3D User Interfaces for Games and Virtual Reality

Lecture #4: Video Game Motion Controllers

Spring 2014

Joseph J. LaViola Jr.

3D Spatial Input Hardware – The Past

Intersense IS-900

Polhemus Patriot

3rd Tech Hi Ball

These Devices cost thousands of Dollars!!

Spring 2014 CAP6121 – 3D User Interfaces for Games and Virtual Reality ©Joseph J. LaViola Jr.

3D Spatial Input Hardware – Today

PlayStation Move

Nintendo Wiimote

Microsoft Kinect

Razer Hydra

These Devices cost hundreds of Dollars!!

Lecture Outline

- Discuss video game motion controller hardware characteristics
  - Nintendo Wiimote
  - Microsoft Kinect
  - PlayStation Move
- Quick start guide for programming
- Case Studies

Devices

The Wiimote Device

- Wiimote features
  - uses Bluetooth for communication
  - senses acceleration along 3 axes
  - optical sensor for pointing (uses sensor bar)
  - provides audio and rumble feedback
  - standard buttons and trigger
  - uses 2 AA batteries
- Supports two handed interaction
  - can use 2 Wiimotes simultaneously
- Easily expandable

Wiimote Attachments

Nunchuk

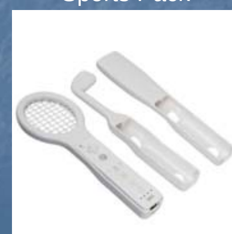
Steering Wheel

Zapper

Wii Helm

Boxing Gloves

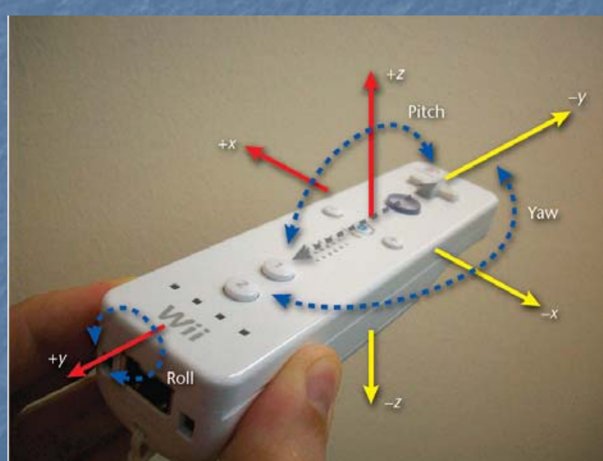
Sports Pack



Fishing Reel

The Wiimote – Coordinates

Wiimote Coordinates

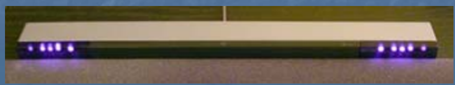


Spring 2014 CAP6121 – 3D User Interfaces for Games and Virtual Reality ©Joseph J. LaViola Jr.

The Wiimote – Optical Data

- Data from optical sensor
  - uses sensor bar
    - 10 LED lights (5 of each side)
    - accurate up to 5 meters
  - triangulation to determine depth
    - distance between two points on image sensor (variable)
    - distance between LEDs on sensor bar (fixed)
  - roll (with respect to ground) angle can be calculated from angle of two image sensor points
- Advantages
  - provides a pointing tool
  - gives approximate depth
- Disadvantages
  - line of sight, infrared light problems
  - only constrained rotation understanding

