Event Between

Overview

The **Event_Between** pipeline command creates a new event label based on the time interval between two existing events in the C3D File (i.e. between the first and last events). This command is useful when analyzing motion data that requires identifying specific time points between known events.

This function allows users to:

- Define a new event occurring between two existing events.
- Specify whether the new event should be placed at the Start, End, or Midpoint of the interval.

Pipeline Command

```
Event_Between
/NEW_EVENT_NAME=
! /RANGE_INSTANCE=0
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /FRAME_OFFSET=0
! /TIME_OFFSET=
! /PERCENT_OFFSET=
```

Command Parameters

The following table shows the command parameters:

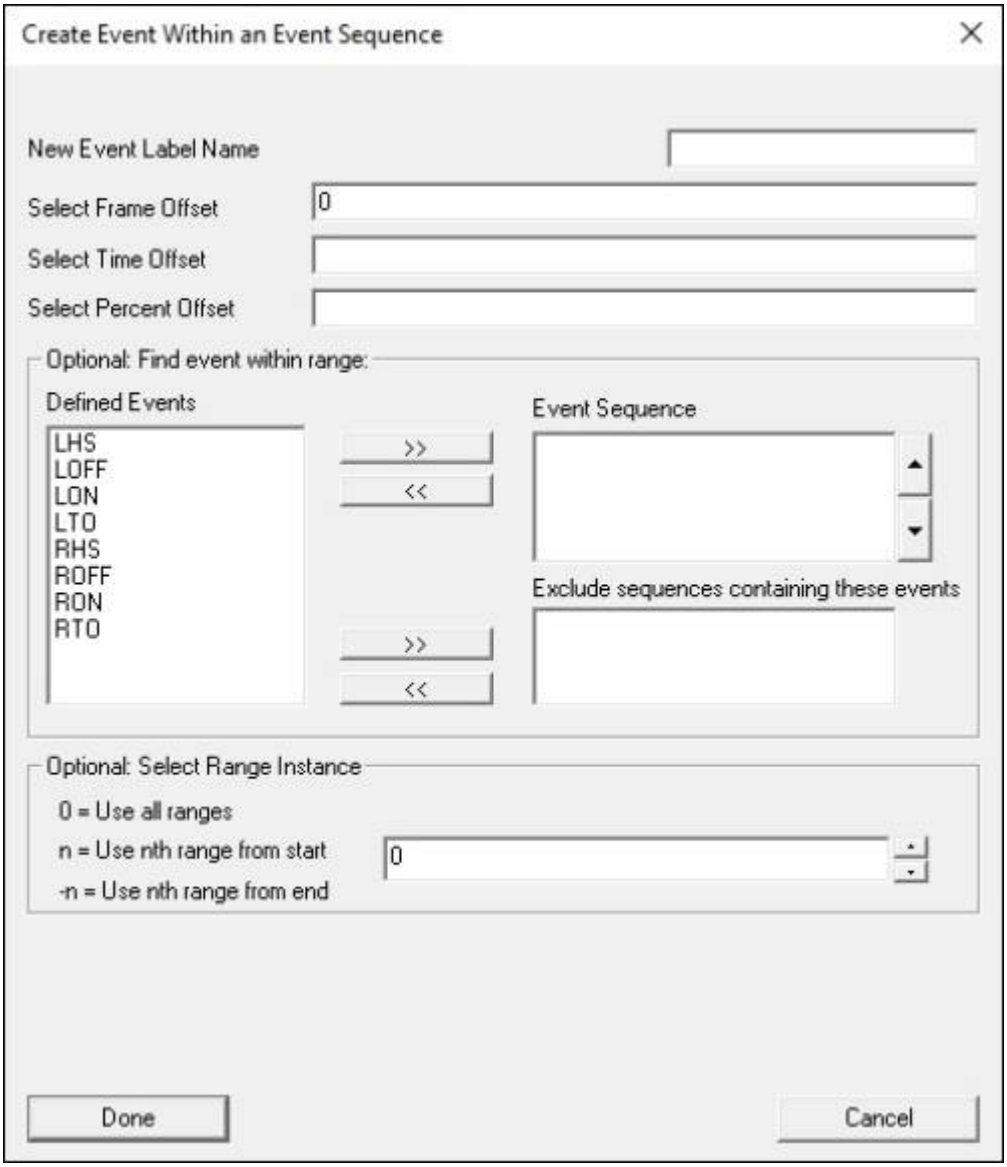
Parameter	Parameter Description
/NEW_EVENT_NAME=	Specifies the name of the new event to be created.
! /RANGE_INSTANCE= 0	The number of instances to label. **0**: Uses all instances found in the dataset, **1,2, etc**: Uses a specific numbered instance.
! /EVENT_SEQUENCE=	Defines the event sequence that the new event should be created between. (i.e. StartEvent, EndEvent)
! /EXCLUDE_EVENT=	Skips an event sequence if it contains one of the events listed in this parameter. (ex. BAD event)
! /FRAME_OFFSET= 0	Specifies frame offset from first event of sequence. **0**: No offset. **Positive value**: Shifts the event forward in time. **Negative value**: Shifts the event backward in time
! /TIME_OFFSET=	Specifies time offset from first event of sequence.
! /PERCENT_OFFSET=	Specifies percent offset from first even of sequence.

