BodyBuilder session 3

Introduction to Kinematics

Course Objectives

1. *Kinematic Analysis*
    - *Vector Algebra review*
    - *An example Plug-In Gait Pelvis Segment Definition*
2. *BodyLanguage Basics*
    - *Basic syntax*
    - *Segment definition syntax*
        - *Example - Script to create Plug-In Gait Pelvis segment*
3. *Review Non Assigning Statements*

Kinematic Analysis

Introduction to Kinematics and Rigid Bodies

Kinematics – Describing the motion of objects without considering the cause of the motion (forces). In BodyLanguage, these objects are referred to as segments.

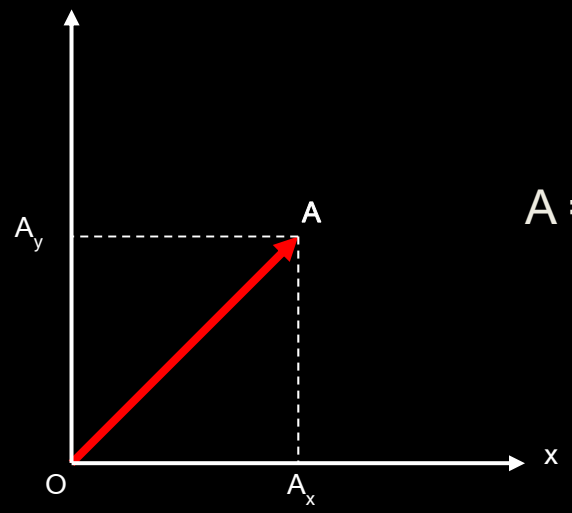
Segment – A portion of the subject assumed to be a rigid body

- Each segment has a position and orientation that can be defined by 3 markers on that segment
  - Using vector algebra, an origin and 3 orthogonal axes can be specified by the markers. This coordinate system defines the position and orientation of the segment.

Vector Algebra Review

Vector – A mathmatical entity described by Direction, Orientation and Magnitude.

- In 3D space, the vector is made up of 3 coordinates (x,y,z)
- i.e. – Marker position, joint forces or moments



$A = (A_x, A_y, 0)$

Scalar - A singular number

- i.e. – Anthropometric parameters, powers

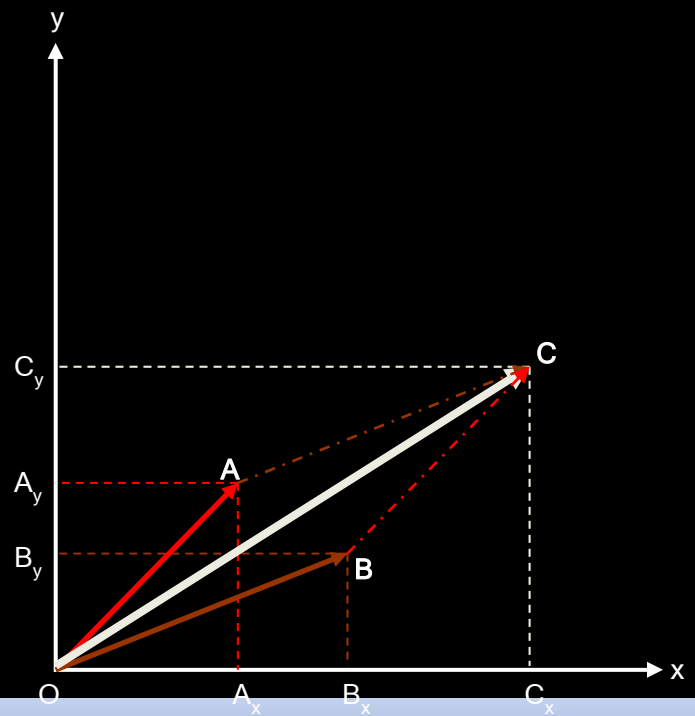
Vector Algebra Review

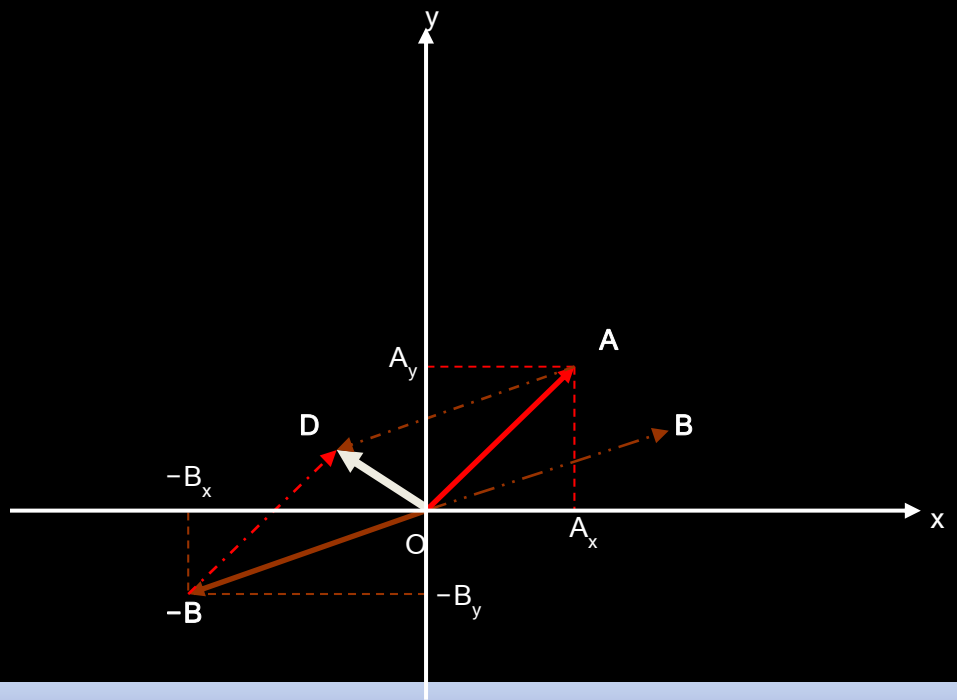
Vectors can be added and subtracted by adding or subtracting the components of the individual vector

