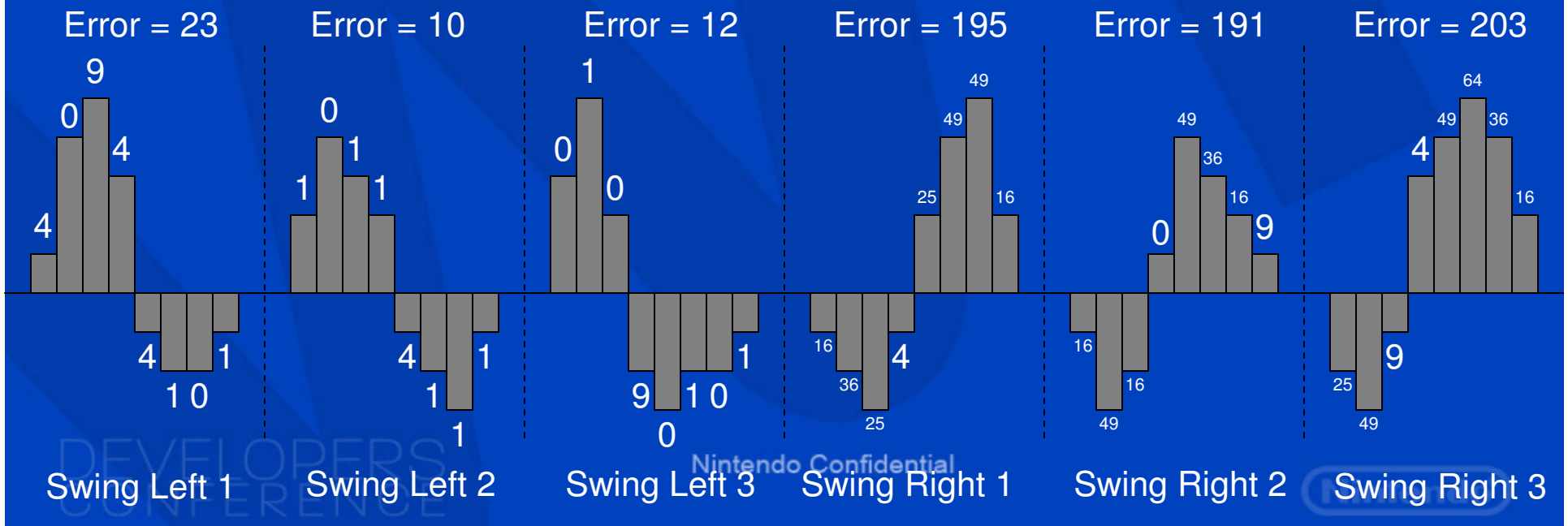




Complex Gesture Recognition: Technique 1—Nearest Neighbor

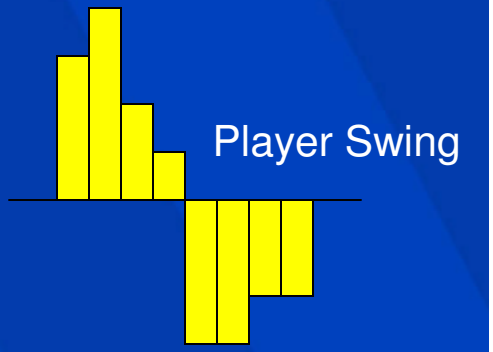
- Compare player input to database of examples
- Lowest error is match (ROOT MEAN SQUARE!)

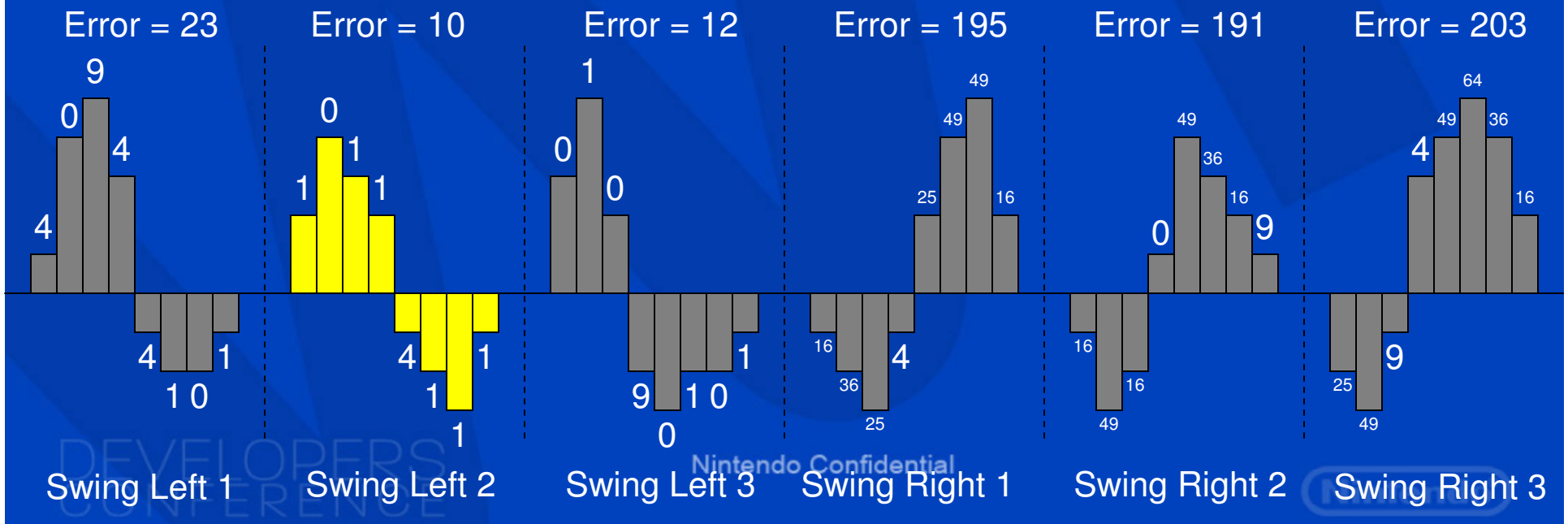




Complex Gesture Recognition: Technique 1—Nearest Neighbor

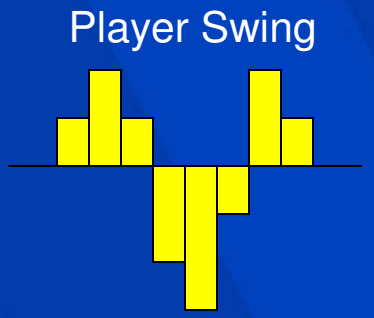
- Compare player input to database of examples
- Lowest error is match (ROOT MEAN SQUARE!)

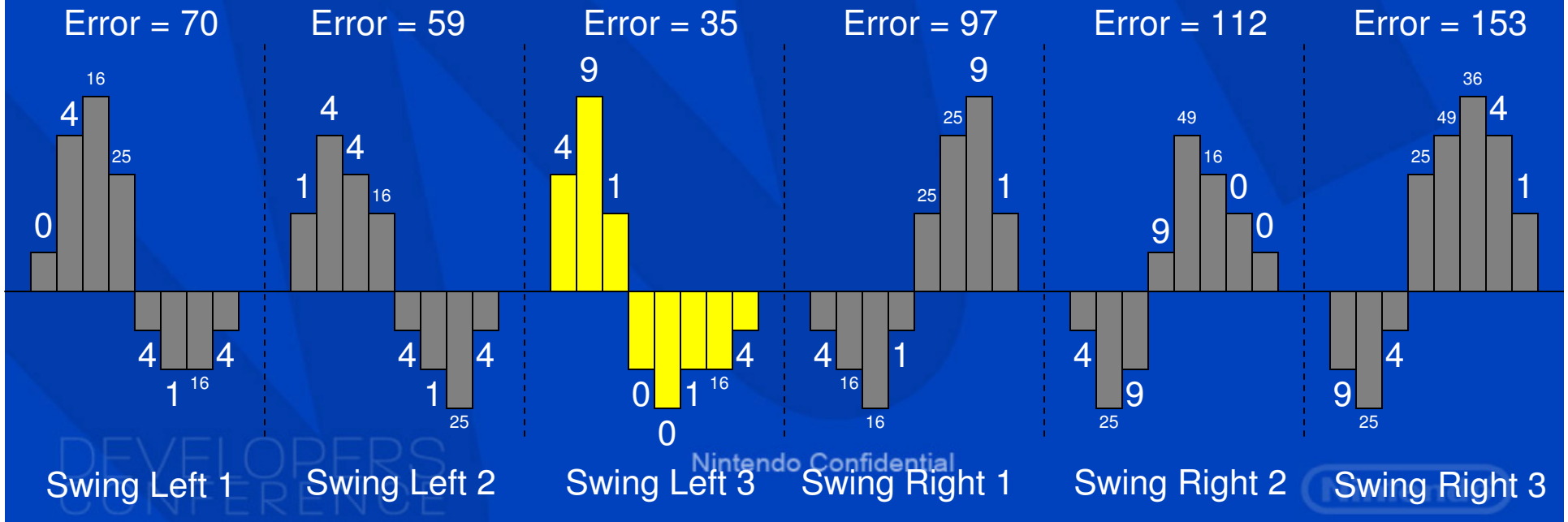




Complex Gesture Recognition: Technique 1—Nearest Neighbor

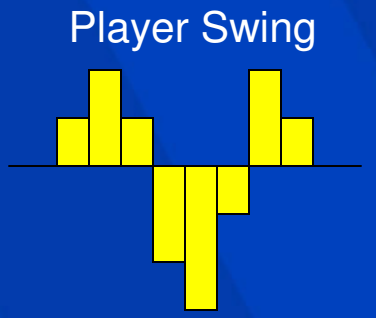
- Compare player input to database of examples
- Lowest error is match

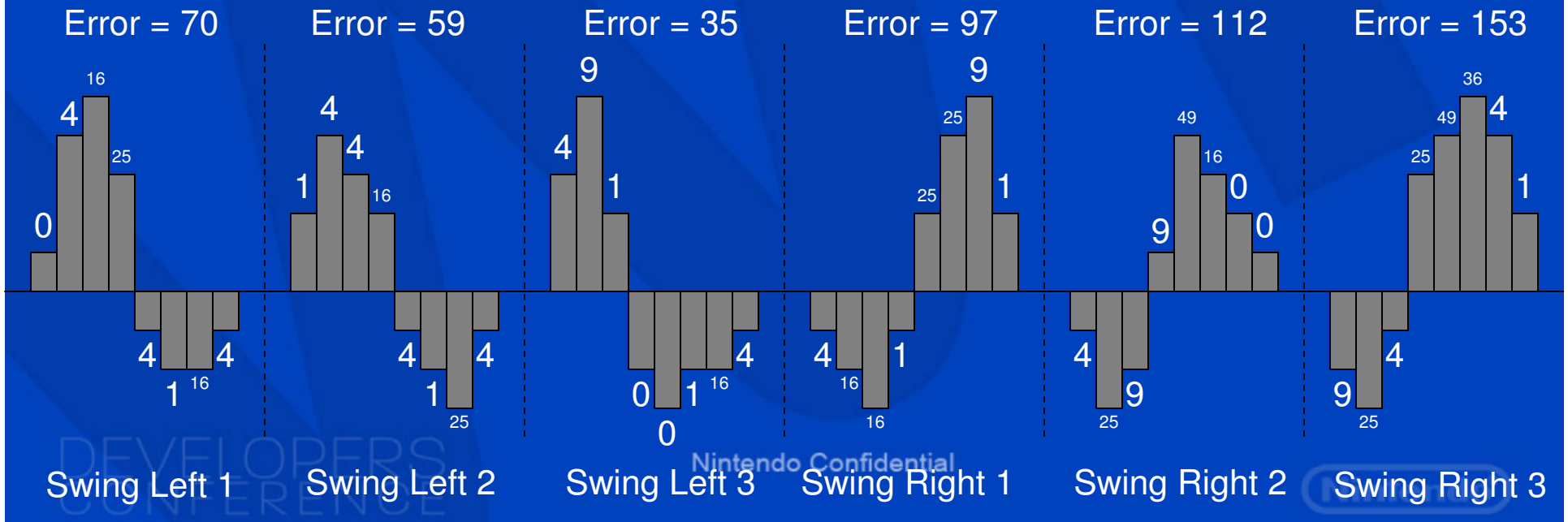




Complex Gesture Recognition: Technique 1—Nearest Neighbor

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



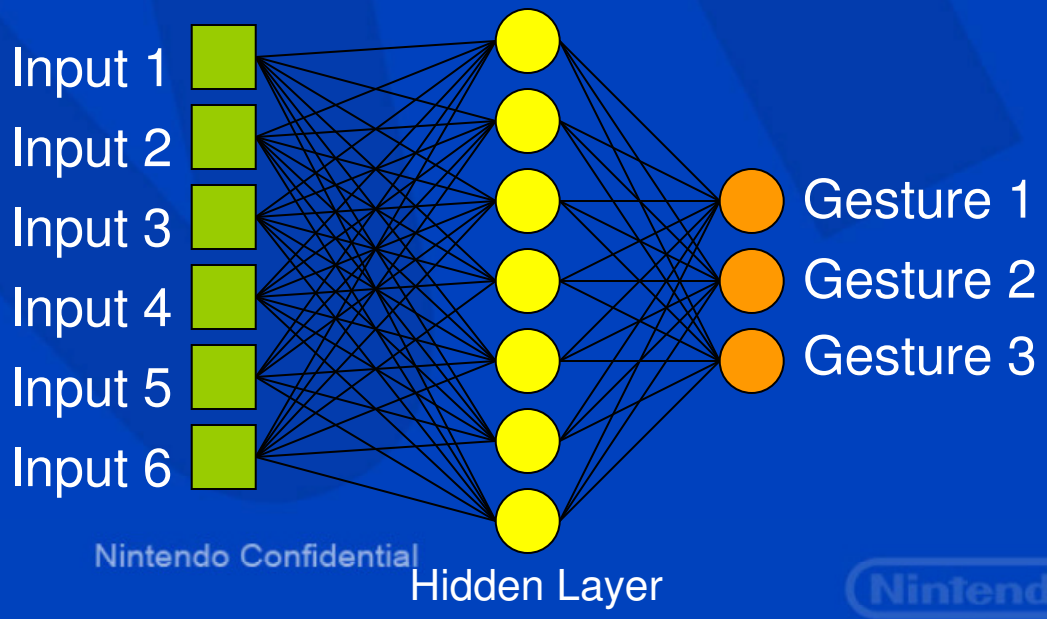


Complex Gesture Recognition: Technique 1—Nearest Neighbor

- General algorithm to match against database
  - Not many examples needed
  - Preprocess data for best matching
- Can constantly monitor input stream
- Player could supply examples