Sensor Bar



The Wiimote – Motion Data

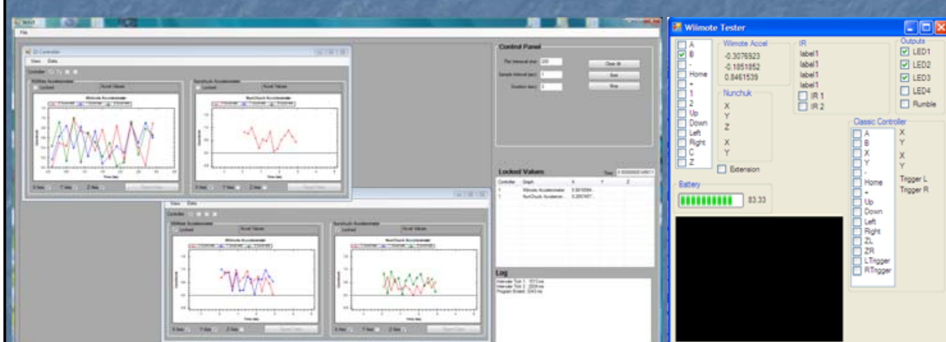
- Data from 3-axis accelerometer
  - senses instantaneous acceleration on device (i.e., force) along each axis
  - arbitrary units (+/- 3g)
  - always sensing gravity
    - at rest acceleration is g (upward)
    - freefall acceleration is 0
  - finding position and orientation
    - at rest – roll and pitch can be calculated easily
    - in motion – math gets more complex
    - error accumulation causes problems
    - often not needed – gestures sufficient
- Advantages
  - easily detect course motions
  - mimic many natural actions
- Disadvantages
  - ambiguity issues
  - player cheating
  - not precise (not a 6 DOF tracker)

The Wii Motion Plus

- Current Wiimote device
  - gives user a lot of useful data
  - not perfect
    - ambiguities
    - poor range
    - constrained input
  - Wii Motion Plus
    - moving toward better device
    - finer control
    - uses dual axis “tuning fork” angular rate gyroscope
    - true linear motion and orientation

Visualizing Wiimote Data

- Important to see data to understand device



Microsoft Kinect

- Kinect features
  - RGB camera
  - depth sensors
  - multi-array mic
  - motorized tilt
  - connects via USB
- Supports controllerless interface
- Full body tracking

Kinect – Hardware Details

- RGB Camera
  - 640 x 480 resolution at 30Hz
- Depth Sensor
  - complimentary metal-oxide semiconductor (CMOS) sensor (30 Hz)
  - infrared laser projector
  - 850mm to 4000mm distance range
- Multi-array mic
  - set of four microphones
  - multi-channel echo cancellation
  - sound position tracing
- Motorized tilt
  - 27^{o} up or down

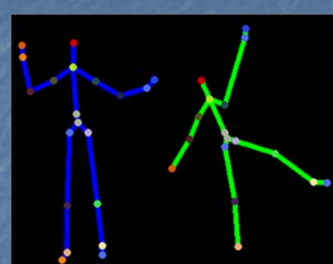
www.hardwaresphere.com

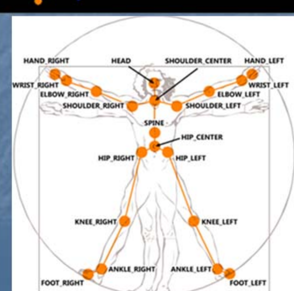
Kinect – Extracting 3D Depth

- Infrared laser projector emits known dot pattern
- CMOS sensor reads depth of all pixels
  - 2D array of active pixel sensors
    - photo detector
    - active amplifier
- Finds location of dots
- Computes depth information using stereo triangulation
  - normally needs two cameras
  - laser projector acts as second camera
- Depth image generation

Kinect – Skeleton Tracking

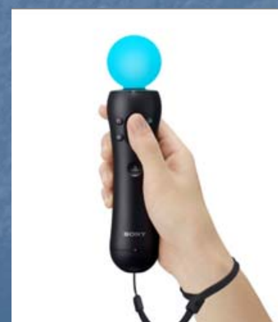
- Combines depth information with human body kinematics
  - 20 joint positions
- Object recognition approach
  - per pixel classification
  - decision forests (GPU)
  - millions of training samples
- See Shotton et al. (CVPR 2011)