- By subtracting one marker position from another, a vector is created between the markers

$$\vec{C} = \vec{A} + \vec{B} = (A_x + B_x, A_y + B_y)$$

$$\vec{D} = \vec{A} - \vec{B} = \vec{A} + (-\vec{B}) = (A_x - B_x, A_y - B_y)$$





Cross Product – of two vectors yields a vector that is orthogonal to the two vectors that were crossed

- i.e. - Segments are defined using Cross Products
- Right hand rule is applies

Dot Product - of two vectors yields one number

Dot Product:

$$\mathbf{A}\cdot\mathbf{B} = Ax\,Bx + Ay\,By + Az\,Bz$$

Result is a Scalar

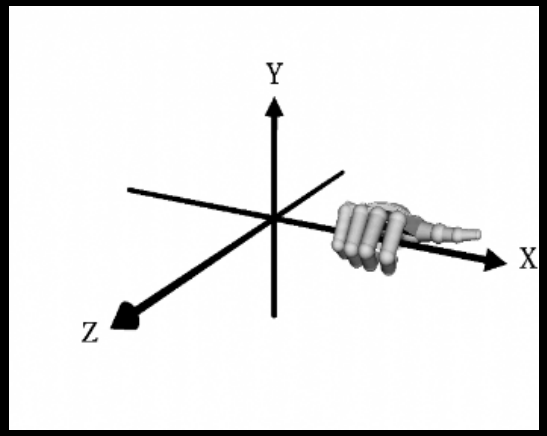
Cross Product:

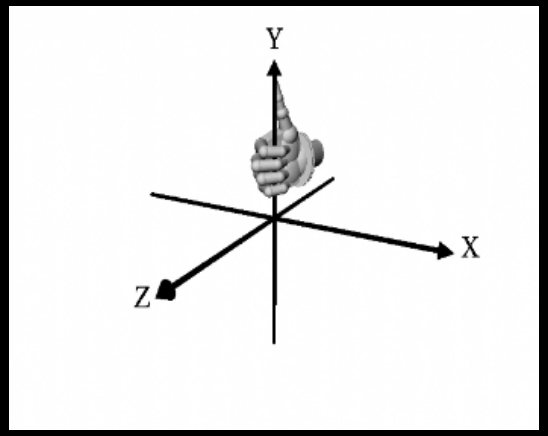
$$\mathbf{A} \times \mathbf{B} = \begin{vmatrix} i & j & k \\ Ax & Ay & Az \\ Bx & By & Bz \end{vmatrix}$$

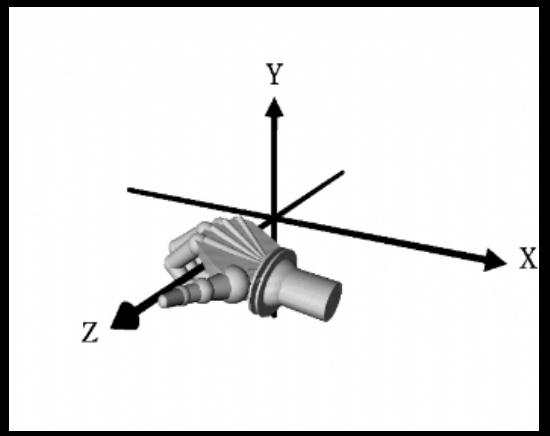
A x B = AyBz-ByAz ; AzBx-BzAx ; AxBy-BxAy

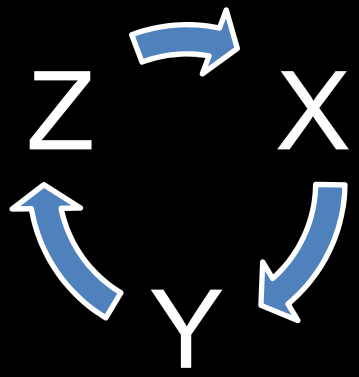
Result is a Vector orthogonal to two original vectors

Segment considerations – Right Hand Rule









Segment considerations

- A segment exists in every frame where the points defining it also exist
  - If a marker used to define a segment has a gap, then the segment and all calculations based on that segment will also have a gap