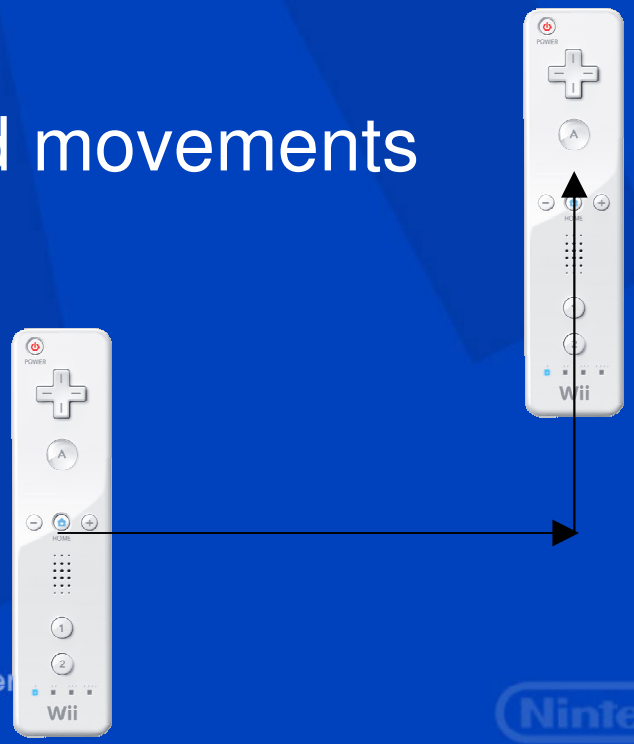
Complex Gesture Recognition: Technique 2—Neural Network

- Black box that tells you the answer
- You train it with 100s or 1000s of examples
  - Network generalizes to examples

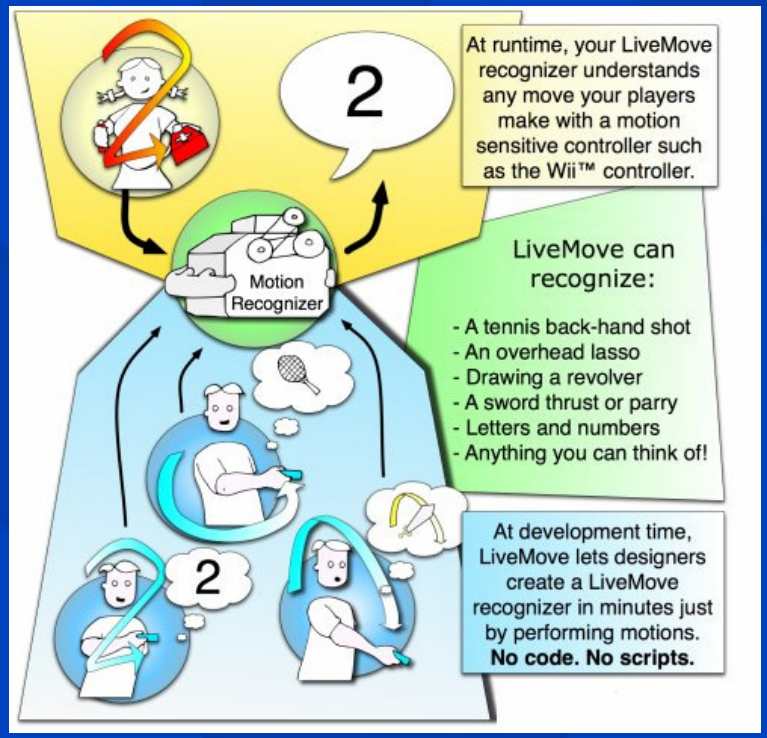


Complex Gesture Recognition: Technique 3—Cheat

- 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



www.ailive.net
support@ailive.net

Complex Gesture Recognition: Technique 5—Use your Brain

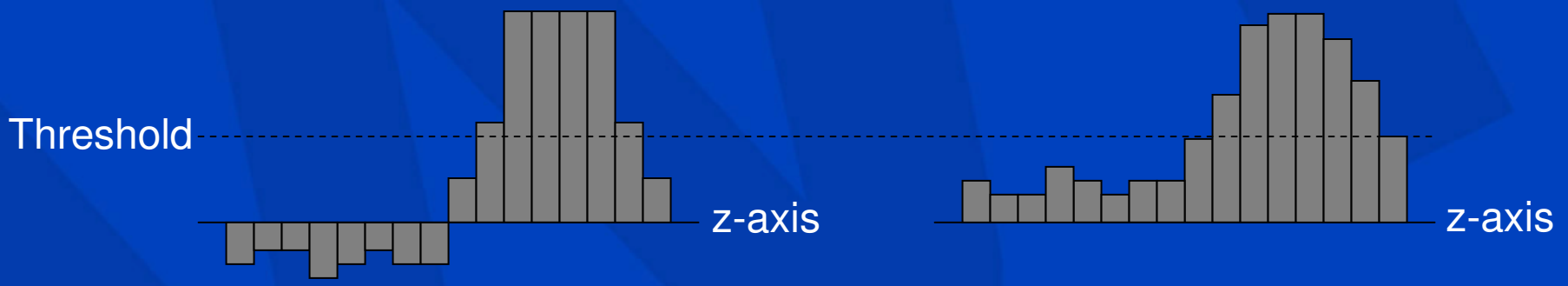
1. Study the move(s) you want to detect
2. Identify its features
   - Is there a single feature that is unique?
   - Is it consistent no matter who does the gesture?
3. Write custom detection code for the single gesture
   - Various threshold tests in sequence
   - Threshold triggering relative to other 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

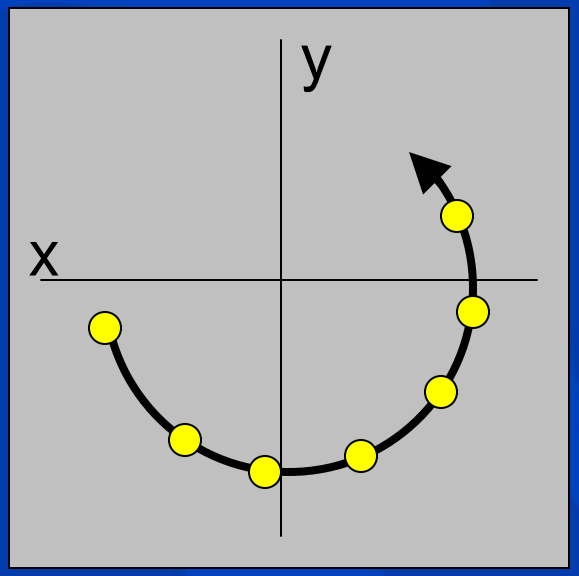
- Recognize any swing
- Recognize left or right swing
- Recognize topspin, backspin, no spin
- Recognize underhand or overhand
- Recognize hard or soft hit

Complex Gesture Recognition:
Recognize Swing

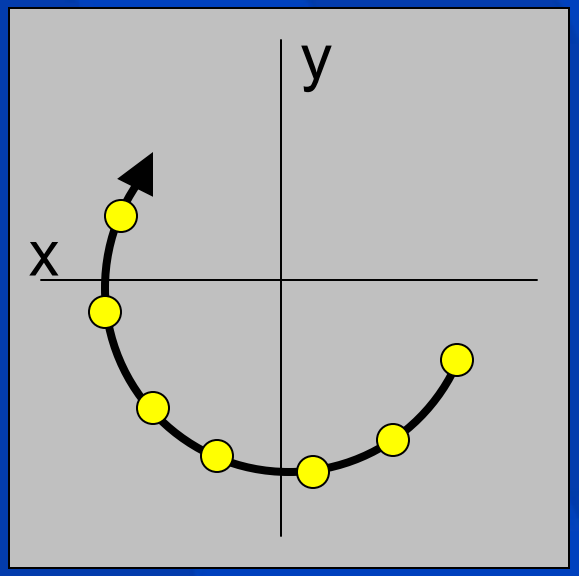
- Threshold on z-axis
  - Something like 1.2G to 1.5G



Complex Gesture Recognition:
Left or Right Swing



Left Swing
(counterclockwise)

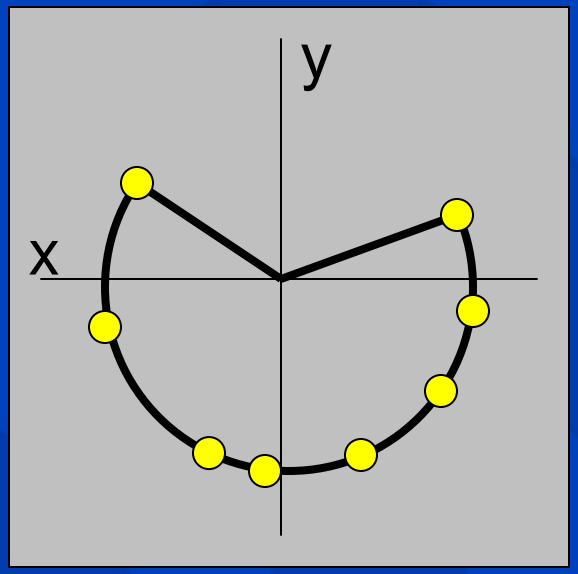


Right Swing
(clockwise)

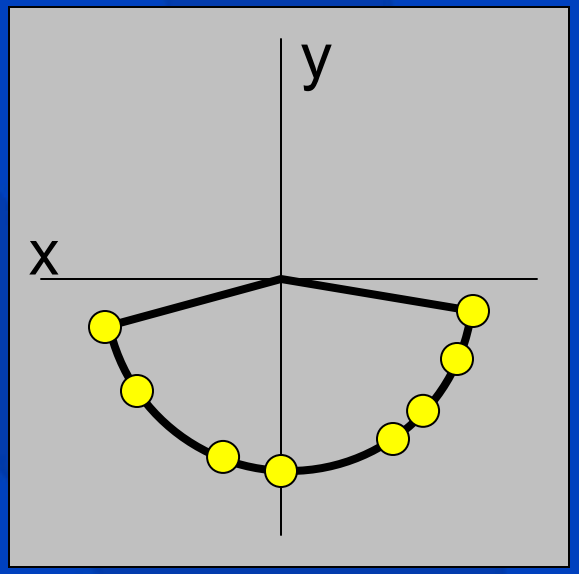
Complex Gesture Recognition:
Left or Right Swing

- Orientation of controller doesn't matter!
- Increase recognition:
  - Predict correct swing
  - Make incorrect swings require larger threshold
    - Avoids mistaking "prep" as swing

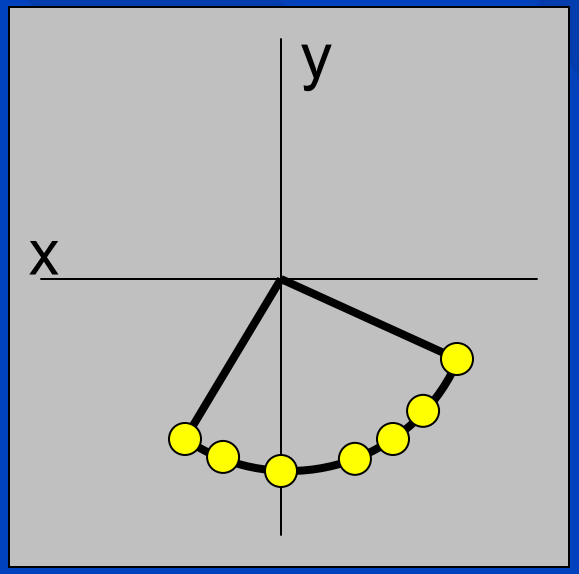
Complex Gesture Recognition:
Topspin, No Spin, or Backspin