PlayStation Move

- Consists of
  - Playstation Eye
  - 1 to 4 Motion controllers
- Features
  - combines camera tracking with motion sensing
  - 6 DOF tracking (position and orientation)
  - several buttons on front of device
  - analog T button on back of device
  - vibration feedback
  - wireless



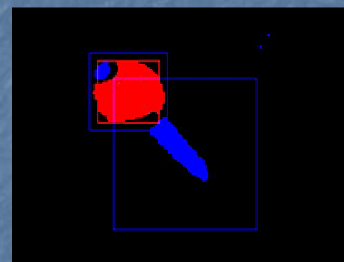
PlayStation Move – Hardware

- PlayStation Eye
  - 640 x 480 (60Hz)
  - 320 x 240 (120Hz)
  - microphone array
- Move Controller
  - 3 axis accelerometer
  - 3 axis angular rate gyro
  - magnetometer (helps to calibrate and correct for drift)
  - 44mm diameter sphere with RGB LED
    - used for position recovery
    - invariant to rotation
    - own light source
    - color ensures visual uniqueness

www.hardwaresphere.com

PlayStation Move – 6 DOF Tracking

- Image Analysis
  - find sphere in image
    - segmentation
      - label every pixel being tracked
      - saturated colors more robust
    - pose recovery
      - convert 2D image to 3D pose
      - robust for certain shapes (e.g., sphere)
  - fit model to sphere projection
    - size and location used as starting point
    - 2D perspective projection of sphere is ellipse
    - given focal length and size of sphere, 3D position possible directly from 2D ellipse parameters



PlayStation Move – 6 DOF Tracking

- Sensor Fusion
  - combines results from image analysis with inertial sensors (Unscented Kalman Filter)
  - contributions
    - camera – absolute 3D position
    - accelerometer
      - pitch and roll angles (when controller is stationary)
      - controller acceleration (when orientation is known)
      - reduce noise in 3D position and determine linear velocity
    - gyroscope
      - angular velocity to 3D rotation
      - angular acceleration

Initialize with:

$$\hat{\mathbf{x}}_0 = E[\mathbf{x}_0]$$
$$\mathbf{P}_0 = E[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T]$$
$$\hat{\mathbf{x}}_0^a = E[\mathbf{x}^a] = [\hat{\mathbf{x}}_0^T \; \mathbf{0} \; \mathbf{0}]^T$$
$$\mathbf{P}_0^a = E[(\mathbf{x}_0^a - \hat{\mathbf{x}}_0^a)(\mathbf{x}_0^a - \hat{\mathbf{x}}_0^a)^T] = \begin{bmatrix} \mathbf{P}_0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{P_v} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{P_n} \end{bmatrix}$$

For $k \in \{1, \dots, \infty\}$,

Calculate sigma points:

$$\mathcal{X}_{k-1}^a = \left[\hat{\mathbf{x}}_{k-1}^a \;\; \hat{\mathbf{x}}_{k-1}^a \pm \sqrt{(L+\lambda)\mathbf{P}_{\mathbf{k-1}}^{\mathbf{a}}}\right]$$

Time update:

$$\mathcal{X}_{k|k-1}^x = \mathbf{F}[\mathcal{X}_{k-1}^x, \mathcal{X}_{k-1}^v]$$
$$\hat{\mathbf{x}}_k^- = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{X}_{i,k|k-1}^x$$
$$\mathbf{P}_k^- = \sum_{i=0}^{2L} W_i^{(c)} [\mathcal{X}_{i,k|k-1}^x - \hat{\mathbf{x}}_k^-][\mathcal{X}_{i,k|k-1}^x - \hat{\mathbf{x}}_k^-]^T$$
$$\mathcal{Y}_{k|k-1} = \mathbf{H}[\mathcal{X}_{k|k-1}^x, \mathcal{X}_{k-1}^n]$$
$$\hat{\mathbf{y}}_k^- = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{Y}_{i,k|k-1}$$

