BodyBuilder session 4

Managing axes and intro to macros

Course Objectives

1. *How to visualize the axes of a segment in the BodyBuilder 3D workspace*
    - *Virtual point review*
    - *Adding the appropriate code to the .MOD and .MKR files*

2. *The Local to Global coordinate system switch*
    - *Introcudtion*
    - *Why switch from Local to Global and vice versa?*

2. *Macros*

How to visualize axes of a segment in the workspace

Review – Virtual Points

Any location in space can be specified as a virtual point.

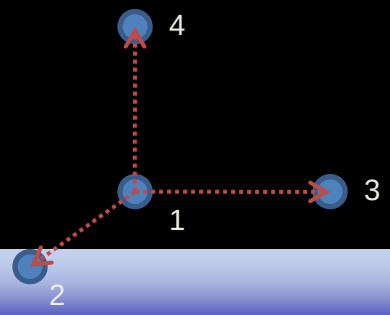
1. Here are some Examples of how to create a virtual point:
    - Example: VirtualMkr = (MKR1+MKR2)/2
        - Specifies the midpoint between markers: MKR1 and MKR2
    - Example: VirtualMkr = SegmentOrigin + 100*Segment(1)
        - Begins at the origin point of a segment (“SegmentOrigin”) and creates a marker 100 mm from that point in the direction of the first axes (X) of the Segment.
2. Any virtual point can then be OUTPUT to the .c3d file and viewed in the workspace
    - OUPUT(VirtualMkr)
3. NOTE – By outputting a Virtual marker, you will be able to view it in the BodyBuilder workspace but you will not be able to graph it or visualize sticks between it and any other marker until it is added to the .MKR file.

Visualizing Axes – Using Virtual Points in .MOD file

It is a good general practice to visualize the axes you create with the Segment definitions to ensure they have been created as intended.

This is done by creating four virtual points

1. The first virtual point will be created at the Origin of the segment
2. The second virtual point will be created at a distance x from the segment origin along the direction of the segment's X axis
3. The third virtual point will be created at a distance y from the segment origin along the direction of the segment's Y axis
4. The fourth virtual point will be created at a distance z from the segment origin along the direction of the segment's Z axis



Visualizing Axes – Using Virtual Points in .MOD file

```
Segment=[Origin, A-B, C-B, xyz]
ORIGINSegment=Origin
AXISXSegment= ORIGINSegment +(1(Segment)*100)
AXISYSegment= ORIGINSegment +(2(Segment)*100)
AXISZSegment= ORIGINSegment +(3(Segment)*100)
OUTPUT(ORIGINSegment,AXISXSegment,AXISYSegment,AXISZSegment)
```

The above translates into:

ORIGINSegment is a virtual point coincident with the point Origin, used in the segment definition

The point Origin could be a real marker or a previously created virtual point

AXISXSegment is a virtual point placed at 100mm along the direction of the first axis of the segment (X axis) from the ORIGINSegment point

*NOTE - the first axis of a segment is always the X axis no matter what order is chosen in the segment definition.

AXISYSegment is a virtual point placed at 100mm along the direction of the second axis of the segment (Y axis) from the ORIGINSegment point

AXISZSegment is a virtual point placed at 100mm along the direction of the third axis of the segment (Z axis) from the ORIGINSegment point

Visualizing Axes – Adding points to .MKR file

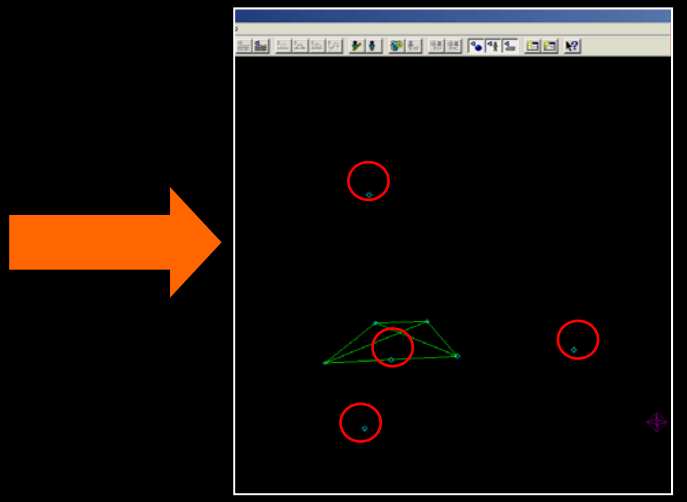
Below is an example of how to run axes visualization code for a specific segment:

```
SACR = (LPSI + RPSI)/2
OUTPUT (SACR)

PelvisO = (LASI +RASI)/2
OUTPUT (PelvisO)

Pelvis = [PelvisO, LASI - RASI, PelvisO - SACR, yzx]

                ORIGINPelvis=PelvisO
                AXISXPelvis= ORIGINPelvis +(1(Pelvis)*100)
                AXISYPelvis= ORIGINPelvis +(2(Pelvis)*100)
                AXISZPelvis= ORIGINPelvis +(3(Pelvis)*100)
                OUTPUT(ORIGINPelvis,AXISXPelvis,AXISYPelvis,AXISZPelvis)
```