Topspin



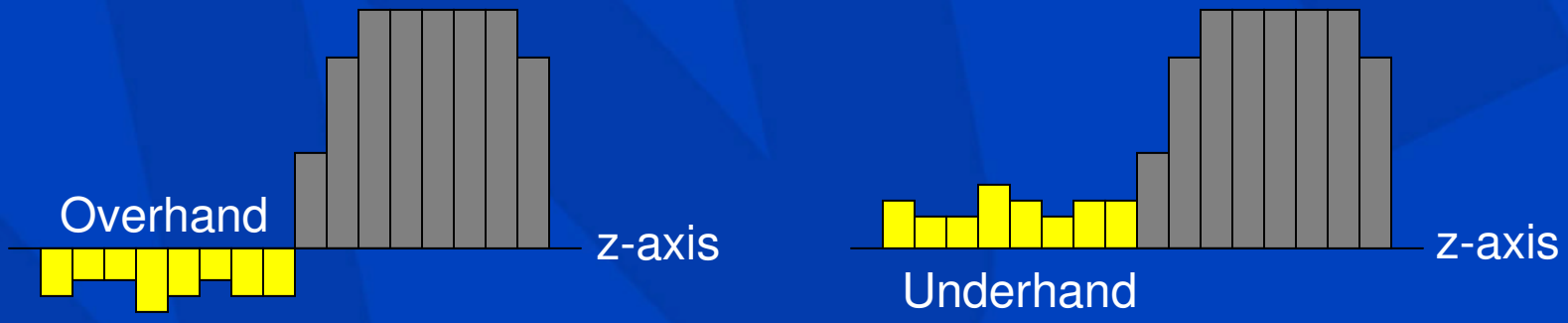
No Spin



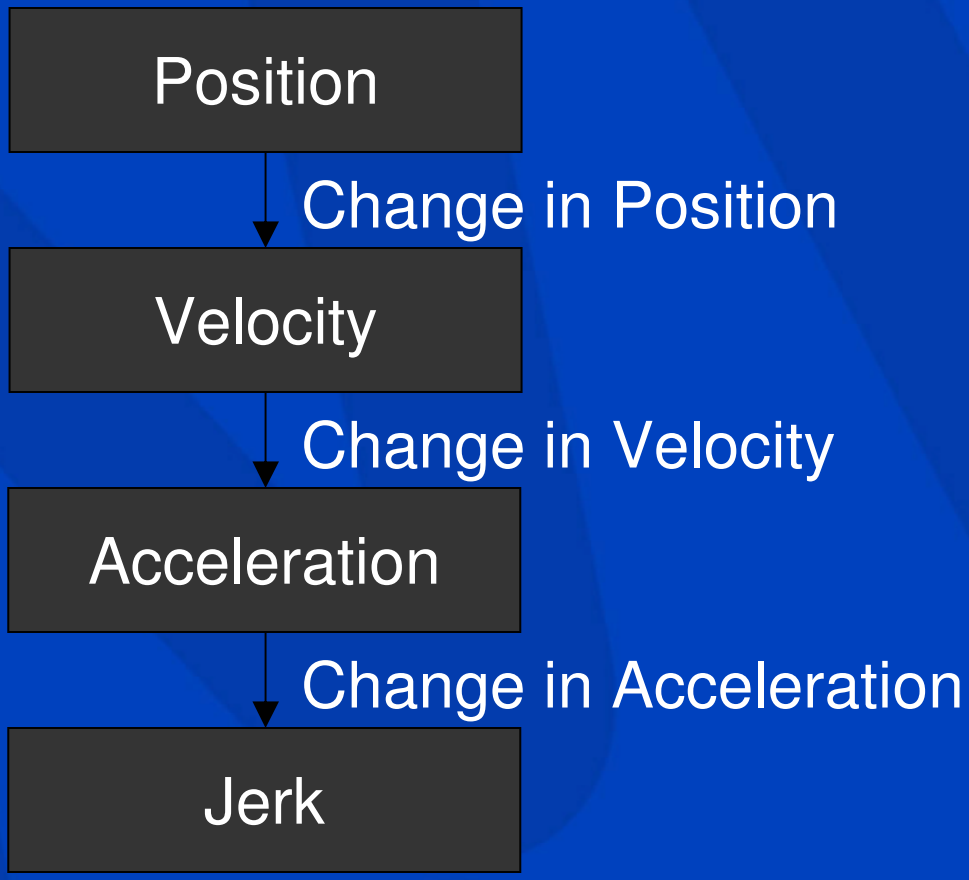
Backspin

Complex Gesture Recognition:
Underhand or Overhand

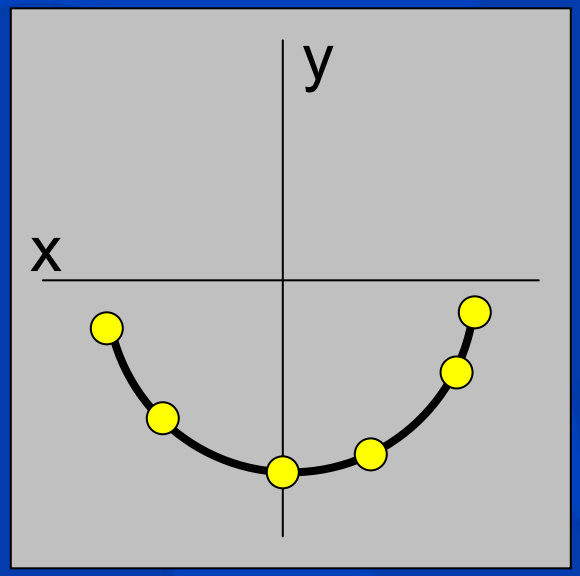
- Look at z-axis before swing
  - Negative = Overhand
  - Positive = Underhand



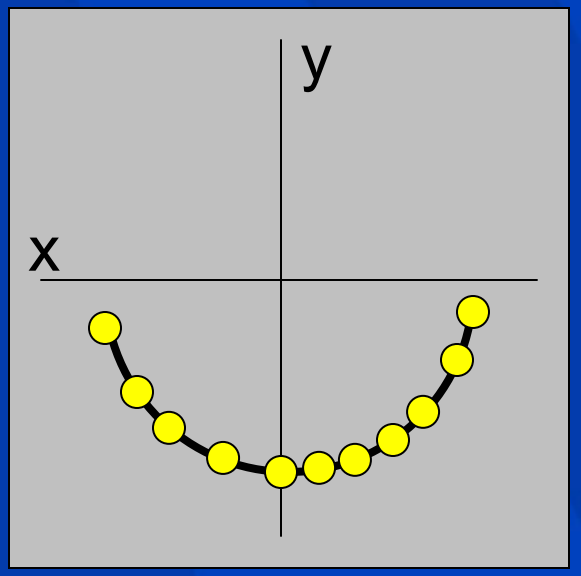
Complex Gesture Recognition:
Hard Hit or Soft Hit



Complex Gesture Recognition:
Hard Hit or Soft Hit



Hard Hit
(high jerk)



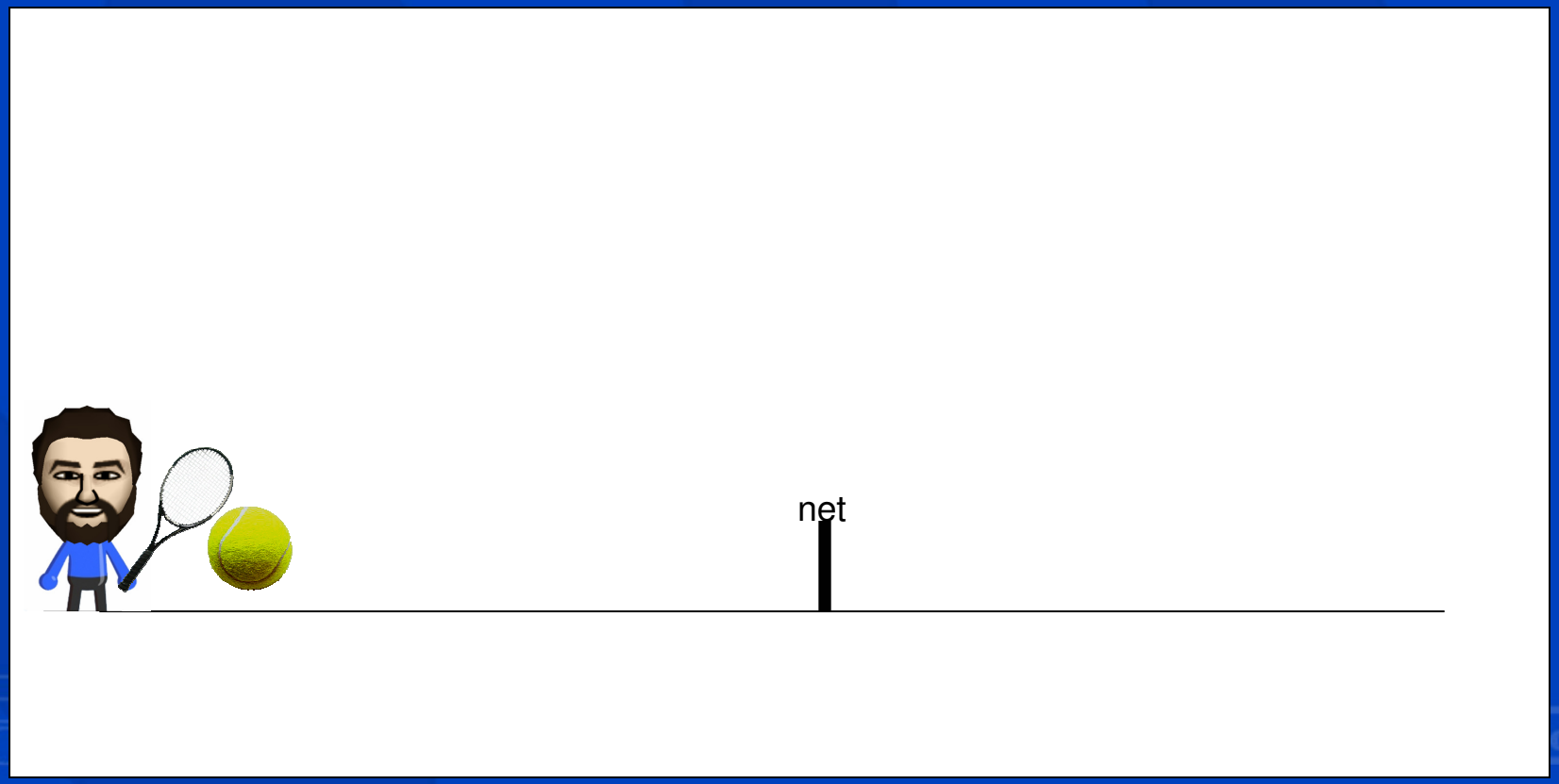
Soft Hit
(low jerk)

Complex Gesture Recognition:
Hard Hit or Soft Hit

- Jerk is loosely correlated with swing speed
  - Slow swing will generally have less jerk
  - Fast swing will generally have more jerk
- However, using this method
  - Quick wrist snap results in highest jerk
  - Hard fast arm swing doesn't result in high jerk
- If you want the hardest hit in Wii Sports Tennis
  - Snap your wrist quickly to create a large acceleration followed by a large deceleration (don't swing your arm hard)
  - You don't need to swing your arm hard

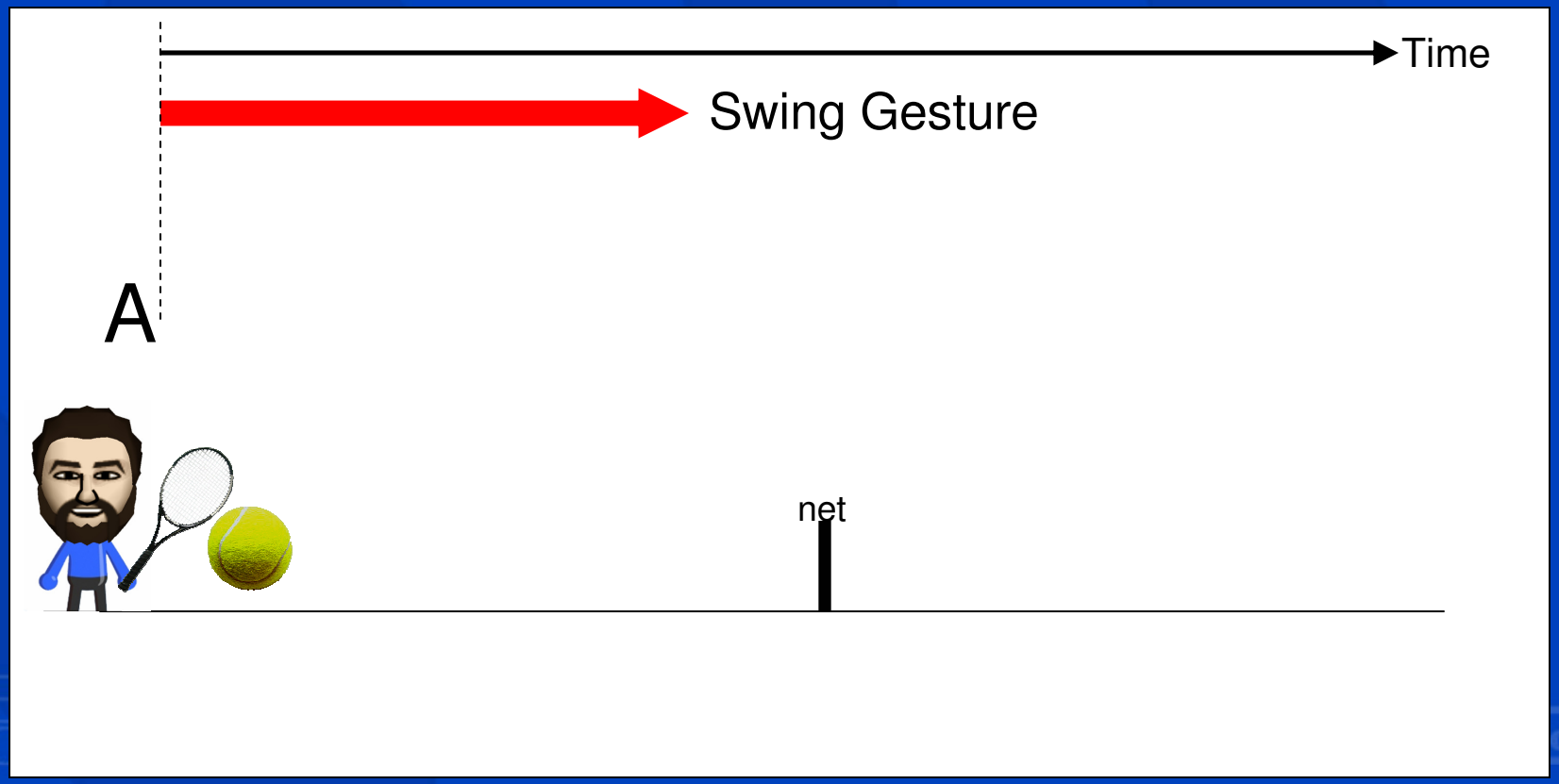
Complex Gesture Recognition:
Wii Sports Tennis Timeline