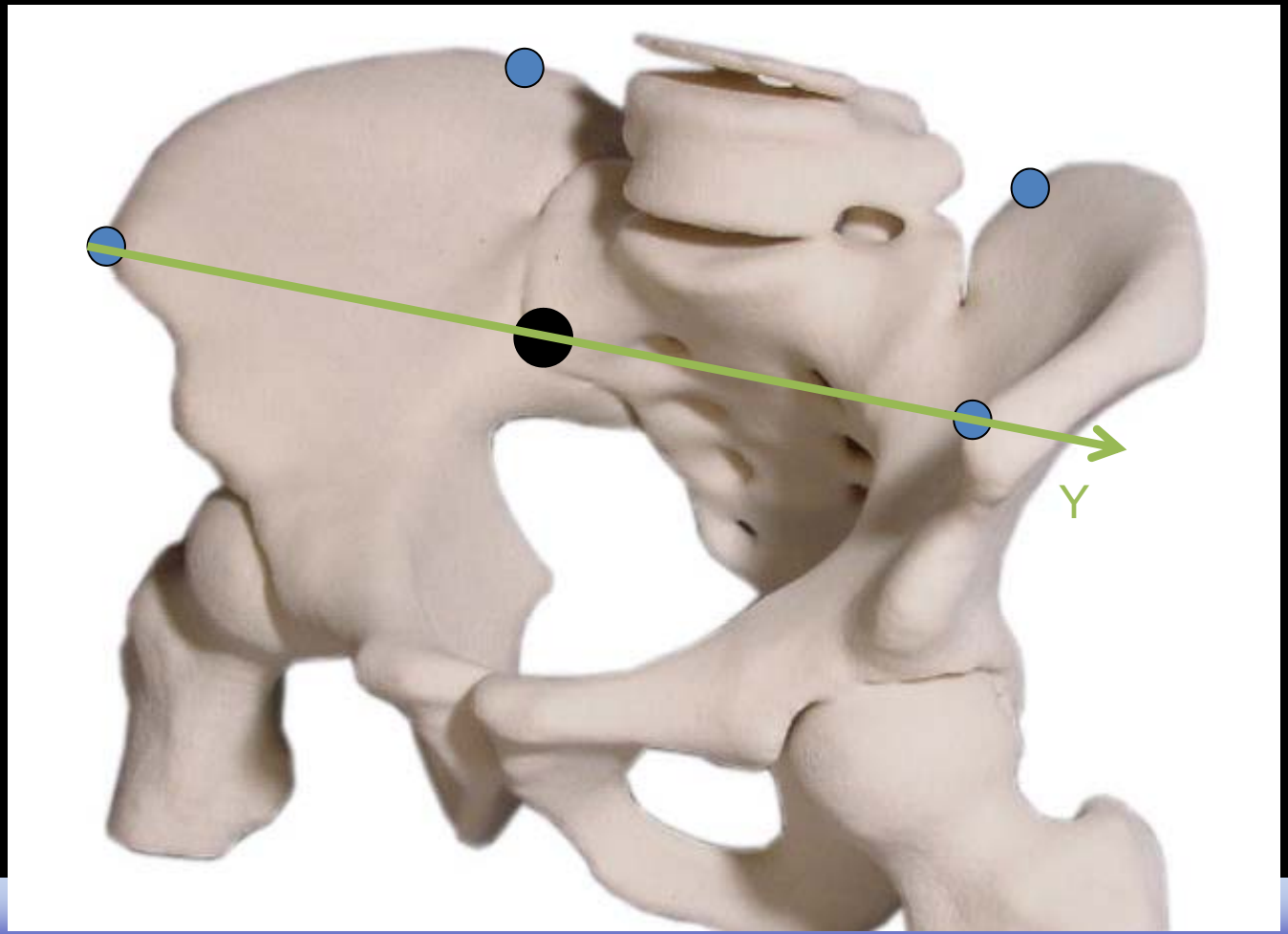
ISB Standards you may wish to follow:

- Segment Origin is the center of rotation of the distal joint
- 1st axes coincides with the line between the proximal and distal joints
- Second axes coincides with the main distal joint axes of rotation

Note – BodyBuilder will not force you to follow and ISB standards. You can set the segments up as you wish.

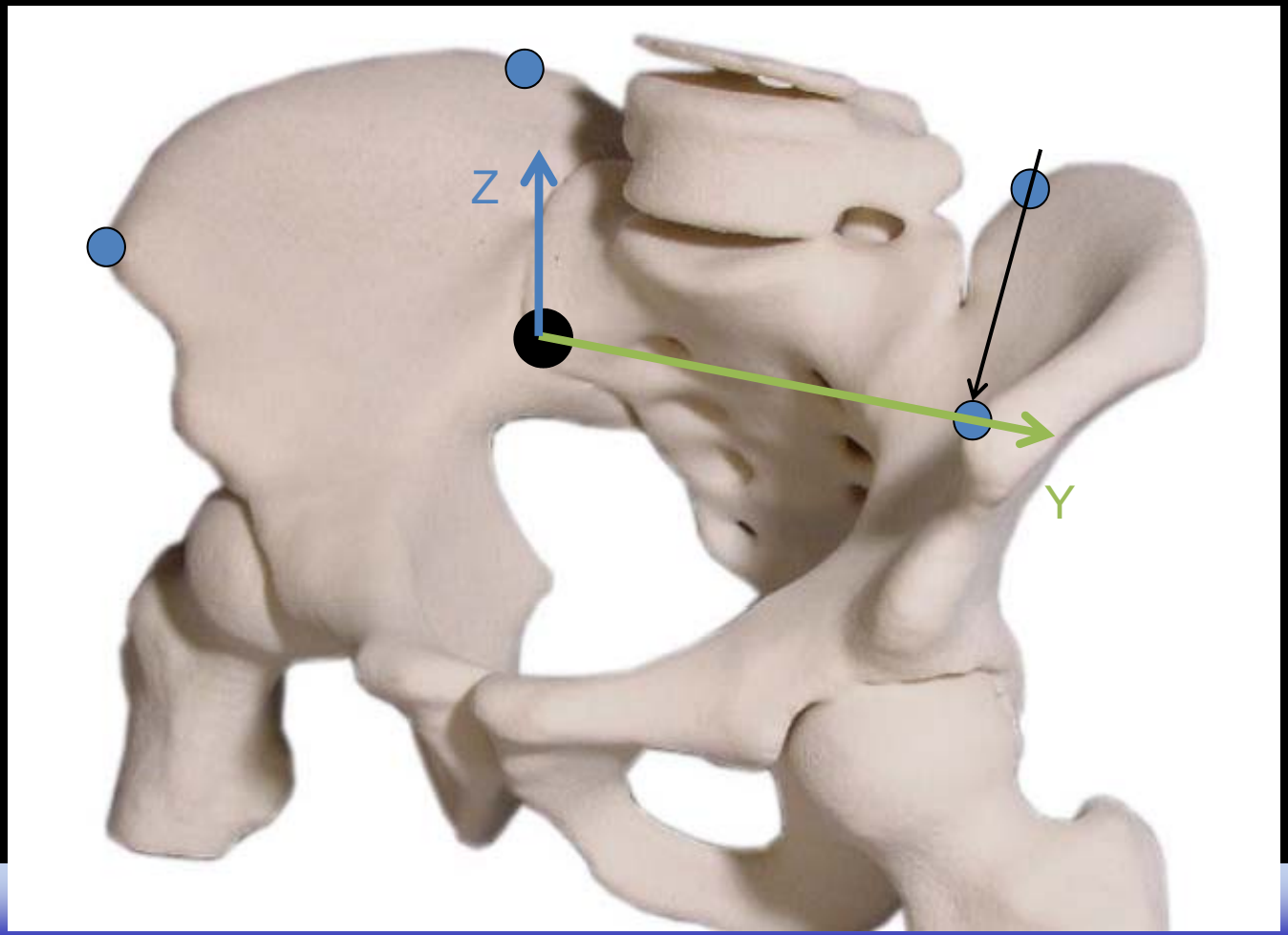
PIG Pelvis example

- Decide to place the Origin in the middle of the two ASI markers
  - Origin = (RASI + LASI)/2
- The first vector (Y axis) will be in the direction from RASI to LASI
  - Y axis = LASI – RASI
  - This axis will begin at the origin of the segment