Once the code fragment above gets executed, the newly defined virtual points are OUTPUT to the .C3D file, and shown in the BodyBuilder 3D workspace. They appear as single markers, not connected together:

Visualizing Axes – Adding points to .MKR file

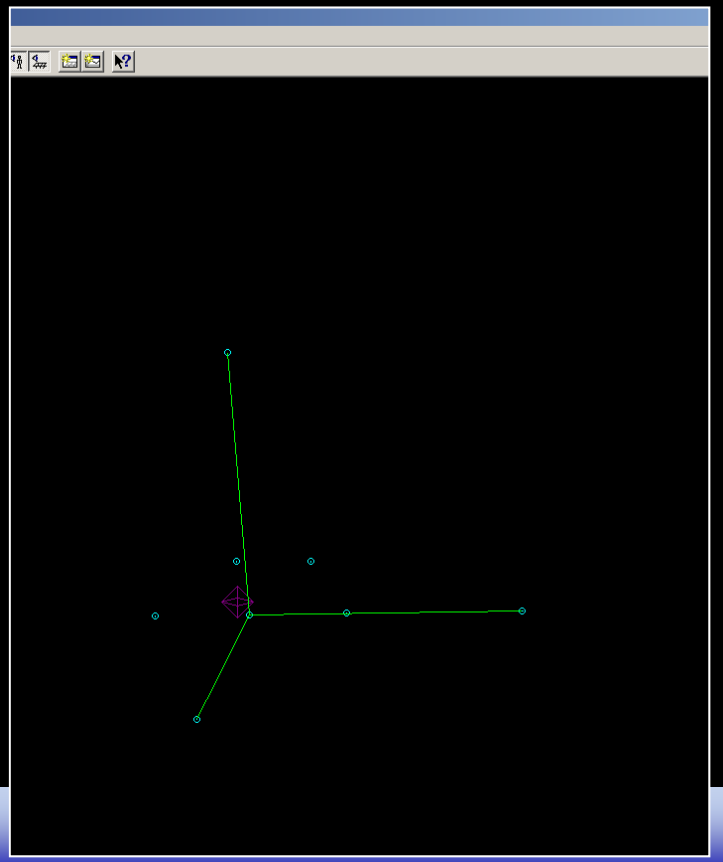
To see the markers being connected together, we have to update the .MKR file as follows:

1. Declare the points in the section we want them to appear
2. Connect the points using the comma (‘,’) operator

Reassign the modified .MKR to the subject using the ‘Subject Settings’ dialogue

[Segment Visualisation]
ORIGINPelvis
AXISXPelvis
AXISYPelvis
AXISZPelvis

ORIGINPelvis,AXISXPelvis
ORIGINPelvis,AXISYPelvis
ORIGINPelvis,AXISZPelvis



Visualizing Axes – Example

Lets go to BodyBuilder and practice adding this code to our example model and see what it looks like.

