3D User Interfaces for Games and Virtual Reality

Lecture #4: Video Game Motion Controllers
Spring 2017
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!!
3D Spatial Input Hardware – Today

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
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
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

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° up or down
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)
Kinect 2

- RGB Camera
  - HD resolution
- Depth Sensor
  - time of flight
- microphone array

- ToF – illuminate it with a beam of pulsed light and calculate time it takes for the light to be detected on an imaging device

Kinect 2 – Other Differences

- Greater accuracy
  - three times the fidelity over Kinect
- Can track without visible light using an active IR sensor
- Has a 60% wider field of view
  - detect a user up to 3 feet from the sensor compared to six feet for the Kinect
  - track up to 6 skeletons at once
- Detect a player’s heart rate and facial expressions,
- Position and orientation of 25 individual joints (including thumbs),
- Weight put on each limb and speed of player movements
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

**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
Programming with the Wiimote

- Connect to computer
  - does not work for every bluetooth device
- Obtain Wiimote software
  - many variations and APIs (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

- 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 2 JointServer – VS2013

- Gathers joint data from the Kinect 2
- Encodes data into a string and sends it over UDP socket
- Run from the VisualStudio or JointServer\bin\Debug\JointServer.exe
- Requires Kinect SDK 2.0
- This needs to be started before you press Play in Unity3D
- Can be left running, i.e. do not need to restart each time to press Play in Unity3D
JointUnity

- Main script – KinectSkeleton.cs
  - Receives data from UDP socket
  - Decodes it and updates joint values
  - This script has to be attached to some object in your scene to work
- Demo use script – SkeletonEmulator.cs
  - Example use of KinectSkeleton API

JointUnity API

- KinectSkeleton kinect
  - main object
- Dictionary<int, PlayerSkeleton> kinect.players
  - Dictionary of players
  - Access with player ID in range [0,5]
  - kinect.players[0] to get first player
JointUnity API

- PlayerSkeleton player = kinect.players[0]
  - Single player data
- bool player.isTracked
  - True if Kinect is currently tracking this player
- int player.id
  - Player ID
- Dictionary<JointType, SkeletonJoint> player.joints
  - Dictionary of joints
  - Access joint data with JointType enum
  - player.joints[JointType.Head] to get access to Head joint data

JointUnity API

- SkeletonJoint joint = player.joints[JointType.Head]
  - Single joint data
- bool joint.isTracked
  - True if Kinect is actively tracking the joint
  - False if the joint position is inferred
  - Inferred position can be very close to the truth or completely wrong.
- Vector3 joint.position
  - Current position of the joint in space relative to the Kinect
- JointType joint.type
  - Joint type
Notes

- Kinect 2 randomly assigns ID to players it sees.
- If you step out of the frame and back you will likely get a new ID.
- Due to this even with a single player in frame you will have to look through all 6 players in API to find one that isTracked.

- At times Kinect cannot see certain joints and it will guess their position.
- In KinectServer joints that are inferred will have thin lines drawn to the instead of thick color ones.
- Color of the skeleton displayed in KinectServer represents player ID.

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

class 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 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 (Kinect 2)
    - 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