Measurement update equations:

$$\mathbf{P}_{\tilde{\mathbf{y}}_k \tilde{\mathbf{y}}_k} = \sum_{i=0}^{2L} W_i^{(c)} [\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-][\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-]^T$$
$$\mathbf{P}_{\mathbf{x}_k \mathbf{y}_k} = \sum_{i=0}^{2L} W_i^{(c)} [\mathcal{X}_{i,k|k-1} - \hat{\mathbf{x}}_k^-][\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-]^T$$
$$\mathcal{K} = \mathbf{P}_{\mathbf{x}_k \mathbf{y}_k} \mathbf{P}_{\tilde{\mathbf{y}}_k \tilde{\mathbf{y}}_k}^{-1}$$
$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathcal{K}(\mathbf{y}_k - \hat{\mathbf{y}}_k^-)$$
$$\mathbf{P}_k = \mathbf{P}_k^- - \mathcal{K} \mathbf{P}_{\tilde{\mathbf{y}}_k \tilde{\mathbf{y}}_k} \mathcal{K}^T$$

where, $\mathbf{x}^a = [\mathbf{x}^T \; \mathbf{v}^T \; \mathbf{n}^T]^T$, $\mathcal{X}^a = [(\mathcal{X}^x)^T \; (\mathcal{X}^v)^T \; (\mathcal{X}^n)^T]^T$, λ=composite scaling parameter, L=dimension of augmented state, $\mathbf{P_v}$=process noise cov., $\mathbf{P_v}$=measurement noise cov., W_i=weights as calculated in Eqn. 15.

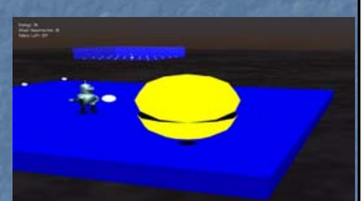
Algorithm 3.1: Unscented Kalman Filter (UKF) equations

www.cslu.ogi.edu/nsel/ukf/node6.html

Programming

Programming with the Wiimote

- Connect to computer
  - does not work for every bluetooth device
- Obtain Wiimote software
  - many variations and APIs (C,C++, C#, Java, Flash)
    - Brian Peek's API (www.coding4fun.com)
      - low level API
    - Paul Varcholik's XNA 3DUI Framework (www.bespokesoftware.org)
      - contained within larger framework
      - include gesture recognizer
    - Unity 3D
- Write code and enjoy (Wingrave et al. 2010)
  - integration
  - heuristics
  - gesture analysis and recognition



Kinect Programming

- Two main approaches
  - NITE and Open NI
  - Microsoft Kinect SDK

Kinect – Microsoft SDK

- Uses subset of technology from Xbox 360 dev version
- Access to microphone array
- Sound source localization (beamforming)
  - connection with Microsoft Speech SDK
- Kinect depth data
- Raw audio and video data
- Access to tilt motor
- Skeleton tracking for up to two people
- Examples and documentation

Kinect SDK – Joints

- Two users can be tracked at once
- <x,y,z> joints in meters
- Each joint has a state
  - tracked, not tracked, inferred
- Inferred – occluded, clipped, or no confidence
- Not tracked – rare but needed for robustness

Kinect SDK – Example

```
using Microsoft.Research.Kinect.Nui;
Runtime nui;

private void Window_Loaded(object sender, EventArgs e) {

     nui = new Runtime();

     try
     {
        nui.Initialize(RuntimeOptions.UseDepthAndPlayerIndex | RuntimeOptions.UseSkeletalTracking |
        RuntimeOptions.UseColor);
      }
      catch (InvalidOperationException)
      {
         System.Windows.MessageBox.Show("Runtime initialization failed. Please make sure Kinect device
         is plugged in.");
         return;
       }

       nui.SkeletonFrameReady += new EventHandler<SkeletonFrameReadyEventArgs>(nui_SkeletonFrameReady);
 }
```