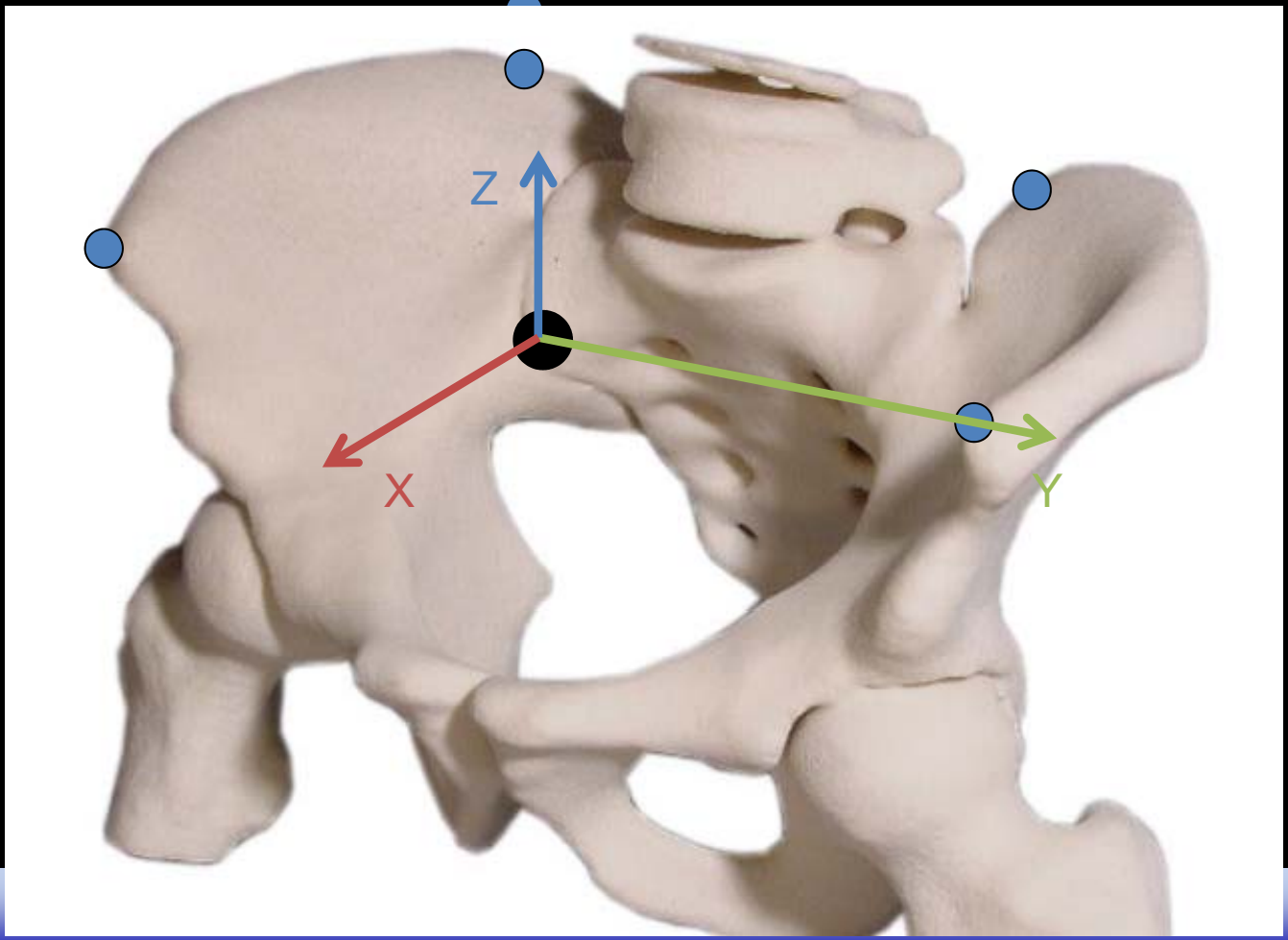
PIG Pelvis example

- Create a second vector from the LPSI to the LASI
  - vector2 = LPSI – LASI
- Take the cross product of the Y Axis and the vector2 to define the orthogonal Z axis
  - Z Axis = (vector2) x (Y axis) ** order matters**
  - This axis will begin at the origin of the segment