- Sequence of events during a swing and hit



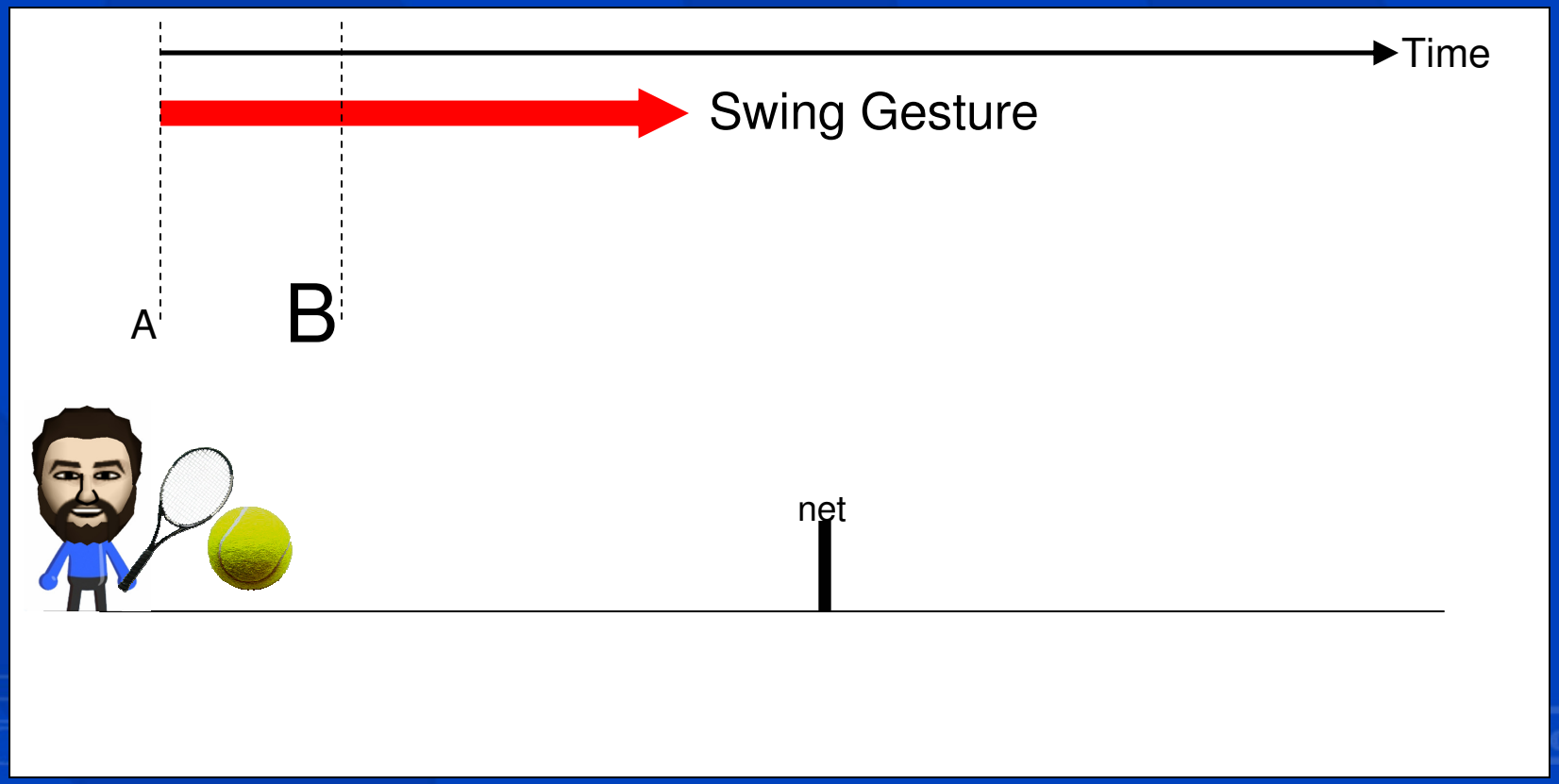
Complex Gesture Recognition:
Wii Sports Tennis Timeline

- Time A: Swing started by player



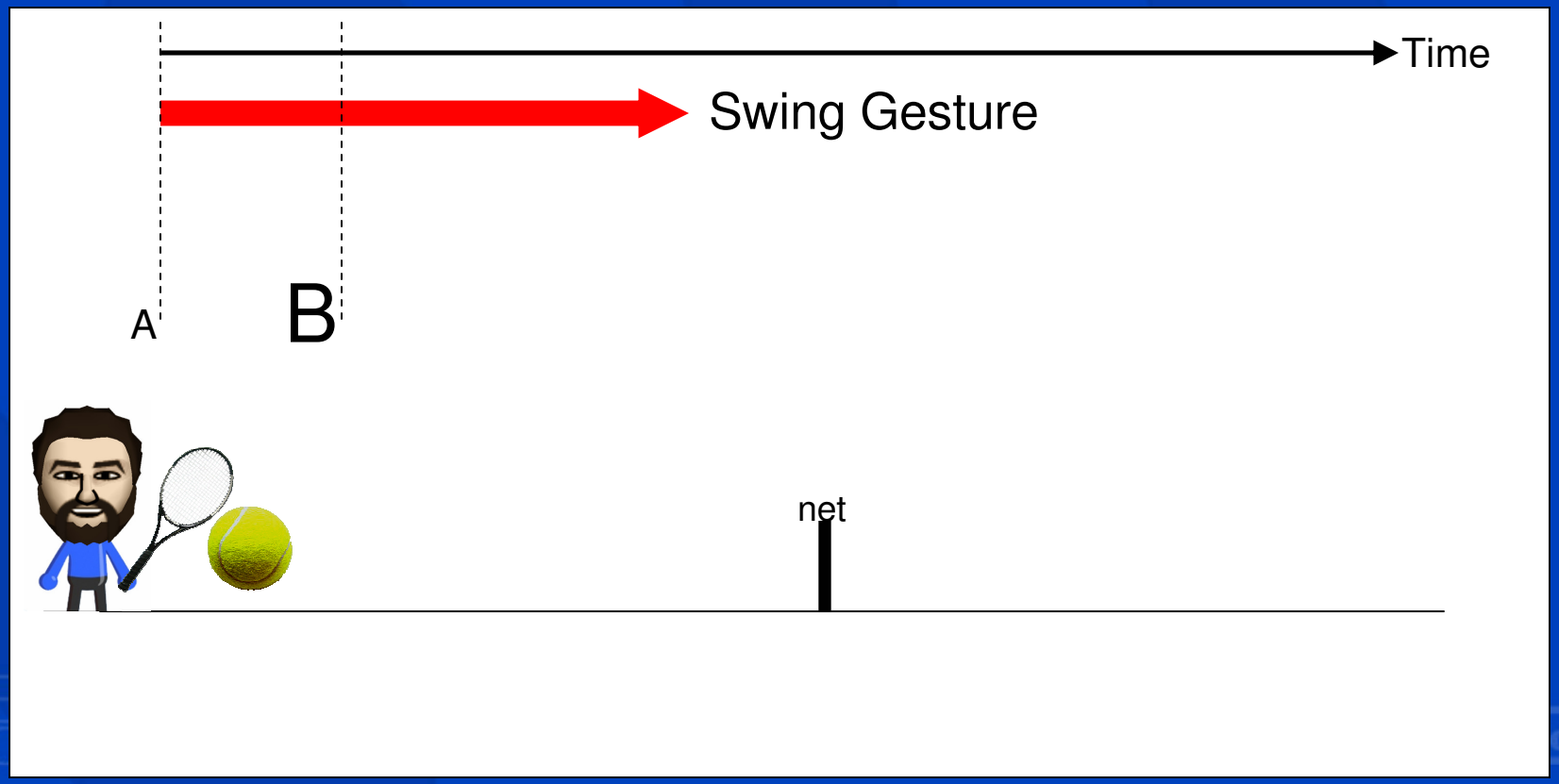
Complex Gesture Recognition:
Wii Sports Tennis Timeline

- Time B: Detect left or right swing



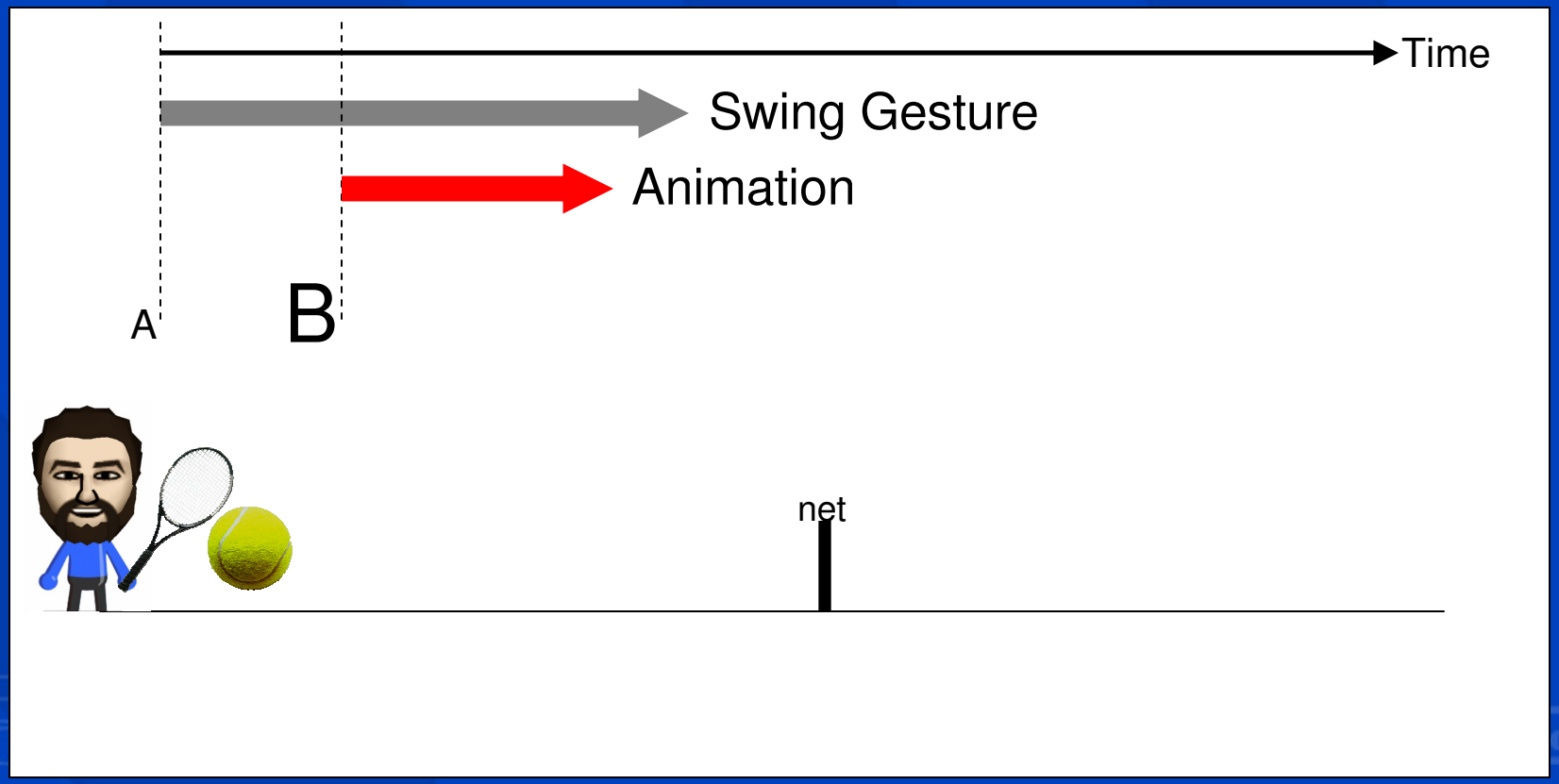
Complex Gesture Recognition:
Wii Sports Tennis Timeline

- Time B: Detect underhand or overhand



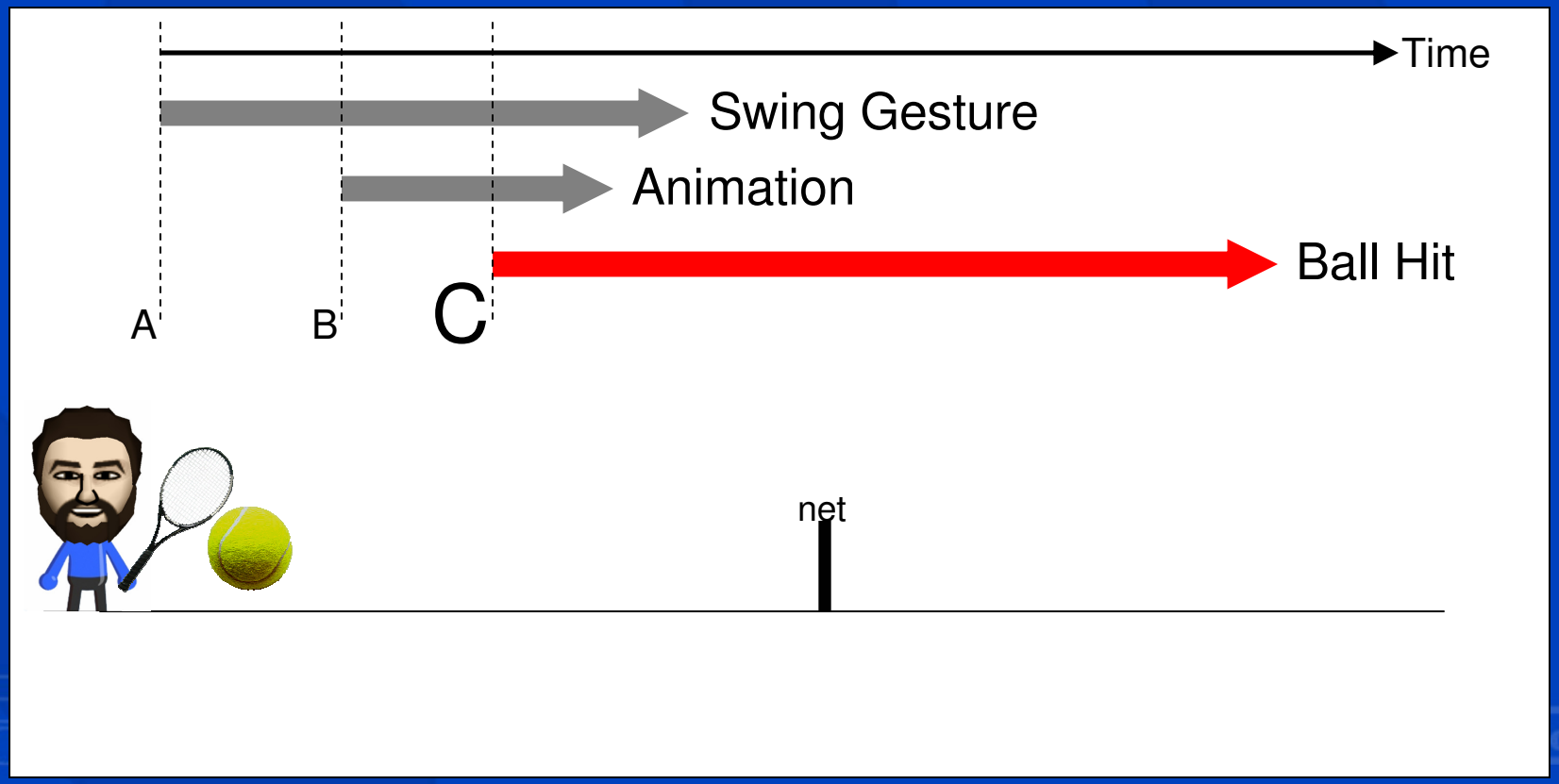
Complex Gesture Recognition:
Wii Sports Tennis Timeline

