Table of Contents

1. Laboratory on LEGO Mindstorms - Programming NXT in C# 2
2. Laboratory on LEGO Mindstorms - Programming EV3 in C# 2
2.1. Leap Motion Overview 2
2.1.1. Hand 2
2.1.2. Arm 2
2.1.3. Finger 2
2.1.4. Bone 3
2.2. Coordinates System 3
2.3. System Architecture 3
2.3.1. Native Application Interface 3
2.3.2. WebSocket Interface 3
2.4. Setting Up a Project 4
2.5. Sample source code 4
2.6. Citations 4

1. Laboratory on LEGO Mindstorms - Programming NXT in C#

2. Laboratory on LEGO Mindstorms - Programming EV3 in C#

2.1. Leap Motion Overview

The Leap Motion is an optic hand tracking device. It is able to track both hands and all 10 fingers as they move through the open space between the user and the controller, letting interaction with over 140 built-in apps for motion control. The Leap Motion Controller (LMC) itself uses optical sensors and InfraRed (IR) light imaging to define the position of predefined objects in real time. Updates are provided as a set (frame) of data. Each Frame object, representing a frame, contains properties (positions, orientations, gestures and motions) of all tracked objects at single moment of time. Then, the sensor software analyzes the objects movements detected in the device's field of view and translates it into actions to be developed. Actually a very small amount of processing is done on the device itself, in order to keep the cost of the units low. All the images are post-processed on the developer's computer. There the noises are being removed and the model of all objects is being created.

There are basically four classes that can be treated as the data model for the Leap Motion:

2.1.1. Hand

1. Reports the physical characteristics of a detected hand.
2. It's tracking data includes a palm position and velocity, vectors for the palm normal and direction to the fingers and lists of the attached fingers.
3. More information: https://developer.leapmotion.com/documentation/cpp/api/gen-cpp/class_leap_1_1_hand.html#details

Figure 1, The Hand PalmNormal and Direction define the orientation of the hand. Source: https://developer.leapmotion.com/documentation/

2.1.2. Arm

1. Bone-like class that represents the forearm.
2. Provides the orientation, length width and end points of an arm.
3. More information: Arm class details.

2.1.3. Finger

1. Class representing a tracked finger.
2. Valid finger objects can be obtained from a Frame or Hand object.
3. Information is provided for each finger on the tracked hand (characteristics or partially/not

visible fingers are estimated)
4. Fingers are identified by type name, i.e. thumb, index.
5. More information: Finger class details.

PAGE 2Figure 2.Finger TipPosition and Direction vectors provide the position of a finger tip and the general direction in which a finger is pointing. Source: https://developer.leapmotion.com/documentation/

2.1.4. Bone

1. Class representing a tracked bone.
2. Provided by a Finger object.
3. Bones are ordered from base to tip, indexed from 0 to 3 (the thumb does not have a base metacarpal bone and therefore contains a valid, zero length at that location).
4. More information: Details on Bone class.

2.2. Coordinates System

The Leap Motion system employs a right-handed Cartesian coordinate system. The origin is centered at the top of the Leap Motion Controller. The x- and z-axes lie in the horizontal plane, with the x-axis running parallel to the long edge of the device. The y-axis is vertical, with positive values increasing upwards (in contrast to the downward orientation of most computer graphics coordinate systems). The z-axis has positive values increasing toward the user. PAGE 3Figure 3. The Leap Motion right-handed coordinate system. Source: https://developer.leapmotion.com/documentation/

2.3. System Architecture

On Windows, the Leap Motion software runs as a service. The physical connection between the Leap Motion Controller and the computer takes place via USB bus. Leap-enabled applications access the Leap Motion service to receive motion tracking data. The Leap Motion SDK provides two different APIs to access the tracking data: a native interface and the WebSocket interface.

2.3.1. Native Application Interface

- Unordered List ItemProvided through dynamically loaded library, that connects to the Leap Motion service and provides the data to the application.
- In case of C# the library is not linked directly, but with use of the special wrapper for binding

PAGE 4Figure 4.Leap-enabled application. Source: https://developer.leapmotion.com/documentation/

2.3.2. WebSocket Interface

- Unordered List ItemService runs a WebSocket server on the localhost, port: 6437
- Tracking data in the form of JSON messages
- JavaScript client is available for convenience

- More information: Details

2.4. Setting Up a Project

NOTE: The Motion Leap libraries (file LeapCSharp.NETx.x.dll located in the lib folder of the Leap SDK package) are expected to be loaded from the same directory as the executable of the application, so it’s required that SDK is downloaded. The linking step will be covered later. PAGE 5Hello World C#: 1. Create new project of the WPF or Windows Forms type. 2. Add a reference to one of the pre-built libraries: 2.1. Select Project > Add Reference. 2.2. In the References dialog, select Browse. PAGE 62.3. Browse to Leap Motion SDK folder. 2.4. Choose the appropriate library for your project (depending on the version of .NET being used) PAGE 73. Set the target platform to x86 or x64 3.1. Select Project > Properties. 3.2. 3.3. 3.4. Select Build page. Set the Configuration to All Configurations. Under General, set Platform target to x86 or x64. PAGE 84. Edit the Post Build Event to copy the native libraries to the project’s target executable directory. 4.1. Important: If you are creating a 64-bit application, use the libraries in the lib/x64 directory, not the 32-bit libraries in the lib/x86 directory. 4.2. Open the project properties page. 4.3. Select the Build Events page. 4.4. Click Edit Post-build. 4.5. Edit the event field to copy the libraries: 4.5.1. For 32-bit projects add: xcopy /yr “$(LEAP_SDK)\lib\x86\LeapC.dll” “$(TargetDir)” 4.5.2. For 64-bit projects add: xcopy /yr “$(LEAP_SDK)\lib\x64\LeapC.dll” “$(TargetDir)” 5. Add your source code!

2.5. Sample source code

Sample source code can be found here:

leapexample.zip



Make sure that the reference to Leap Motion dll library is valid

2.6. Citations

All citations, used images and more information about Leap Motion can be found here:
https://developer.leapmotion.com/documentation/csharp/devguide/Sample_Tutorial.html
https://developer.leapmotion.com/documentation/python/devguide/Leap_Overview.html
https://developer.leapmotion.com/documentation/csharp/devguide/Leap_Architecture.html
https://developer.leapmotion.com/documentation/csharp/devguide/Leap_Coordinate_Mapping.html

From:
https://dokuwiki.zmitac.aei.polsl.pl/ - **RAU6 Dokuwiki**

Permanent link:
https://dokuwiki.zmitac.aei.polsl.pl/en:edu:zmitac:macro:lego:lab:csharp?rev=1516049488

Last update: **2018/01/15 21:51**