PIG Pelvis example

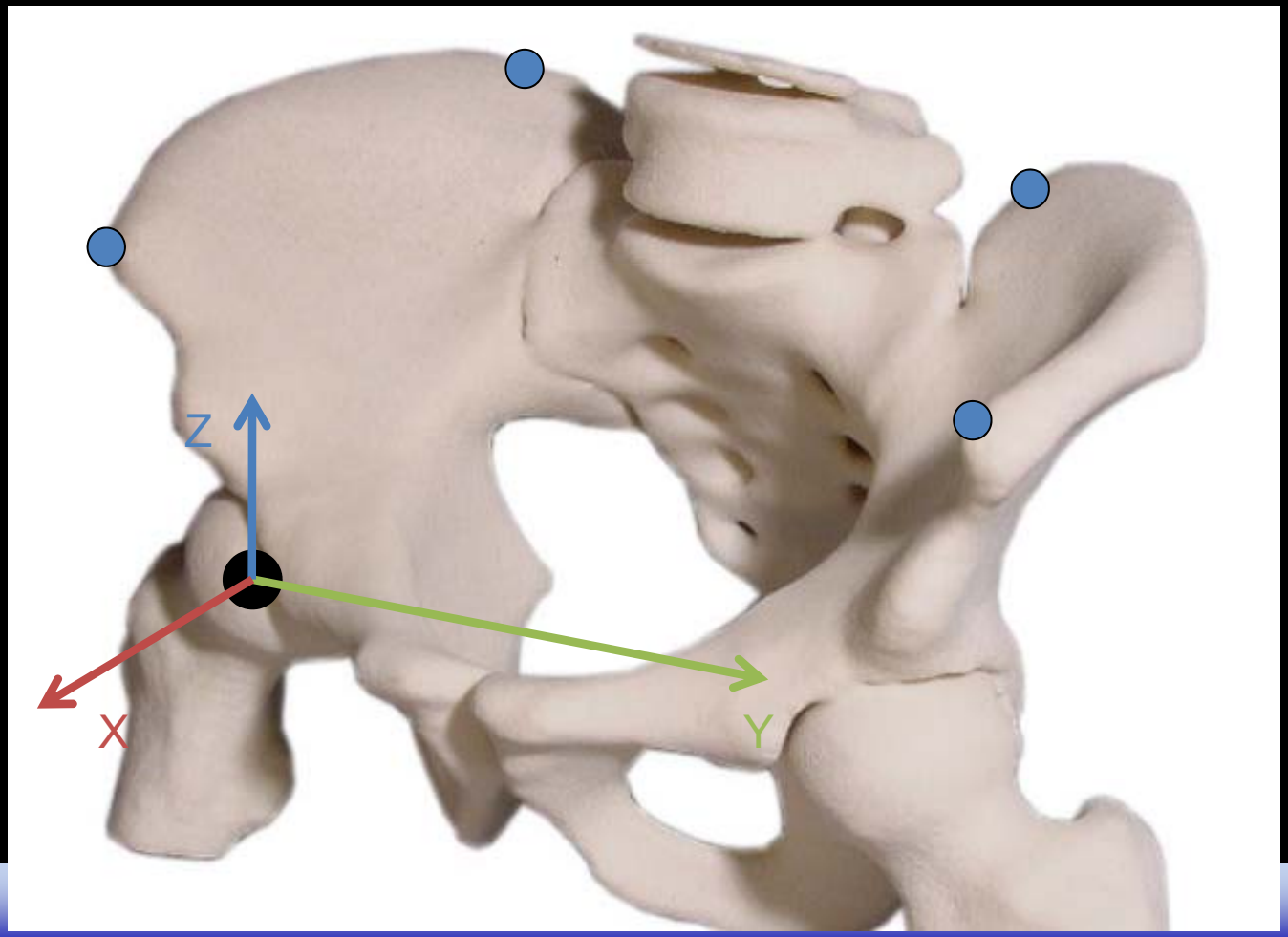
- Now take the cross product of the Y axis and the Z axis to define the orthogonal X axis
  - X Axis = (Y axis) x (Z axis)
  - This axis will begin at the origin of the segment



PIG Pelvis example

Finally, we may want to move the coordinate system (without any rotation) to the center of rotation of the distal joint of the segment. In this case, the Hip Joint Center.

We will talk more about this in an upcoming session on advanced segment topics



BodyLanguage Basics

Review of BodyBuilder file types

We are ready to begin the creation of our .MOD file!

.MP File – Model Parameters

- Constants used in calculations (Anthropometric measurements)
- These can be entered by user or calculated by in the Body Language code

.MKR File – Marker File

- A list of marker names used in the model
- In Workstation this file is also used for Autolabeling. This section is replaced by the .vst file in Nexus.

.MOD File – Model Code

- **This file contains the Body Language code the user writes**

.C3D File – Processed Data

- This file is originally created in Nexus (Workstation) to hold processed marker data
- When a Body Language file is run on the data, the calculations are appended to the .C3D file

Basic Syntax

Comments – You can write text for reference that is not part of the calculations
- Text must be surrounded by the symbols: {* ….*}

Calling markers – You can call the position of any marker by using the name of the marker specified in the .MKR file
- When you type the name of a marker, you are calling all 3 coordinates (x,y,z)
- To call just one coordinate of any vector (including a marker) use:
  - 1(marker_name) or marker_name(1) to call the first coordinate and so on…

Calling a Model Parameter – You can call any constant from the .MP file by typing that name of that constant value exactly as it appears in the .MP file

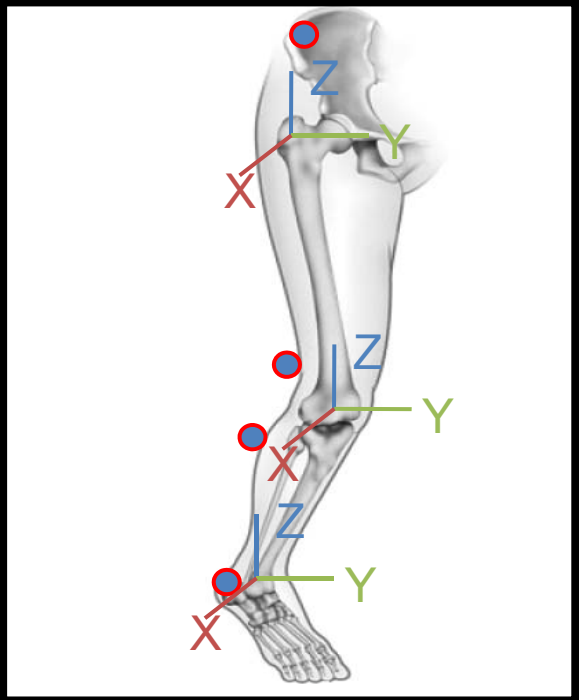
Operations – Standard operators and order of operations apply
- +, -, /, … add, subtract, divide, etc… repsectively
- (3+9)/4 = 3 parenthesis > multiply/divide > Add/subtract

Example:

PelvisO = (LASI+RASI)/2

A new point called, PelvisO is calculated as the mid point between the real markers, LASI and RASI

Recall your rigid body diagram…



We need to tell BodyBuilder how to use the markers on the subject to create segment definitions.