- Time B: Start animation (left/right, over/under)



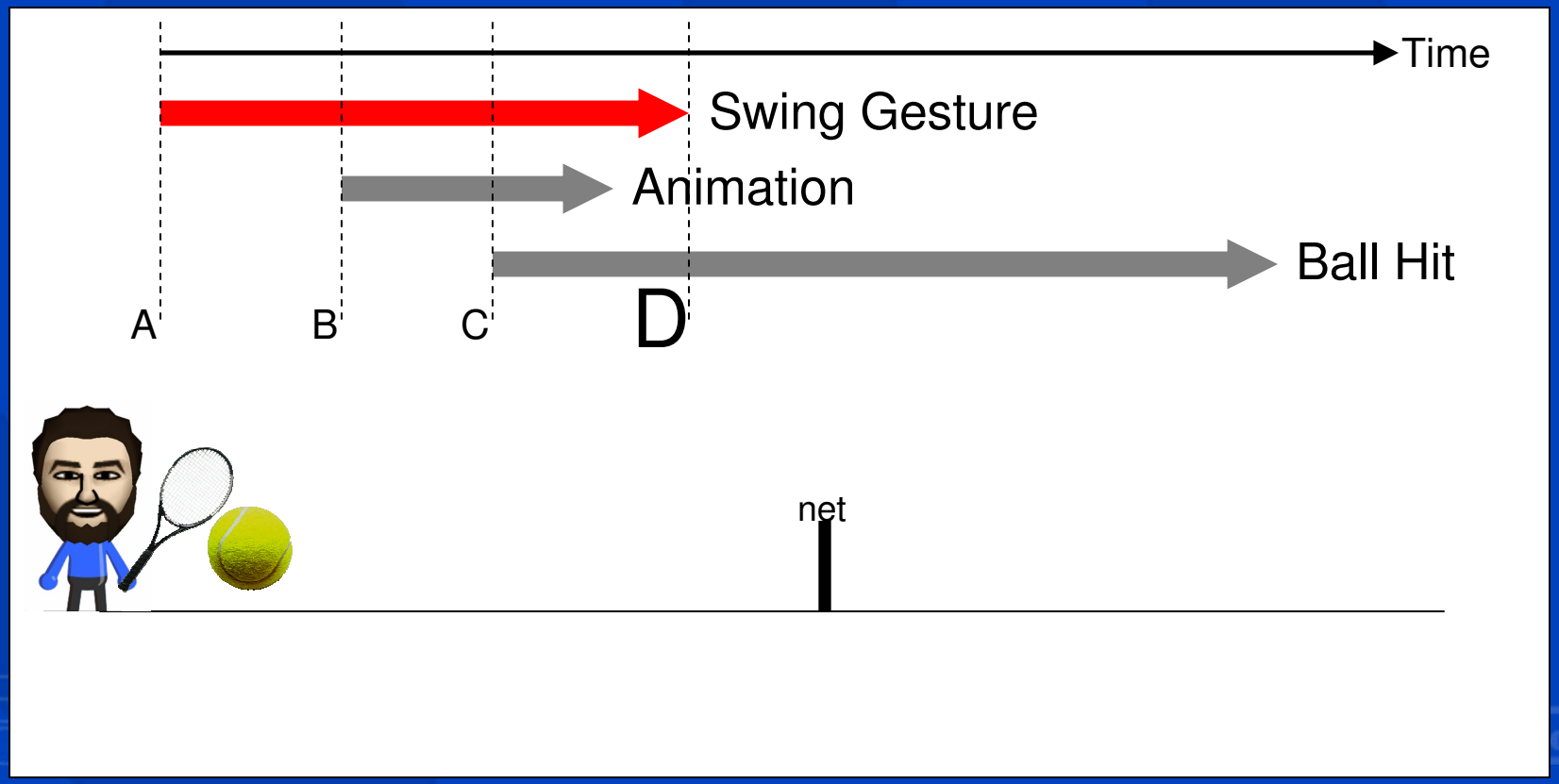
Complex Gesture Recognition:
Wii Sports Tennis Timeline

- Time C: Racket collides with ball



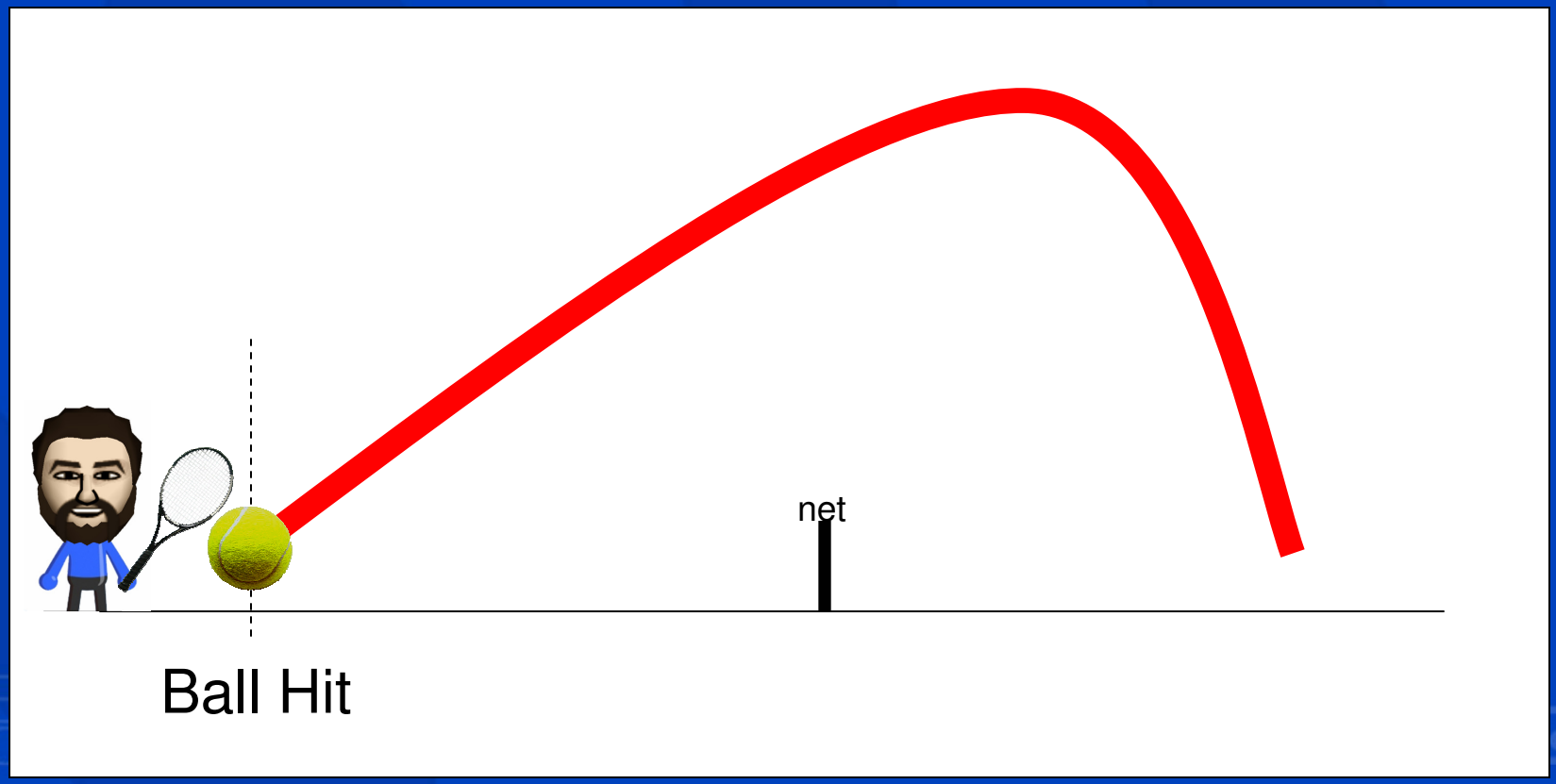
Complex Gesture Recognition:
Wii Sports Tennis Timeline

- Time D: Velocity and spin recognized



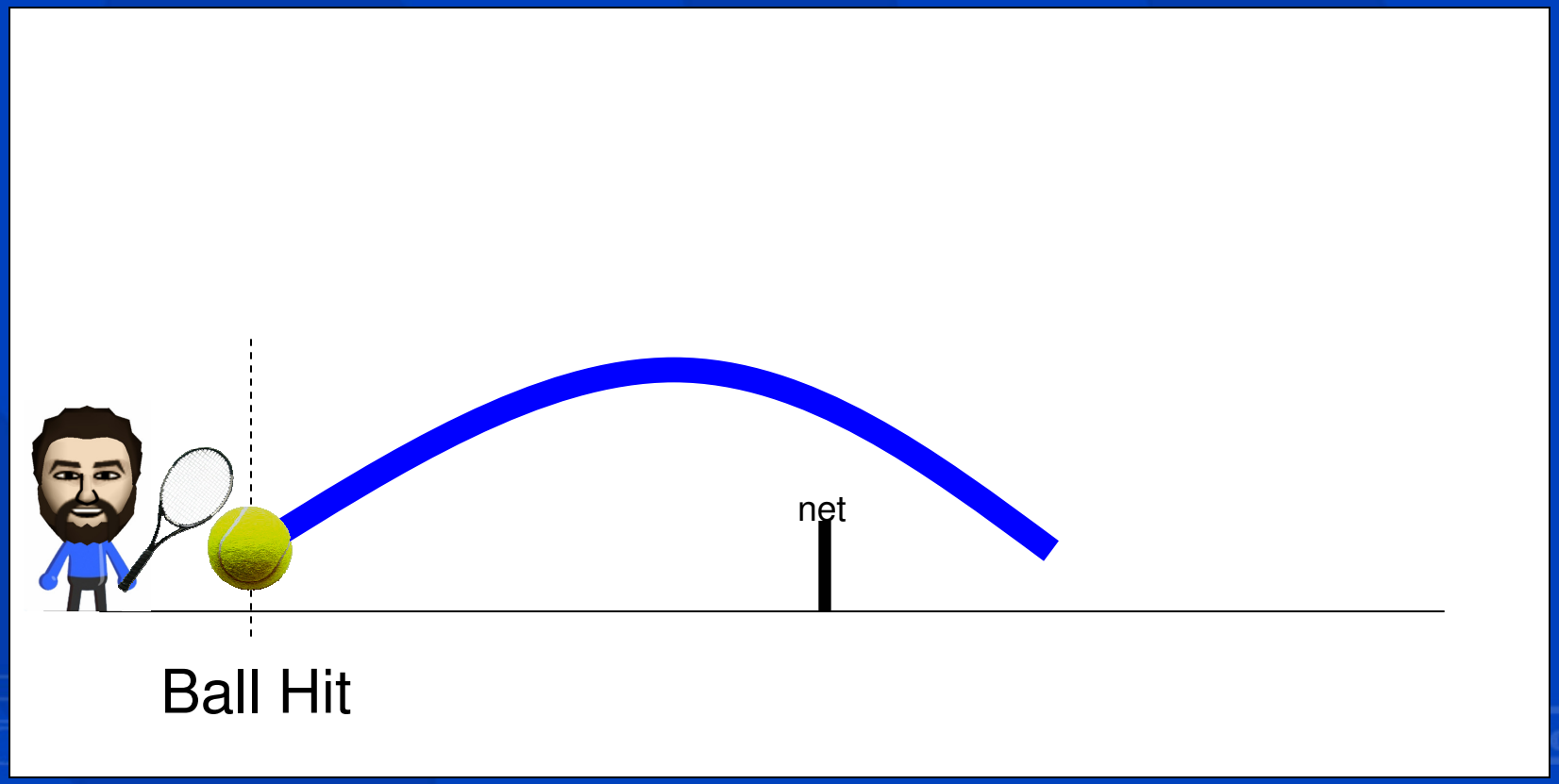
Complex Gesture Recognition:
Wii Sports Tennis Timeline

- High velocity and backspin



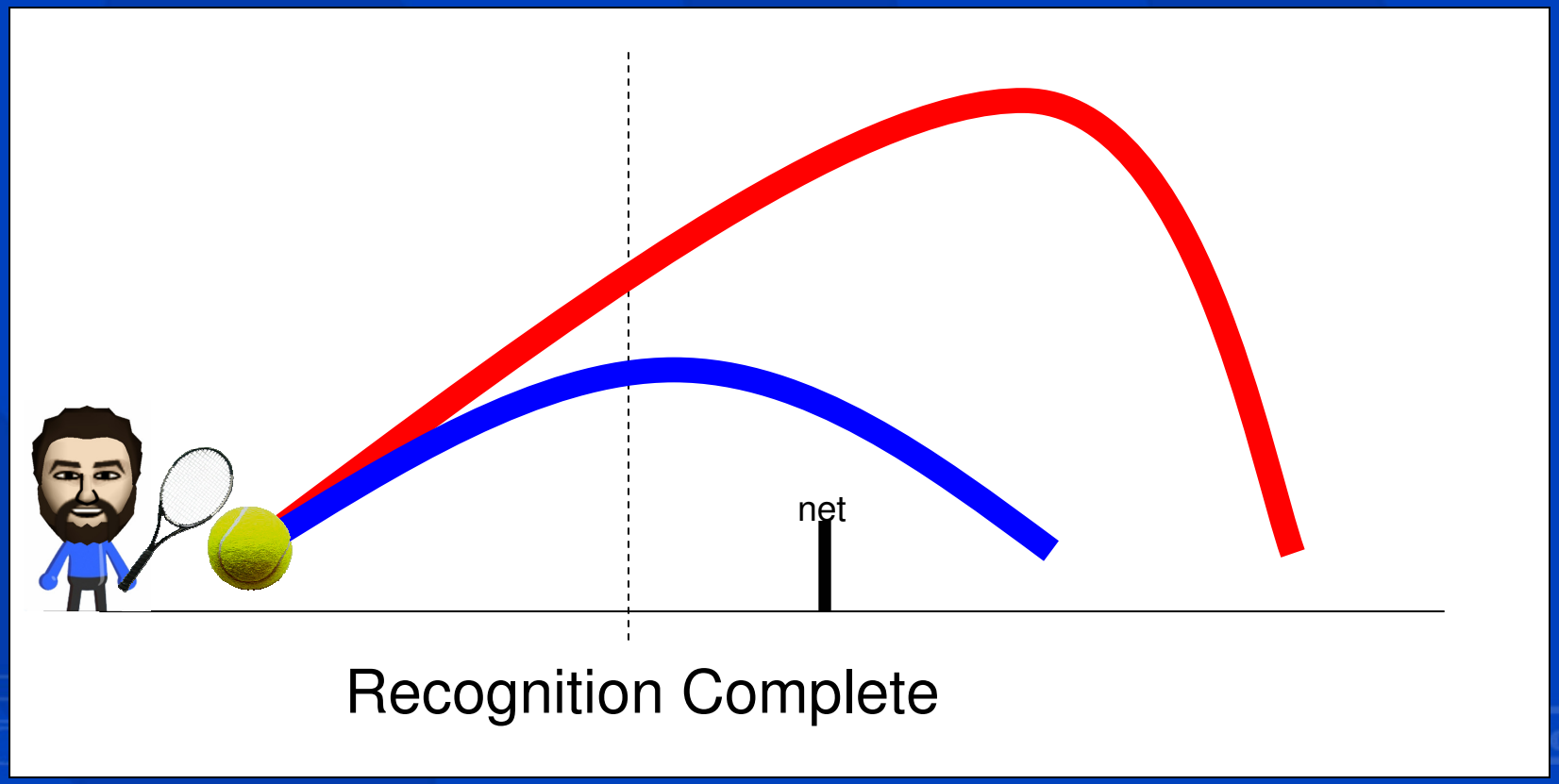
Complex Gesture Recognition:
Wii Sports Tennis Timeline