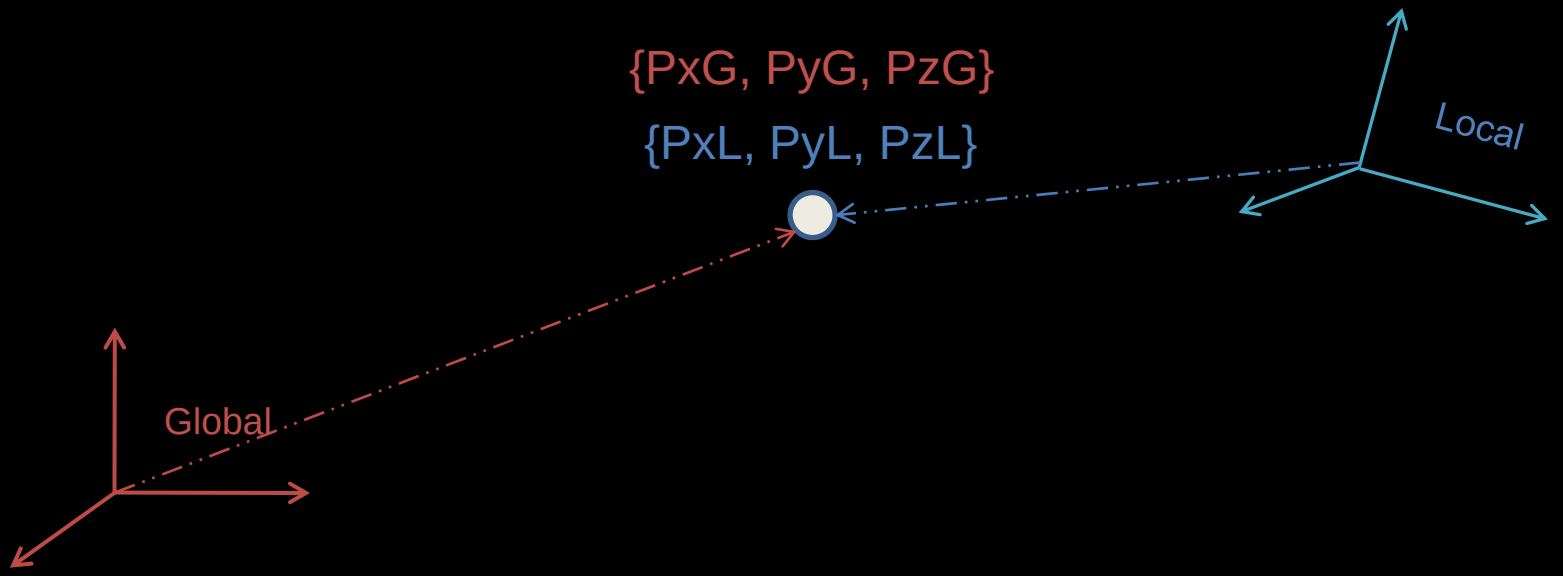
Axes switch from Local to global coordinate systems

What is a Global to Local switch?

Global point – A point that is expressed in the global Vicon coordinate system, specified by the position of the calibration object during the static step of the system calibration procedure

Local Point – A point that is expressed in a local segment coordinate system, specified by the segment definition in the .MOD file.

Note - Any point can be defined by the global or any local coordinate system



Why switch from Local to Global and vice versa?

EXAMPLE

Markers placed on prominent bony landmarks (i.e. lateral femoral condyle) suffer from skin motion artefacts.

Some people prefer to track a body segment using tracking markers placed in any position on the segment. This implies that the most relevant anatomical points need to be 'calibrated' with respect to the tracking markers during the static trial.

Global To Local:

1. Create a technical coordinate system rigidly associated with the segment using the tracking markers
2. **Calculate the local coordinates of the marker on the anatomical point with respect to the technical coordinate system** defined at step #1.
3. Store the local coordinates in the .MP file

In the same way, once the marker placed on the anatomical points gets removed, its global position needs to be reconstructed starting from the position of the tracking markers.

Local To Global:

4. Create a technical coordinate system rigidly associated with the segment using the tracking markers (identical segment definition as in Step #1.)
5. **Calculate the global position of the point** whose local coordinates are stored in the .MP file

Why switch from Local to Global and vice versa?