Kinect SDK Example

```
void nui_SkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e)
        {
            SkeletonFrame skeletonFrame = e.SkeletonFrame;
            int iSkeleton = 0;
            Brush[] brushes = new Brush[6];
            brushes[0] = new SolidColorBrush(Color.FromRgb(255, 0, 0));
            brushes[1] = new SolidColorBrush(Color.FromRgb(0, 255, 0));
            brushes[2] = new SolidColorBrush(Color.FromRgb(64, 255, 255));
            brushes[3] = new SolidColorBrush(Color.FromRgb(255, 255, 64));
            brushes[4] = new SolidColorBrush(Color.FromRgb(255, 64, 255));
            brushes[5] = new SolidColorBrush(Color.FromRgb(128, 128, 255));

            skeleton.Children.Clear();
            foreach (SkeletonData data in skeletonFrame.Skeletons)
            {
                if (SkeletonTrackingState.Tracked == data.TrackingState)
                {
                    // Draw bones
                    Brush brush = brushes[iSkeleton % brushes.Length];
                    skeleton.Children.Add(getBodySegment(data.Joints, brush, JointID.HipCenter,
                                          JointID.Spine,JointID.ShoulderCenter, JointID.Head));
                    skeleton.Children.Add(getBodySegment(data.Joints, brush, JointID.ShoulderCenter,
                                          JointID.ShoulderLeft, JointID.ElbowLeft, JointID.WristLeft,
                                          JointID.HandLeft));
```


Kinect SDK Example

```
                skeleton.Children.Add(getBodySegment(data.Joints, brush, JointID.ShoulderCenter,
                                      JointID.ShoulderRight, JointID.ElbowRight, JointID.WristRight, JointID.HandRight));
                skeleton.Children.Add(getBodySegment(data.Joints, brush, JointID.HipCenter, JointID.HipLeft,
                                      JointID.KneeLeft, JointID.AnkleLeft,  JointID.FootLeft));
                skeleton.Children.Add(getBodySegment(data.Joints, brush, JointID.HipCenter, JointID.HipRight,
                                      JointID.KneeRight, JointID.AnkleRight, JointID.FootRight));

// Draw joints
   foreach (Joint joint in data.Joints)
   {
    Point jointPos = getDisplayPosition(joint);
    Line jointLine = new Line();
    jointLine.X1 = jointPos.X - 3;
    jointLine.X2 = jointLine.X1 + 6;
    jointLine.Y1 = jointLine.Y2 = jointPos.Y;
    jointLine.Stroke = jointColors[joint.ID];
    jointLine.StrokeThickness = 6;
    skeleton.Children.Add(jointLine);
   }
  }
  iSkeleton++;
 } // for each skeleton
}
```

Spring 2014 CAP6121 – 3D User Interfaces for Games and Virtual Reality ©Joseph J. LaViola Jr.

Kinect SDK Example

```
Polyline getBodySegment(Microsoft.Research.Kinect.Nui.JointsCollection joints, Brush brush, params JointID[] ids)
        {
            PointCollection points = new PointCollection(ids.Length);
            for (int i = 0; i < ids.Length; ++i )
            {
                points.Add(getDisplayPosition(joints[ids[i]]));
            }

            Polyline polyline = new Polyline();
            polyline.Points = points;
            polyline.Stroke = brush;
            polyline.StrokeThickness = 5;
            return polyline;
        }
```

Microsoft Kinect SDK Documentation

http://msdn.microsoft.com/en-us/library/hh855347.aspx

PlayStation Move – Programming

- Move.Me
- Uses PS3 as device server
- Up to four controllers at once
- Controller state info
  - 3D position and orientation
  - 3D velocity and acceleration
  - 3D angular velocity and acceleration
  - button and tracking status
- Set color of sphere and initiate rumble feedback