Segment – section of the body, assumed to be rigid, with both a position and orientation

- At least three markers are needed to define the Location and orientation of a segment

We can both define a segment and create a coordinate system for the segment with one line of code in BodyBuilder!

<u>BodyLanguage Expression</u>

Segment = [OriginPoint, (PointB-PointA), (PointD-PointA), Tolken]

Segment definition BodyLanguage syntax

Segment = [OriginPoint, (PointB-PointA), (PointD-PointA)]

"Segment" – name of the segment

"Origin Point" – Marker representing the location of the origin of the local coordinate system.

"(PointB – PointA)" – First defining line is a vector from point A to point B

- parallel or coincident with the first axis (SA1) on the local coordinate system, through Origin

"(PointD-PoingA)" – Second defining line is a vector from Point A to point D

- Cross product between second and first defining lines established the second axis (SA2), through the Origin
- Order of cross product must me (DL2 x DL1) and Right Hand rule must apply

•The first and second axes are then crossed to establish the remaining third axis

- Order of cross product must me (SA1 x SA2) and Right Hand rule must apply

NOTES

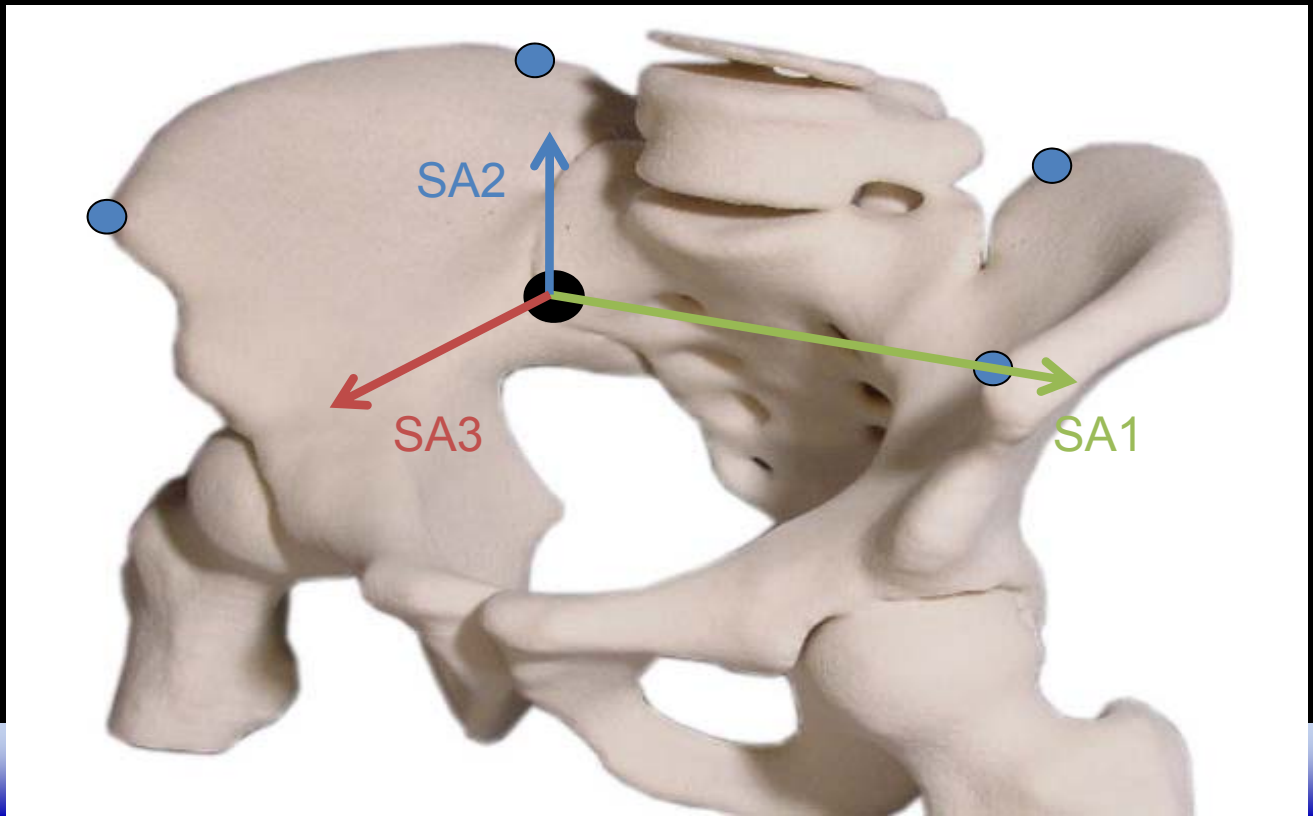
- Must use […] around the segment definition
- Any of the points can be "virtual points" – points in space you define based on actual markers
- The cross products are taken as part of the segment definition. You need take no special steps

PIG Pelvis example

BodyLanguage in .MOD file to define this segment:

Segment = [OriginPoint, (PointB-PointA), (PointD-PointA)]

PelvisO = (LASI+RASI)/2
Pelvis = [PelvisO, LASI-RASI, LASI-LPSI]