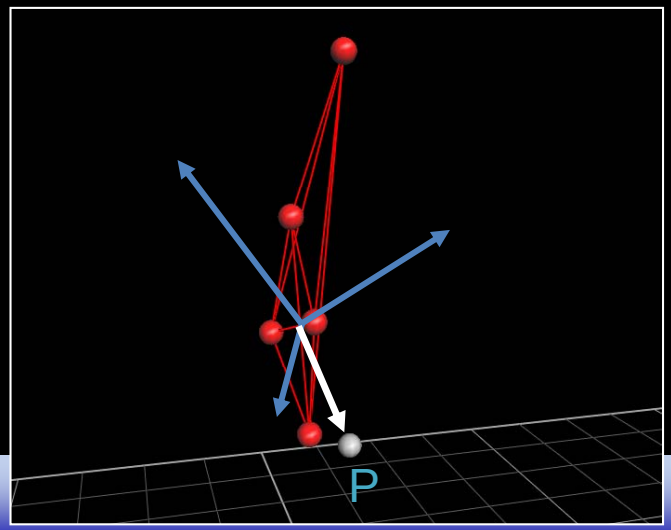
Global to Local

- The Global to Local transformation comes handy when we need to know the position of a marker with respect to a ‘technical’ coordinate system rigidly associated to a body segment

Example

A Medial Epicondyle marker is placed on the elbow during the static trial and then it gets removed for dynamic trials (medial markers do not have a good visibility and are easily knocked off during motion).

Before removing the marker we need to know its local coordinates with respect to a coordinate system defined using other markers attached to the same segment of the medial epicondyle marker (due to the Rigid Body Hypothesis, it is assumed that all the markers attached to the same segment do not move relatively to each other). The local coordinates of the marker are then saved to the .MP file

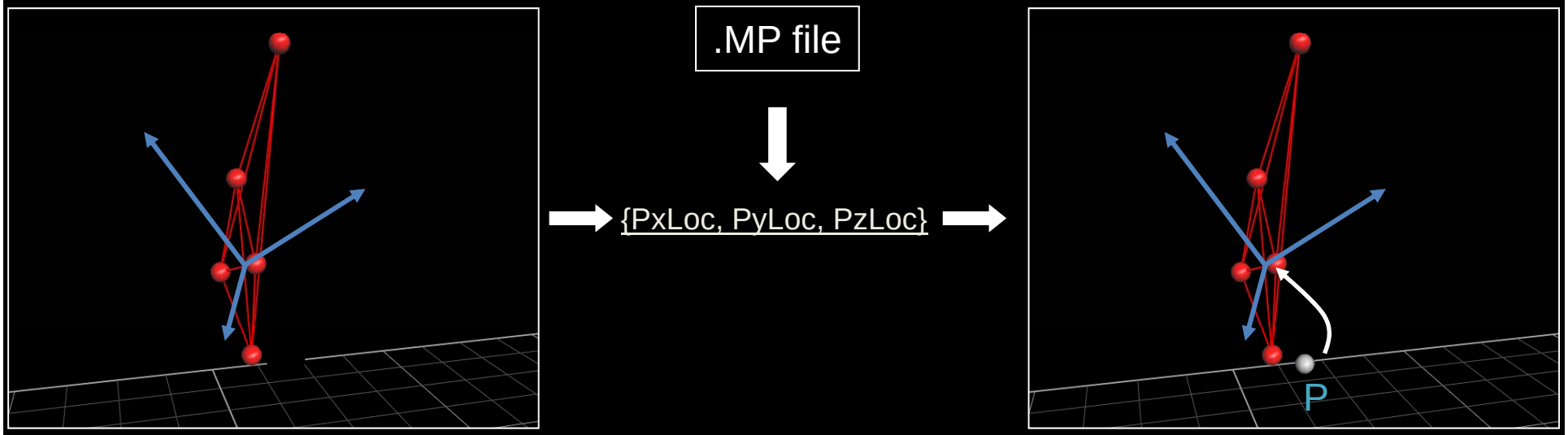


{PxLoc, PyLoc, PzLoc} → .MP file

Why switch from Local to Global and vice versa?

Local to Global

The Local to Global transformation is needed to recreate the global position of a marker previously ‘calibrated’ with respect to a technical coordinate system during the static trial and successively removed for dynamic trials. The local coordinates of the marker are retrieved from the .MP file



Global/Local switch syntax

- Converting FROM %LocalPoint in SegmentA, TO GlobalPoint
  - Use of operator '*'
  - *GlobalPoint = %LocalPoint*SegmentA*
- Converting FROM GlobalPoint, TO %LocalPoint in SegmentA
  - Use of operator '/'
  - *%LocalPoint = GlobalPoint / SegmentA*

NOTE – the symbols "" and "/" do not mean multiply and divide when used this way*

NOTE – The symbol "%" does not change the meaning of the expression. It is only used to flag a parameter as being a value in a local reference frame

Macros

What are Macros?