- Average speed with no spin



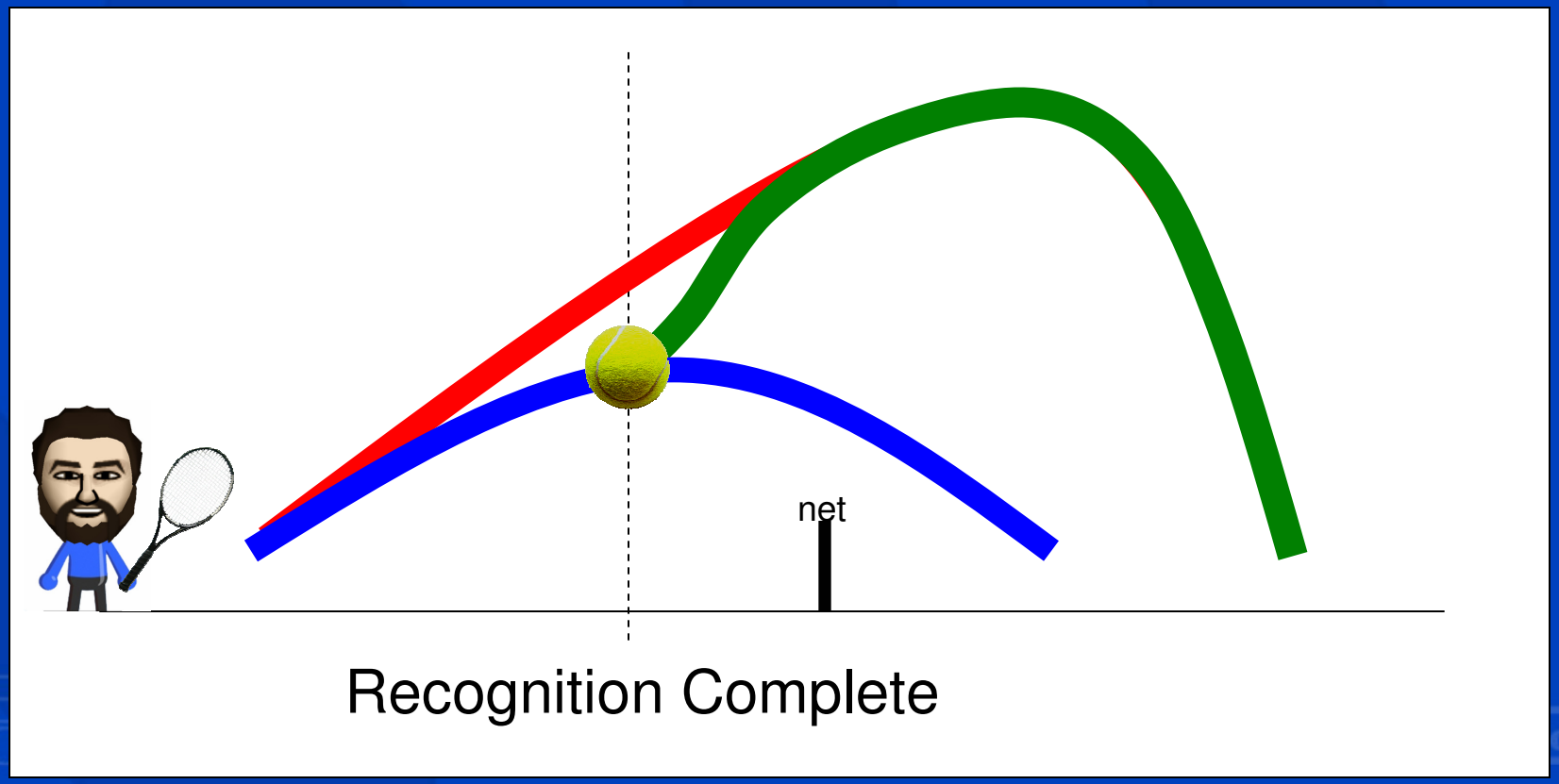
Complex Gesture Recognition:
Wii Sports Tennis Timeline

- Velocity and spin are detected late

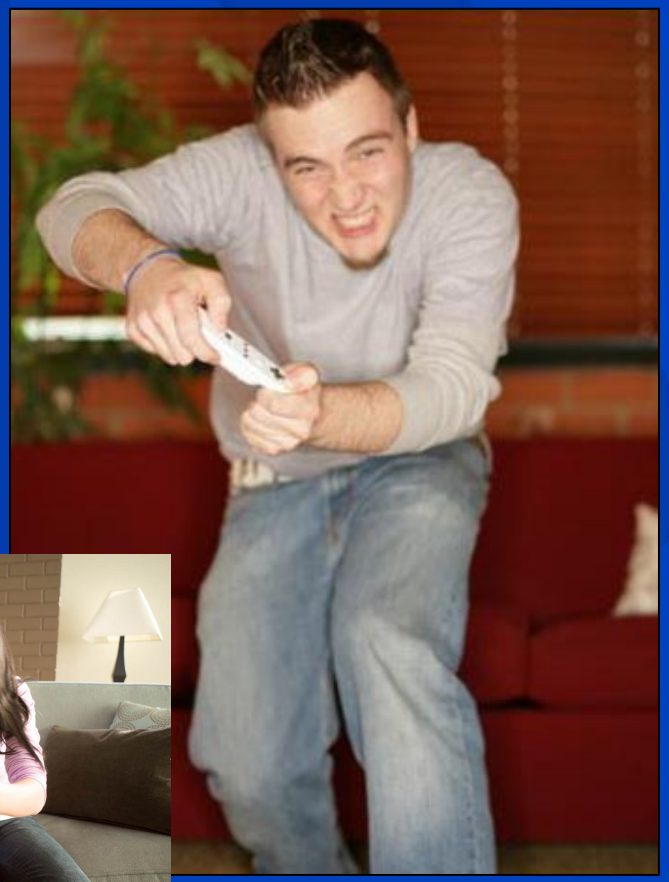


Complex Gesture Recognition:
Wii Sports Tennis Timeline

- Interpolate ball to desired trajectory



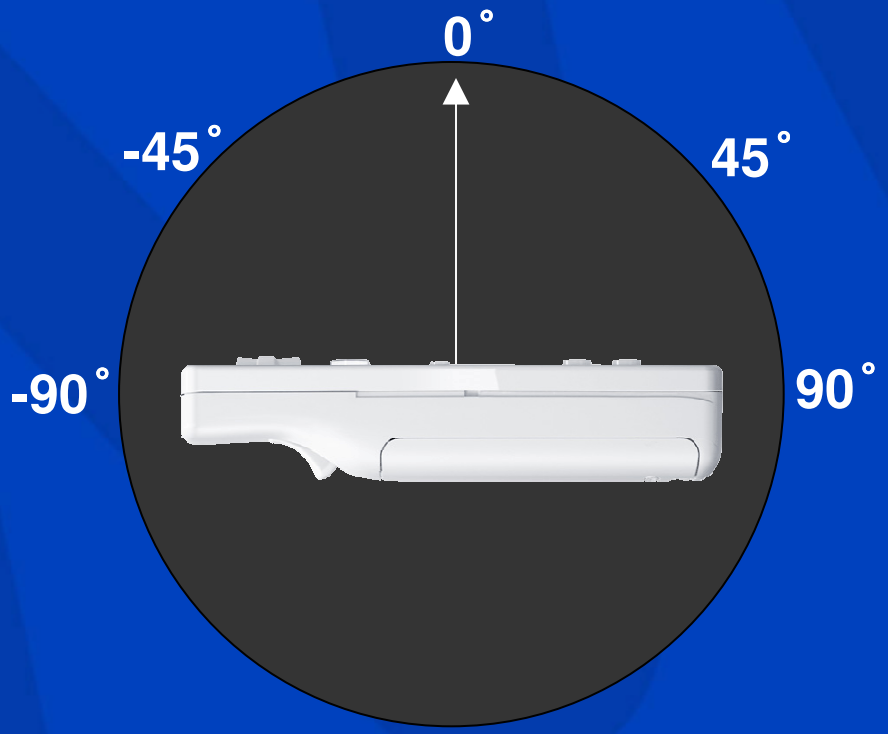
Accelerometer Applications: Steering



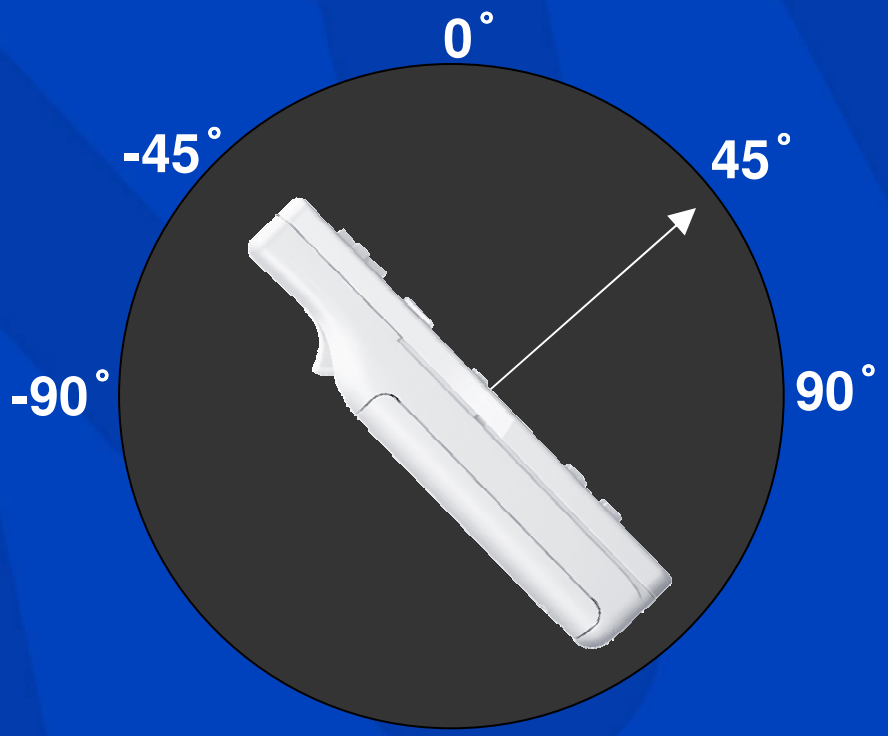
Steering and Rotating

- Robust and reliable
- Various orientations
  - Sideways
  - Paper airplane
  - Flight stick