Move.Me Code Snippets

Connecting to Move.Me Server

```
public void Connect(String server, int port)
        {
            _tcpClient = new TcpClient();
            _tcpClient.Connect(server, port);
            _udpClient = new UdpClient(0);
            Console.WriteLine("Initial recieve buffer size: {0}",
                                  _udpClient.Client.ReceiveBufferSize);
            _udpClient.Client.ReceiveBufferSize = 655360; // 640 KB
            Console.WriteLine("Expanded recieve buffer size: {0}",
                                   _udpClient.Client.ReceiveBufferSize);
            uint udpport = (uint)((IPEndPoint)_udpClient.Client.LocalEndPoint).Port;
            SendRequestPacket(ClientRequest.PSMoveClientRequestInit, udpport);
        }
```


Move.Me Code Snippets

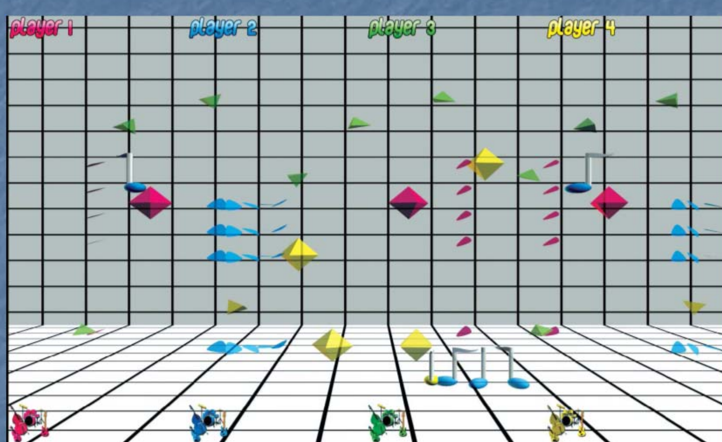
class PSMoveSharpGemState

```
public struct PSMoveSharpGemState
    {
        public Float4 pos;
        public Float4 vel;
        public Float4 accel;
        public Float4 quat;
        public Float4 angvel;
        public Float4 angaccel;
        public Float4 handle_pos;
        public Float4 handle_vel;
        public Float4 handle_accel;
        public PSMoveSharpPadData pad; // 4 bytes
        public Int64 timestamp;
        public float temperature;
        public float camera_pitch_angle;
        public UInt32 tracking_flags;
}
```

```
PSMoveSharpState state = moveClient.GetLatestState();
PSMoveSharpCameraFrameState camera_frame_state = moveClient.GetLatestCameraFrameState();
```

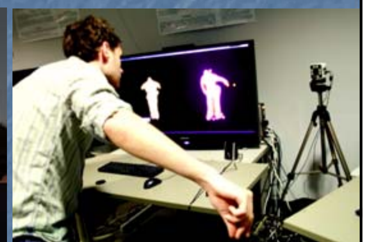
Case Studies

One Man Band



Bott et al., 2009

Real Dance



Charbonneau et al., 2009 Charbonneau et al., 2010 Charbonneau et al., 2011

Football

Williamson et al., 2010

Kinect Football by Andrew Devine

RealEdge – FPS

Williamson et al., 2011

Robots

Pfeil et al., 2013

Conclusions – Which to Choose?

- Wiimote
- Positives
  - cost ~ $40
  - buttons
  - something to hold in hand
- Negatives
  - not true 6 DOF
  - challenging to program
  - reasonable accuracy
  - no company support

Conclusions – Which to Choose?

- Microsoft Kinect
- Positives
  - cost ~ $130
  - full body tracking
    - joint position
    - joint orientation (not yet)
  - multimodal input
  - good SDK and support
- Negatives
  - no buttons (temporal segmentation problem)
  - more data to process
  - not really designed with physical props in mind
  - latency issues (gesture recognition)

Conclusions – Which to Choose?

- PlayStation Move
- Positives
  - accurate and fast 6 DOF tracking
  - buttons
  - multimodal input
  - good SDK and support
- Negatives
  - cost ~ $400 to $500
  - requires PS3 (positive as well)
  - does not track full body (more restrictive)

Next Class

- Visual displays
- Readings
  - Siggraph 2010, 2011 course notes on 3D UI and Video Game Hardware