Microsoft Kinect Intro

CMPS179 Game Design Practicum Lecture 1

John Murray
PhD Student
Who am I?

John Murray

I’m a PhD Student with the Expressive Intelligence Studio

I work on Augmented Reality related research, especially related to the Foresight and the Kinect

My Kinect experience ranges from a series of prototypes – one which, called “Preoccupied” integrates flying a UAV over a Bing map of campus

Also was at the launch of the Kinect for Windows SDK (Made a presentation/whiteboard demo)
This Lecture

General introduction to the concepts of the Kinect SDK

We’ll dig further into the details of working with Unity/XNA, gestures and other topics later in the course

Get you thinking around and through a new set of constraints

Prepare yourself: this course will move quickly, be demanding and require your best work.
What is the Kinect?

- RGB, Depth Sensor, and Multi-mic Array
- Works with structured light to determine distance for each pixel
Sensors/Components

- IR Emitter
- Color Sensor
- IR Depth Sensor
- Microphone Array
- Tilt Motor
History

• Original technology developed in 2005
• Announced in 2009 as codenamed Natal after the city and because of its relation to being “of or related to birth”
• Released Kinect for Windows Beta on June 16, 2011
• On February 1\textsuperscript{st}, released commercial version (which you now have access to for this class)
What does it do?

• Tracks location of 48 skeletal points, after massive amounts of data provided to algorithms developed by Microsoft (for their SDK)
• OpenNI has similar capabilities, though not as easy to use.
• Provides the first affordable depth sensor.
Kinect Titles

• Roughly categorized based on primary mechanics:
  • 1-1 games have a representation of an avatar on screen
  • Abstract games have some
1-1 Games

• Dance Central, Dance Paradise, Just Dance
• Zumba, YourShape Fitness
• Fighters Unleashed (less so)

• All of these titles track the player’s body, to some degree, and represent it in a space.
“Abstract” Games

• Child of Eden
• Gadgets and “hacks”
• Happy Action Theater
• Others?
More “Traditional” Games

• Star Wars
• Mass Effect 3
• Fable: The Journey
• Steel Battalion
Gadgets

Build A Buddy, Air Band, Kinect Googly Eyes, Kinect Me, Bobblehead, Kinect Sparkler, Junk Fu and Avatar Kinect

Short form games/experiences. Toys, involves limited gameplay mechanics, but explores possibilities of the depth camera and skeleton tracking.
Fundamental Data

- Skeletal Data: Joints in 3D space
- Provided in meters
- Depth Data: “Near” and “Far” modes. Depth of each pixel.
- Image Data: Different resolutions/frame rates
- Audio Array: Speech Recognition SDK
Kinect Bookkeeping

- Manage Kinect state
  - Connected
  - Enable Color, Depth, Skeleton
  - Start Kinect

- Get Data
  - Events - AllFramesReady
  - Polling – OpenNextFrame

- API support for detecting and managing device status changes, such as device unplugged, device plugged in, power unplugged, etc. Apps can reconnect to the Kinect device after it is plugged in, after the computer returns from suspend, etc. See the Shape Game sample code for the best example.
Speech Recognition

• Very cool

<!-- Confirmation“Yes”._value: string ["Yes"]-->
<rule id="Confirmation_Yes" scope="public">
  <example> yes </example>
  <example> yes please </example>
  <one-of>
    <item> yes </item>
    <item> yeah </item>
    <item> yep </item>
    <item> ok </item>
  </one-of>
  <item repeat="0-1"> please </item>
  <tag> out._value = "Yes";</tag>
</rule>

Sound Position

• Sound Source Angle – the angle and confidence level of where audio is coming from
• Beam Angle – The angle used to record audio that you can set as a “directional microphone”

```javascript
var grammar = new Choices();
grammar.Add("yes please");
grammar.Add("yes");
grammar.Add("yeah");
grammar.Add("ok");
```
Camera Data

• Events return ImageFrame
  – PixelDataLength
  – FrameNumber
  – Timestamp
  – Dimensions: Height, Width

• Use AllFramesReady event to synchronize
RESOLUTIONS

• Color
  – 12 FPS: 1280X960 RGB
  – 15 FPS: Raw YUV 640x480
  – 30 FPS: 640x480

• Depth
  – 30 FPS: 80x60, 320x240, 640x480
Depth Data

• Returns the **distance** and **player** for every pixel
  – Ex: 320x240 = 76,800 pixels

• Distance
  – Distance in mm from Kinect ex: 2,000mm (6.56 feet)

• Player
  – 1-6 players
# Depth Continued

<table>
<thead>
<tr>
<th>Mode</th>
<th>Depth &amp; Player</th>
<th>Center Hip Joint</th>
<th>Other 19 Joints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Near</td>
<td>Yes</td>
<td>Yes</td>
<td>No, for v1.0</td>
</tr>
</tbody>
</table>

**Distance Formula**

```c
int depth = depthPoint >>
DepthImageFrame.PlayerIndexBitmaskWidth;
```

**Player Formula**

```c
int player = depthPoint &
DepthImageFrame.PlayerIndexBitmask;
```
SKELETON DATA

Kinect Intro – CMPS179

Sensor Direction

Expressive Intelligence Studio http://eis.ucsc.edu
SKELETAL JOINTS

- Each player with set of $<x, y, z>$ joints in meters
- Each joint has associated state
  - Tracked, Not tracked, or Inferred
- Inferred - Occluded, clipped, or low confidence joints
- Use TransformSmoothParameters to smooth joint data to reduce jitter
nui.SkeletonEngine.TransformSmooth = true;
    TransformSmoothParameters parameters = new TransformSmoothParameters();
    parameters.Smoothing = 0.7f;
    parameters.Correction = 0.3f;
    parameters.Prediction = 0.4f;
    parameters.JitterRadius = 1.0f;
    parameters.MaxDeviationRadius = 0.5f;
    nui.SkeletonEngine.SmoothParameters = parameters;
Gotchas

• Infers location of occluded joints
• Need to smooth for tracking and representations of locations
• Delay is noticeable
Closing Thoughts/Questions

• What are your first thoughts for games using the Kinect?
• User interface paradigms/Communication with user. FEEDBACK!
• Still wide open as to what can be done