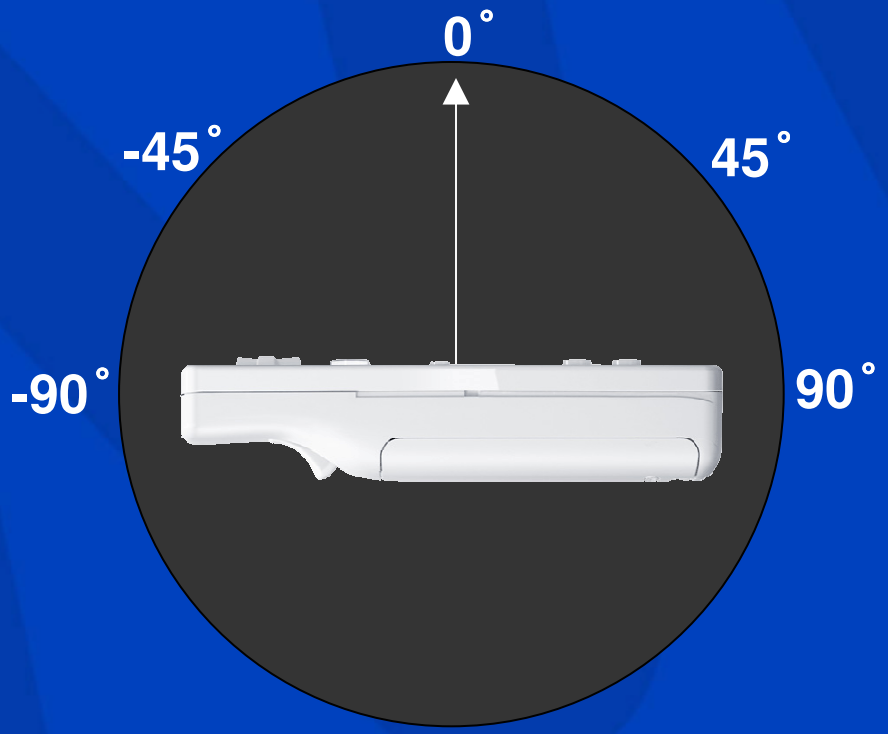
Steering and Rotating: Desired Angles



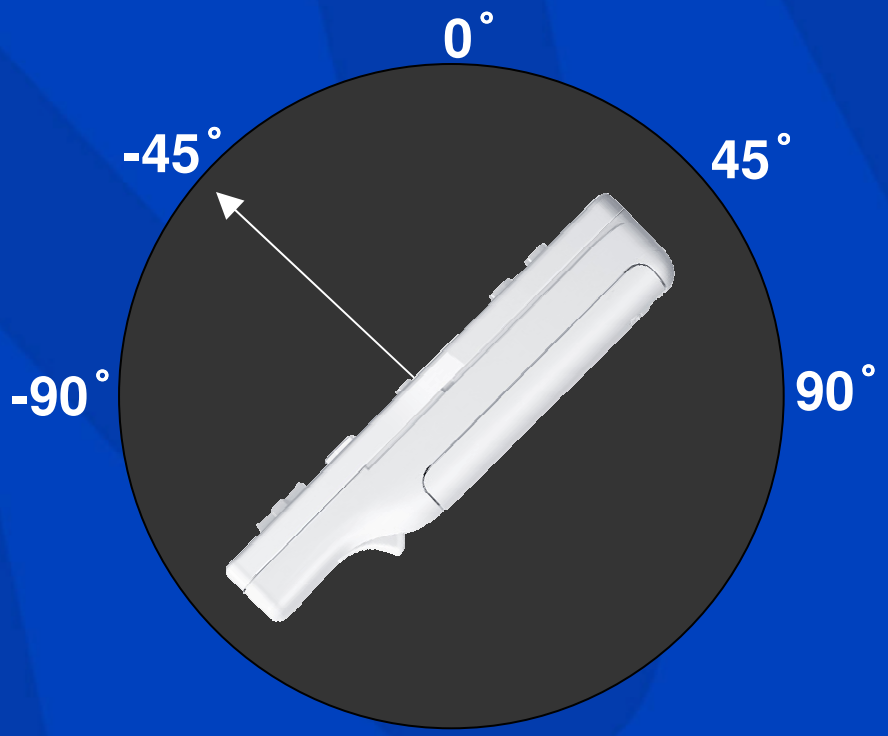
Steering and Rotating: Desired Angles



Steering and Rotating: Desired Angles

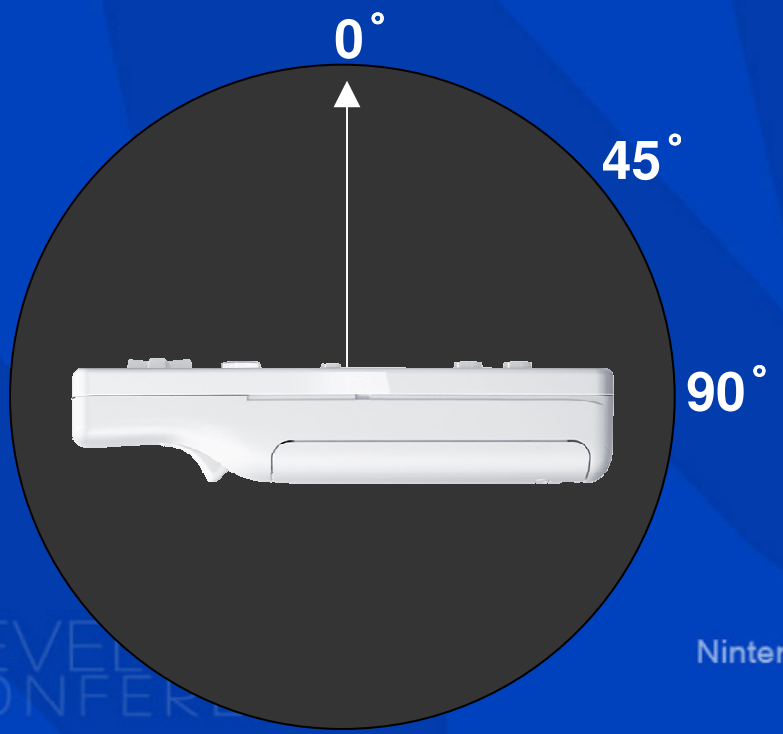


Steering and Rotating: Desired Angles



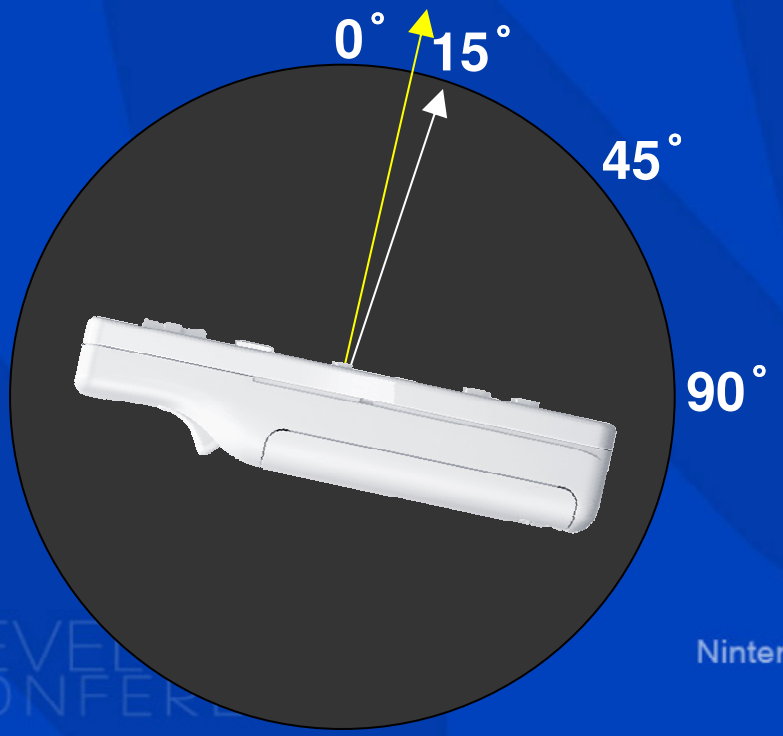
Steering and Rotating: Angle Conversion

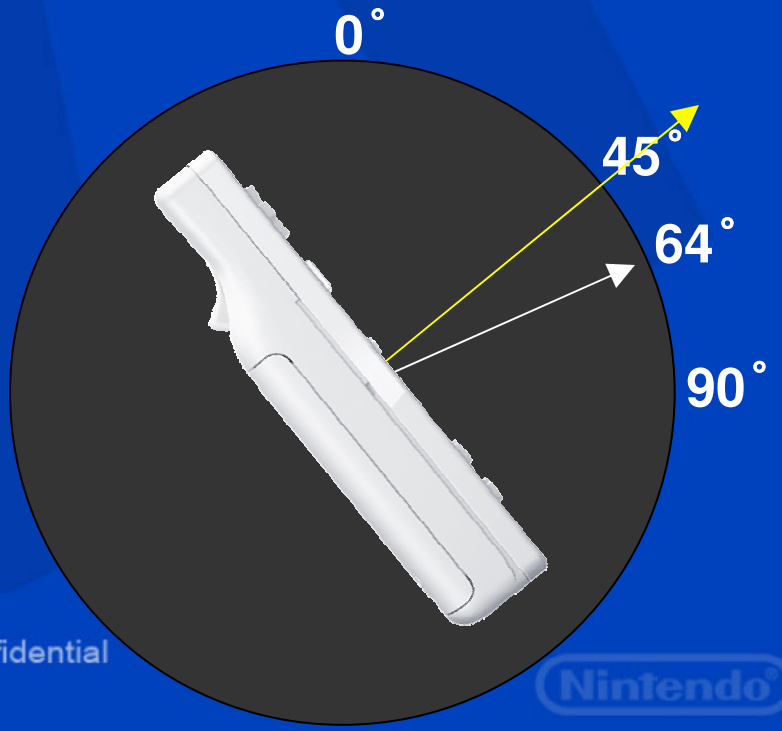
- Wrong way
  - Multiply z-axis by 90 degrees



Steering and Rotating: Angle Conversion

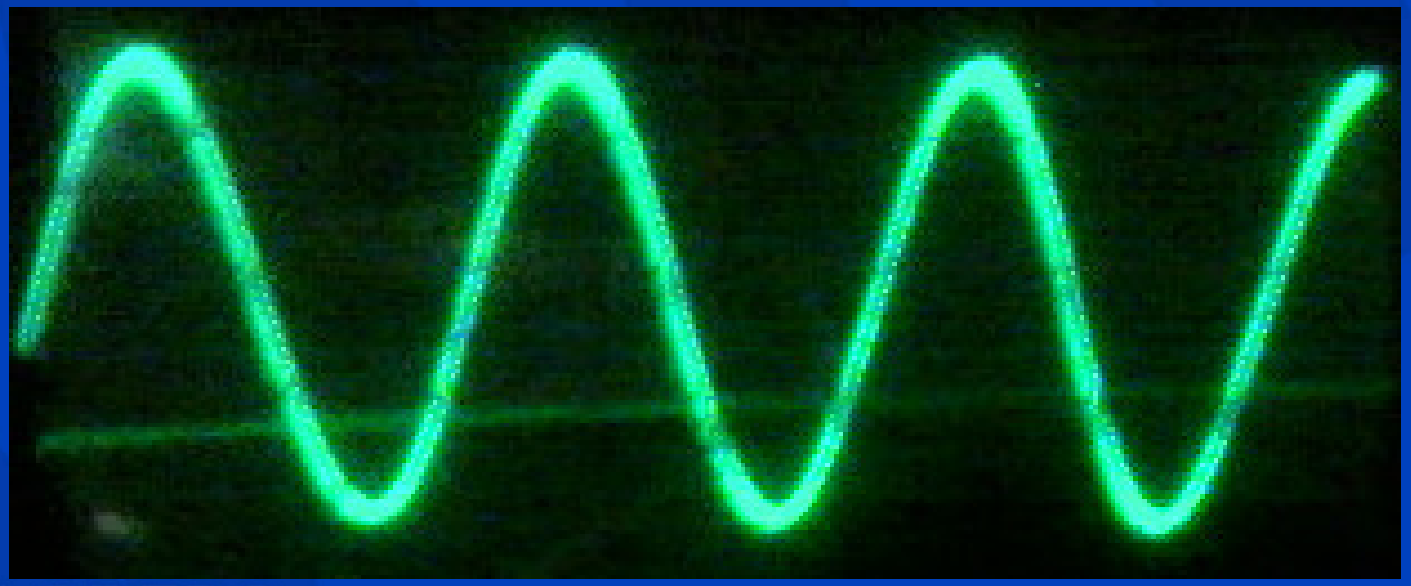
- Wrong way (multiply z-axis by 90 degrees)
  - Close, but causes "swerving" near zero degrees



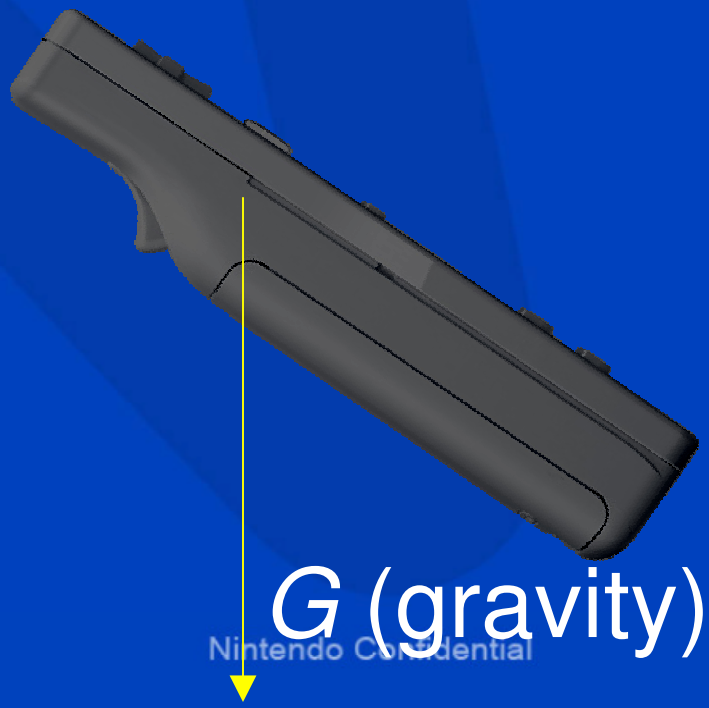


Steering and Rotating: Angle Conversion

- Correct Way
  - Use trigonometry (sin or cos)

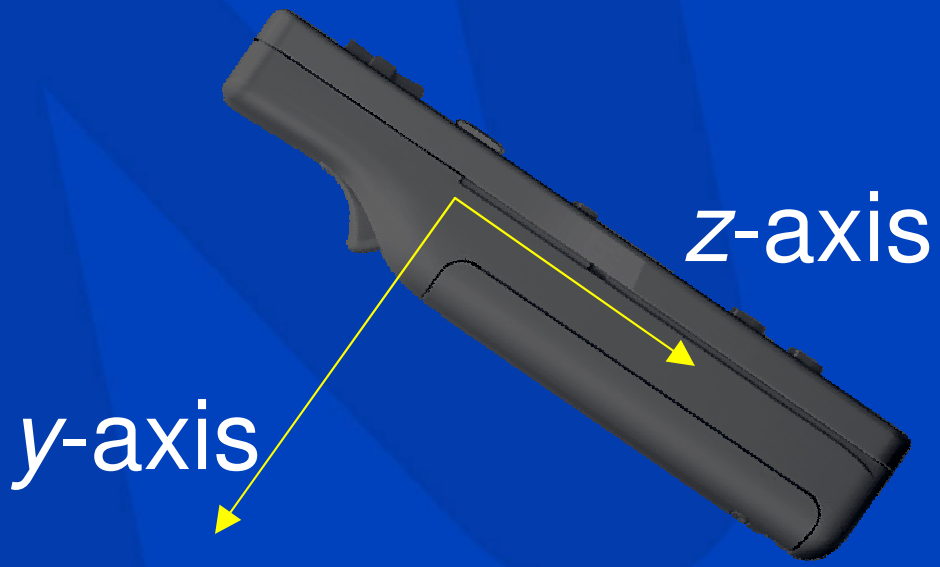


Steering and Rotating: Trigonometry Visualization

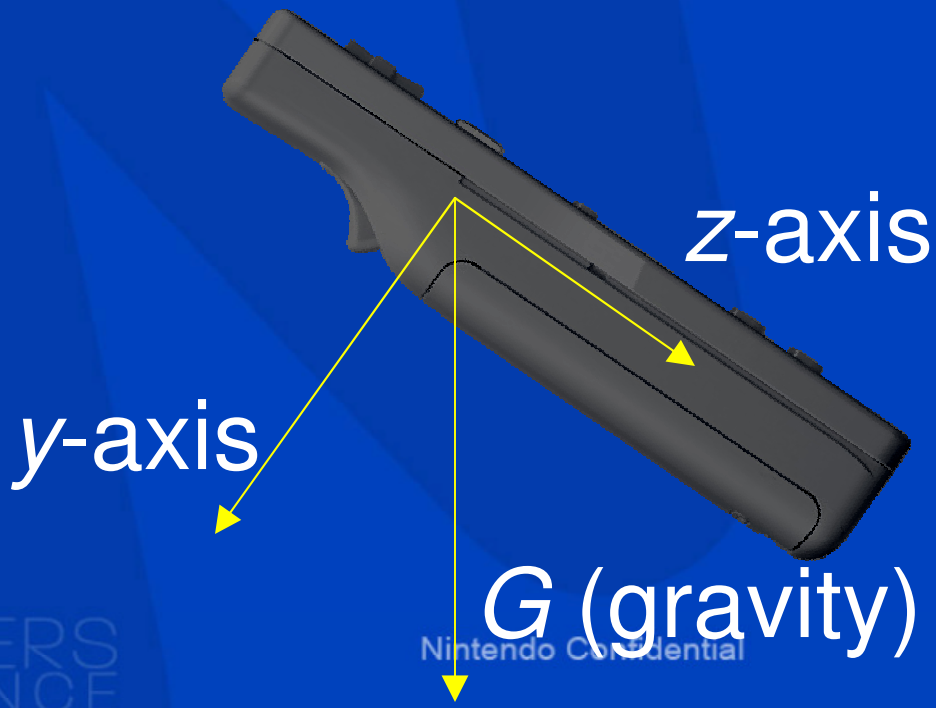


G (gravity)