In order to make scripts shorter, easier to read, and more reliable, BodyLanguage supports the definition and calling of macros.

A macro works like a subroutine in a compiled program. Once defined, a macro can be called as many times as required, thereby eliminating repetitions of text.

In order to make a macro applicable in different situations, it can be defined using a set of parameters. These parameters are listed at the start of the macro definition. When the macro is executed, the parameters are replaced by a matching set of real variable names.

How to use a Macro

1. Define the Macro at the beginning of your script
   - Signal the start of a macro with the statement “DEFINE MACRO” or “MACRO”
   - Next to this statement, list the name of the macro and then in parentheses, any parameter used in the macro code.

2. Below, create the calculations to be run when the macro is called, using the parameters.

3. Close the macro definition with the statement ENDMACRO

```
MACRO mac_name(D1, D2, D3)
        <code using variables D1, D2 and D3>
        <code using variables D1, D2 and D3>
        <code using variables D1, D2 and D3>
ENDMACRO
```

in Macros – The Text Concatenation Operator

The “#” symbol is known as the ‘Text Concatenation Operator’ and is used to create output variable names in the macro definitions

```
Macro AXISVISUALISATION(Segment)
        ORIGIN#Segment = 0(Segment)
        AXISX#Segment = {100,0,0}*Segment
        AXISY#Segment = {0,100,0}*Segment
        AXISZ#Segment = {0,0,100}*Segment
OUTPUT (ORIGIN#Segment,AXISX#Segment,AXISY#Segment,AXISZ#Segment)
Endmacro
```

When the macro above is invoked in the code using the following instruction:
AXISVISUALISATION(head)

the points created will have the following names:

- ORIGINhead
- AXISXhead
- AXISYhead
- AXISZhead

Note:
ORIGINhead comes from ORIGIN#Segment. The ‘Segment’ part gets then replaced with the real name of the input parameter of the macro, i.e. ‘head’
The same applies to the other output variables

The Replace Macro

```
{*Start of macro section*}
{*====================*}
Macro REPLACE4(p1,p2,p3,p4)
{*Replaces any point missing from set of four fixed in a segment*}
s234 = [p3,p2-p3,p3-p4]          {*Defines a segment s234 using all points except p1*}
p1V = Average(p1/s234)*s234      {*Finds the average position of p1 in the s234 local
Co-ord system and creates virtual point p1V from this reference system*}
s341 = [p4,p3-p4,p4-p1]          {*Defines a segment s341 using all points except p2*}
p2V = Average(p2/s341)*s341      {*Finds the average position of p2 in the s341 local
Co-ord system and creates virtual point p2V from this reference system*}
s412 = [p1,p4-p1,p1-p2]          {*Defines a segment s412 using all points except p3*}
p3V = Average(p3/s412)*s412      {*Finds the average position of p3 in the s412 local
Co-ord system and creates virtual point p3V from this reference system*}
s123 = [p2,p1-p2,p2-p3]          {*Defines a segment s123 using all points except p4*}
p4V = Average(p4/s123)*s123      {*Finds the average position of p4 in the s123 local
Co-ord system and creates virtual point p4V from this reference system*}
{* Now only replaces if original is missing  11-99 *}
p1 = p1 ? p1V
p2 = p2 ? p2V
p3 = p3 ? p3V
p4 = p4 ? p4V
endmacro
```

Lets call this macro for the Pelvis which is a rigid segment with 4 points

REPLACE4(LASI, RASI, LPSI, RPSI)
OUTPUT(LASI,RASI,LPSI,RPSI)

I use the output command to make sure I see the filled in points in the .C3D file

Continuing our BodyBuilder example code

Lets go to BodyBuilder and add these macros and see how this grows our code

Assignment 4

1. Open your training.mod file

2. Add code to visualize all of the segment axes you created in Assignment 2.
    1. First try to put the code in for one of the segments without using the macro
    2. For the rest of the segments, use the Axes Visualization macro

3. Add the Replace4 macro to your code.

4. Open the example walking trial. Create a number of gaps in the pelvis markers using the Edit | Delete Points tool

5. Run the model to allow the Replace4 to fill in all of the gaps in the Pelvis markers.