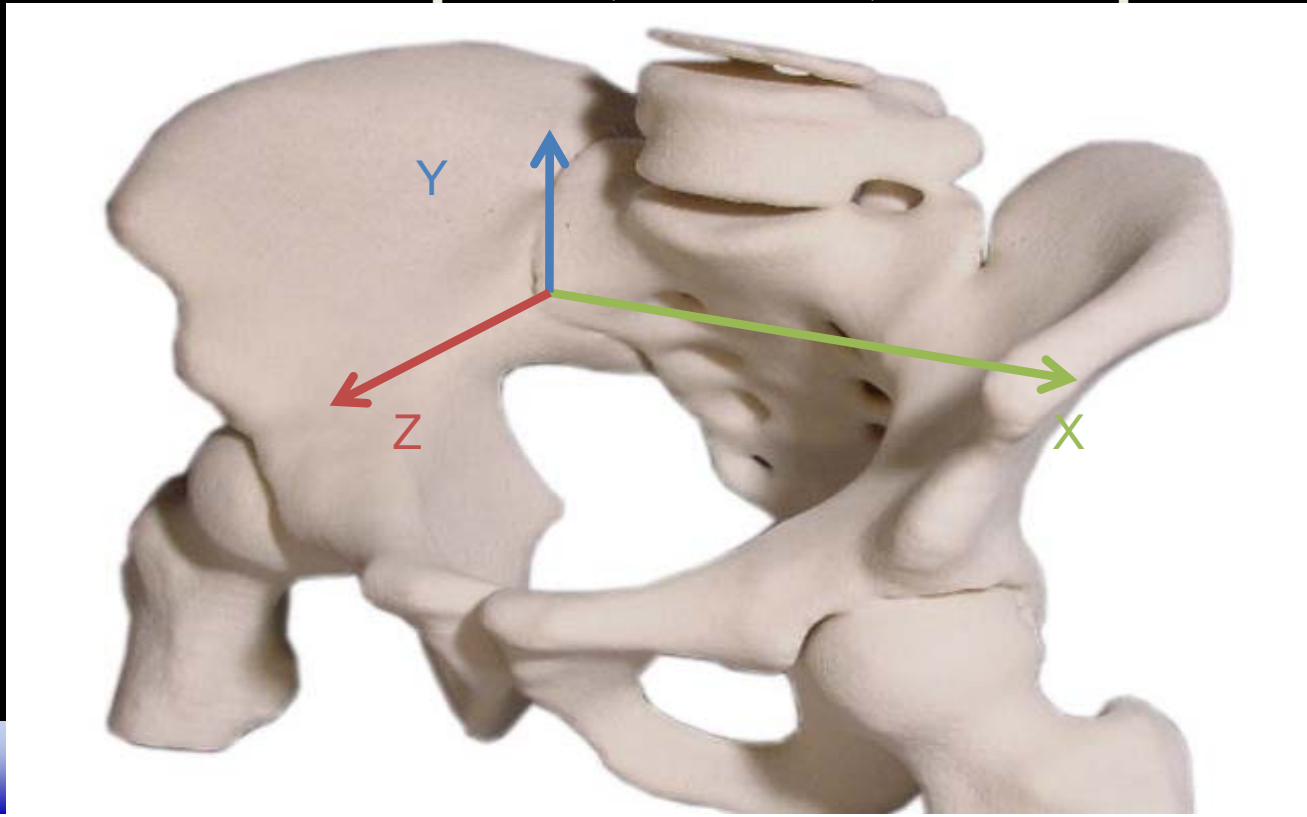
Segment definition BodyLanguage syntax – Adding a Token

Segment = [OriginPoint, (PointB-PointA), (PointD-PointA), Token]

A Token specifies the order of the Axes. If no token is specified, the axes order is defaulted to be:

X – First defining line
Y – Orthogonal to First and Second defining lines
Z – Orthogonal to X and Y axes.

Pelvis = [PelvisO, LASI-RASI, LASI-LPSI]



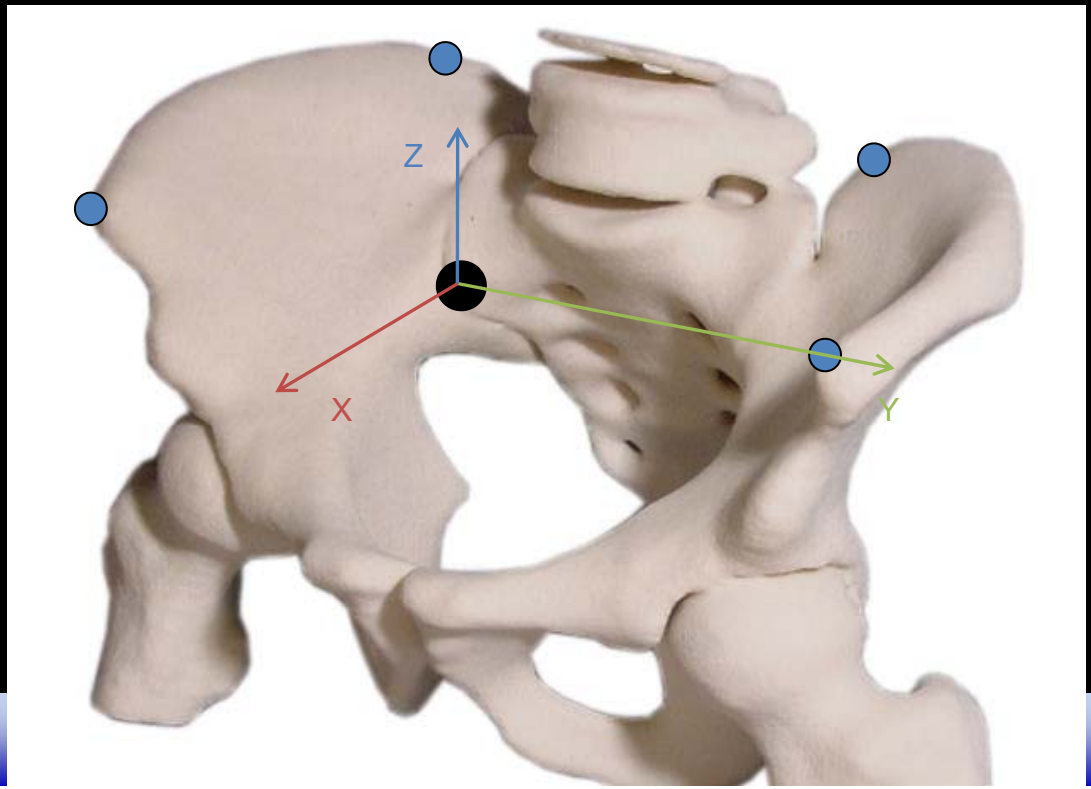
Segment definition BodyLanguage syntax – Adding a Token

You can change the order of the axes that are created. However, Right-Hand rule must be obeyed. For example:

Segment = [OriginPoint, (PointB-PointA), (PointD-PointA), yzx]

Y – First defining line
Z – Orthogonal to First and Second defining lines
X – Orthogonal to X and Y axes.