Steering and Rotating: Trigonometry Visualization

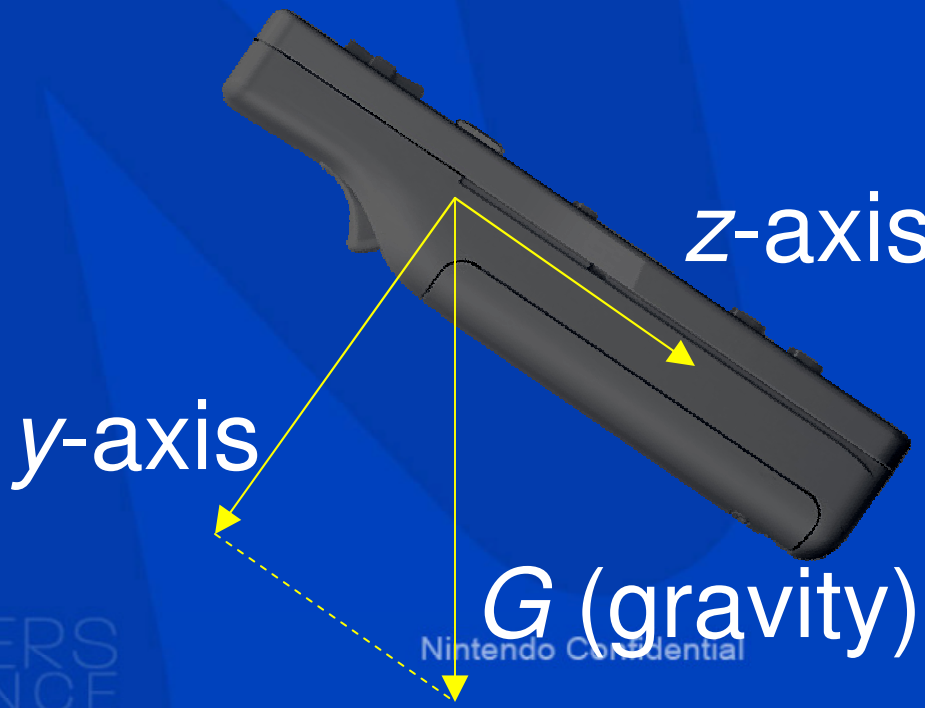


Steering and Rotating: Trigonometry Visualization



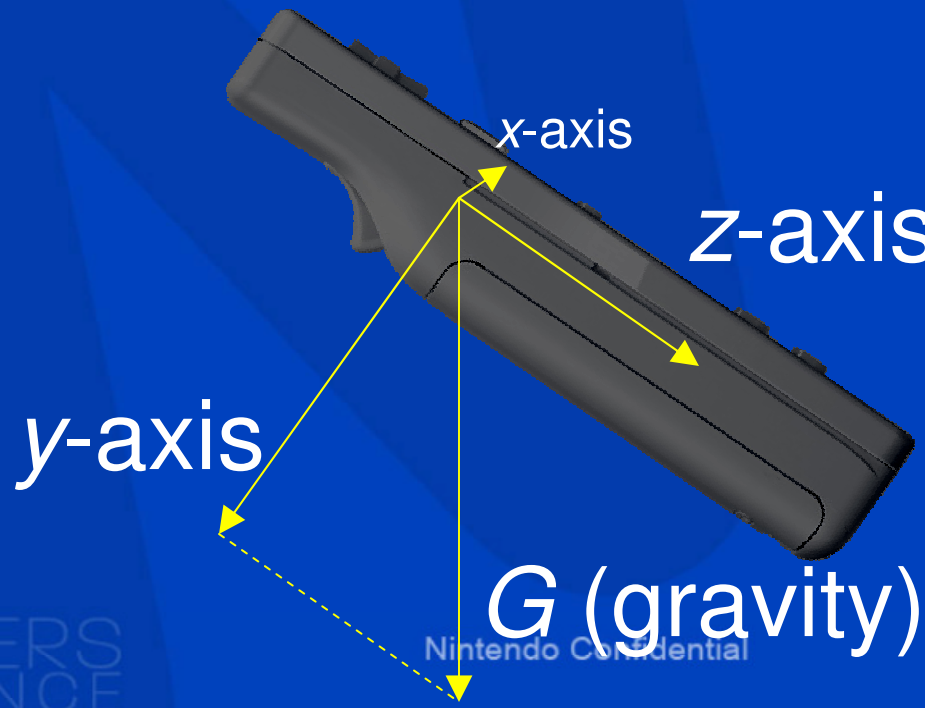
Steering and Rotating: Trigonometry Visualization

$$G = \sqrt{yAxisAcceleration^2 + zAxisAcceleration^2}$$



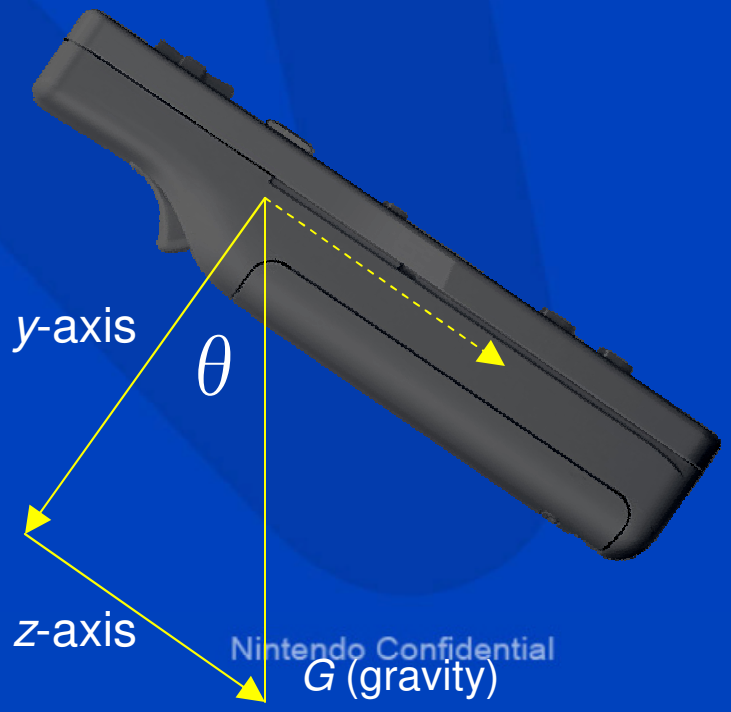
Steering and Rotating: Trigonometry Visualization

$$G = \sqrt{yAxisAcceleration^2 + zAxisAcceleration^2}$$



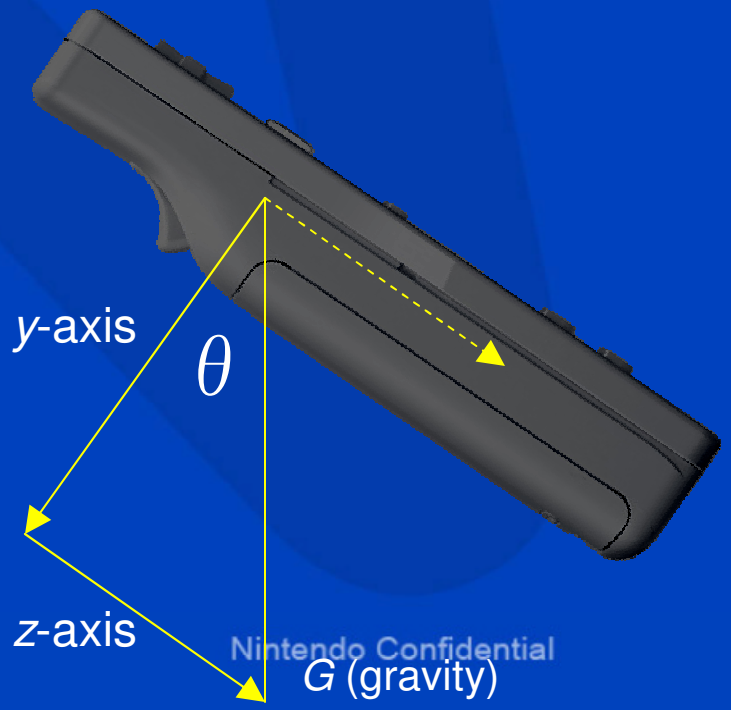
G (gravity) ≠ 1.0!

Steering and Rotating: Trigonometry Visualization



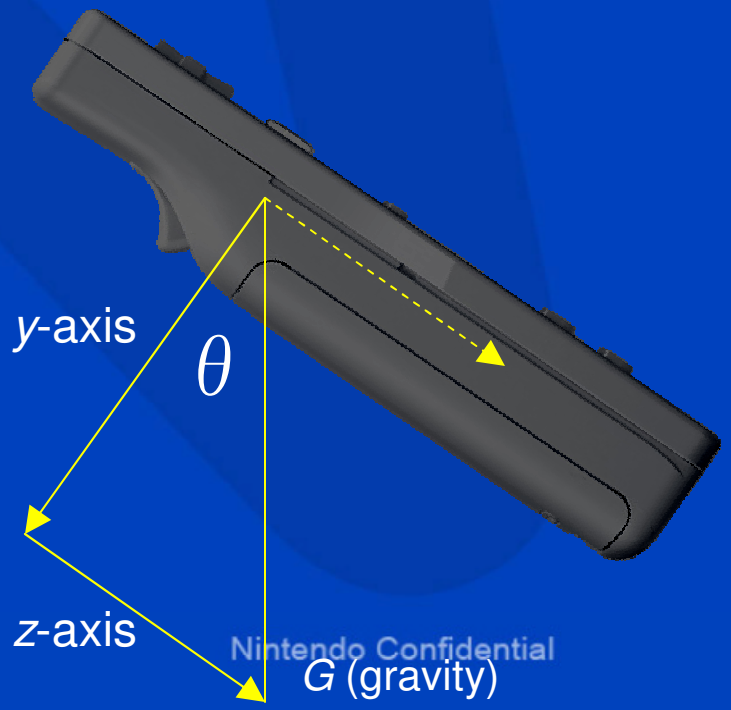
Steering and Rotating: Trigonometry Visualization

$$\sin(\theta) = \frac{\text{opposite}}{\text{hypotenuse}} = \frac{\text{zAxisAcceleration}}{\sqrt{\text{yAxisAcceleration}^2 + \text{zAxisAcceleration}^2}}$$



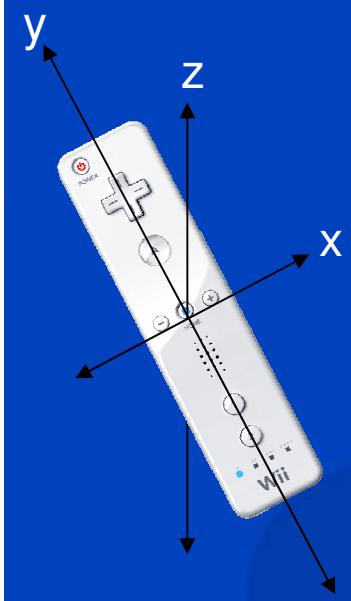
Steering and Rotating: Trigonometry Visualization

$$\theta = \arcsin\left(\frac{\text{zAxisAcceleration}}{\sqrt{\text{yAxisAcceleration}^2 + \text{zAxisAcceleration}^2}}\right)$$

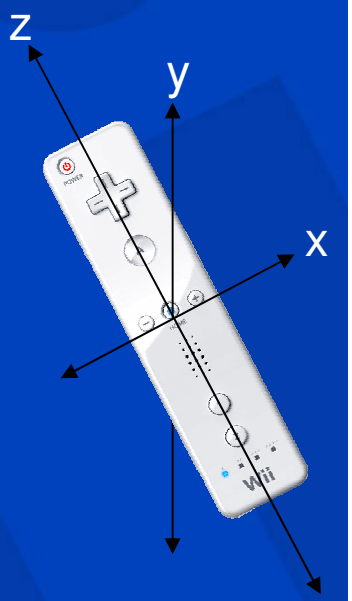


Avoiding Jitter in Steering

- Player's hands are shaky
  - Smooth out accelerometer data
  - KPADSetAccelParam(chan, play, sensitivity);
    - <play> should be between 0 and 0.05



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
- Steering
  - Remember to use trigonometry
  - Swerving is a sign that it was implemented wrong
  - Use KPAD to smooth values

Questions?

Ask me during the reception/breaks

Or e-mail support@noa.com