Pelvis = [PelvisO, LASI-RASI, LASI-LPSI, YZX]



Non Assigning Statements

What are Non Assigning Statements?

Non-assigning statements are used in BodyLanguage for three functions. Each function has a Keyword.

Function	BodyLanguage Keyword
generate data points in the 3D workspace and .C3D file	OUTPUT
write parameters to the model parameter (.MP) file	PARAM
flag the presence of objects as being optional	OptionalPoint OptionalOptional…

Non Assigning statement - OUPUT

Once a variable has been calculated, you may want to output it to the .C3D file to allow you to view the results.

Back to our Pelvis example:
PelvisO = (LASI+RASI)/2
OUTPUT(PelvisO)

I have calculated a new point called PelvisO. I can refer to that point in future equations. However, if I want to view that point in a workspace or graph that point, I need to output it to my .C3D file with the command:
PelvisO = (LASI+RASI)/2

The same is true for all calculated Kinematic values. Later, when we discuss how to calculation angles, you will need to OUPUT those variables to the .C3D file to be able to view/export them for any analysis

Non Assigning statement - PARAM

Many calculated values are intended to be used as a scalar in future calculations. You will want to write these values to the .MP file using the PARAM command.

An example: determine the knee width as the distance between a medial knee marker (MKNE) and a lateral knee marker (KNE)

KNEwidth = dist(RMKNE – RKNE)

When a .MOD containing this line of code is run on a trial, this equation will be run on every frame in the trial. We are actually looking for one value.

Add the line:
PARAM(KNEwidth)

The PARAM command automatically takes the average of a variable and exports this scalar value into the .MP file. From here, this value can be called on later in the code.

Non Assigning statement – Optional Values

By default, if a line of code references an object, such as a marker, that has not been previously defined, that line is assumed erroneous and flagged.

The exception is an “OptionalPoint”.

You can use OptionalPoints to set up conditional expressions that will run if the point is present in the trial but will not cause the model to fail if the point is not present.

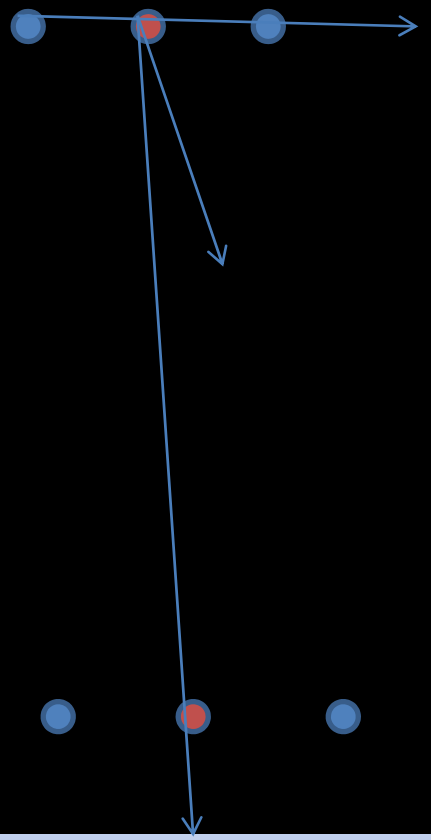
You specify optional objects with one of the following commands where multiple objects of the specified type can be listed in the parenthesis, separated by a “ , “ :

- *OptionalNumbers()* - Scalar values
- *OptionalForces()*
- *OptionalMoments()*
- *OptionalPowers()*
- *OptionalReactions()*
- *OptionalPoint*

Review

Lets look at some example BodyBuilder code that includes the syntax covered today:

```
{*Humerus Segments*}
{*================*}
ElbowOS = ($MarkerDiameter+$Elbowwidth)/2
REJC = CHORD(ElbowOS,RELB,RSJC,RSJC-500*Thorax(2))

{*Bar across back of wrist; A towards thumb*}
RWRI = (RWRA+RWRB)/2
RRadius = [RWRI,RWRA-RWRB,REJC-RWRI,xyz]
RWJC = RWRI+$WristThickness*2(RRadius)

RHumerus = [REJC,RSJC-REJC,REJC-RWJC,zyx]
AXISVISUALISATION (RHumerus)

{*Radius (and Ulnar) Segments*}
{*============================*}
{*RRadius = [RWJC,RWRB-RWRA,RWJC-REJC,yxz]*}
RRadius = [RWJC,REJC-RWJC,REJC-RSJC,zyx]
AXISVISUALISATION(RRadius)

{*Wrist Segments (dummy)*}
{*======================*}
{*Bar across back of wrist; A towards thumb*}
RWrist = [RWJC,REJC-RWJC,RWRA-RWRB,zxy]

{*Hand Segments*}
{*=============*}
HandOS = ($MarkerDiameter + $HandThickness)/2
RHND = RFIN - (HandOS*RWrist(1))
RHand = [RWJC,RRadius(2),RHND-RWJC,yxz]

OUTPUT(RHND)
AXISVISUALISATION(RHand)
```



Assignment 3

1. Open the training.mod file we started in the last class.
2. Create a pelvis segment
    1. Origin will be at a point midway between the LASI and RASI
    2. Z axis will point up
    3. X axis will point anterior
    4. Y axis will point towards the subject's left
3. Create Left and Right thigh segments using just the thigh markers
    1. The origins will be at the lateral KNE markers
    2. Z axis will point up from the KNE to the ASI
    3. X axis will point anterior (roughly)
    4. Y axis will point towards the subject's left (roughly)
4. Create Left and Right shank segments using just the shank markers
    1. The origins will be at the later ANK markers
    2. Z axis will point up from the ANK to the KNE
    3. X axis will point anterior (roughly)
    4. Y axis will point towards the subject's left (roughly)

* Next class we will learn how to transfer these segment definitions to anatomical joint centers. We will also learn how to view the axes we create here to check that they are correct. For now, just practice thinking through the segment definitions and give it your best shot.