Offsets (**FRAME_OFFSET**, **TIME_OFFSET**, and **PERCENT_OFFSET**) allow fine control over the event

placement, only one may be active and specified when the command is executed. By default, value is 0.

Dialog

After adding the **Event_Between** command into the pipeline on the Visual3D application, the user can double-click with the Left Mouse Button in order to to open the following dialog:



- **Event Parameters**: This section defines the new event being created. Users specify the event name and can adjust its position using frame, time, or percentage offsets. These offsets allow for fine-tuning where the event is placed relative to the reference events.
- **Event Selection and Filtering**: Users choose which events to use as reference points for defining the new event. A list of predefined events is available for selection, and users can specify the event sequence between which the new event should be placed. Additionally, certain sequences containing specific events can be excluded to refine event placement.
- **Range Instance Selection**: This section determines how the command handles multiple occurences of the event sequence.

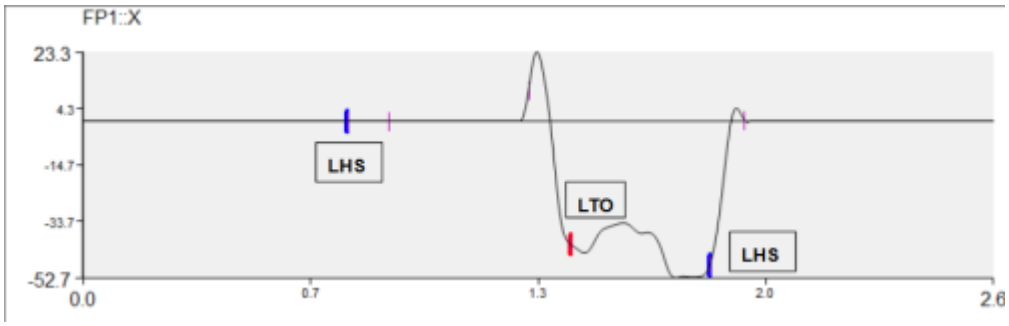
Examples

The following examples will go through the use of the **Event_Between** command in the Visual3D application.

Example 1

This pipeline will be showcasing the use of the **Event_Between** command when we want to place an event between the LHS and LTO in our trial, determined by the starting command, Automatic_Gait_Events.

As a reminder, force assignments must exist in a dynamic trial in order for gait events to be detected. In the force signal graph below, we are able to see the **LHS and LTO** gait events are highlighted.



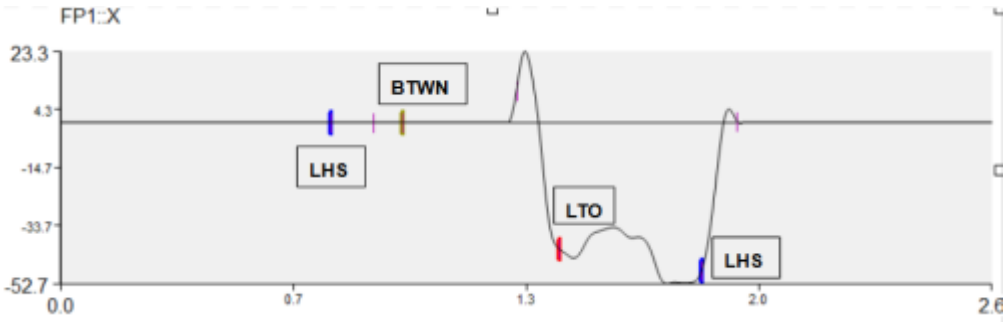
The following pipeline can be used to generate the “BTWN” event. Parameter values used are listed below:

- **RANGE_INSTANCE = 0** → all instances of the Event_Sequence will be used.
- **FRAME_OFFSET = 10** → the new event will be created 10 frames after the first event.

```
Automatic_Gait_Events
! /FRAME_WINDOW=8
! /USE_TPR=TRUE
! /TPR_EVENT_INSTANCE=1
;

Event_Between
/NEW_EVENT_NAME= BTWN
! /RANGE_INSTANCE=0
/EVENT_SEQUENCE=LHS+LTO
! /EXCLUDE_EVENTS=
/FRAME_OFFSET=10
! /TIME_OFFSET=
! /PERCENT_OFFSET=
;
```

After running this pipeline, we can highlight this event to show the updated graph with the new event label.



Example 2

This next example provides a more in-depth use of the command in a larger pipeline. Here, the objective is to use **Event_Between** to identify the **MidSwing** event between Left Toe Off (LTO) and Left Heel Strike (LHS), then use additional commands to:

1. Calculate hip and knee angles at the MidSwing event.
2. Create a new event when knee flexion surpasses a threshold.

First, we will use the same step from the previous example to generate gait events for the data, and then use the command to create an event named “MidSwing”:

```
Automatic_Gait_Events
! /FRAME_WINDOW=8
! /USE_TPR=TRUE
! /TPR_EVENT_INSTANCE=1
;

Event_Between
/NEW_EVENT_NAME=MidSwing
/RANGE_INSTANCE=1
/EVENT_SEQUENCE=LTO+LHS
! /EXCLUDE_EVENTS=
! /FRAME_OFFSET=0
! /TIME_OFFSET=
/PERCENT_OFFSET= 50
;
```

Next, we must output signals Left_Hip_Angle and Left_Knee_Angle can be used for further analysis.

```
Compute_Model_Based_Data
/RESULT_NAME= Left_Hip_Angle
! /SUBJECT_TAG=
/FUNCTION=JOINT_ANGLE
/SEGMENT=Left_Hip
! /REFERENCE_SEGMENT=LAB
! /RESOLUTION_COORDINATE_SYSTEM=LAB
! /USE_CARDAN_SEQUENCE=FALSE
! /NORMALIZATION=FALSE
! /NORMALIZATION_METHOD=
```

```
! /NORMALIZATION_METRIC=
! /NEGATEX=FALSE
! /NEGATEY=FALSE
! /NEGATEZ=FALSE
! /AXIS1=X
! /AXIS2=Y
! /AXIS3=Z
! /TREADMILL_DATA=FALSE
! /TREADMILL_DIRECTION=UNIT_VECTOR(0,1,0)
! /TREADMILL_SPEED=0.0
;

Compute_Model_Based_Data
/RESULT_NAME= Left_Knee_Angle
! /SUBJECT_TAG=
/FUNCTION=JOINT_ANGLE
/SEGMENT=Left_Knee
! /REFERENCE_SEGMENT=LAB
! /RESOLUTION_COORDINATE_SYSTEM=LAB
! /USE_CARDAN_SEQUENCE=FALSE
! /NORMALIZATION=FALSE
! /NORMALIZATION_METHOD=
! /NORMALIZATION_METRIC=
! /NEGATEX=FALSE
! /NEGATEY=FALSE
! /NEGATEZ=FALSE
! /AXIS1=X
! /AXIS2=Y
! /AXIS3=Z
! /TREADMILL_DATA=FALSE
! /TREADMILL_DIRECTION=UNIT_VECTOR(0,1,0)
! /TREADMILL_SPEED=0.0
;
```

Following the creation of the hip and knee angle signals for the left leg during the gait cycle, the **Metric_Signal_Value_At_Event** command is used to extract the values specifically at the MidSwing event

```
Metric_Signal_Value_At_Event
/SIGNAL_TYPES= LINK_MODEL_BASED
! /SIGNAL_FOLDER=ORIGINAL
/SIGNAL_NAMES= Left_Hip_Angle
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME= Left_Hip_Angle_at_MidSwing
! /APPLY_AS_SUFFIX_TO_SIGNAL_NAME=FALSE
! /SIGNAL_COMPONENTS=
! /COMPONENT_SEQUENCE=
/EVENT_NAME= MidSwing
! /EVENT_INSTANCE=0
! /SCALE_FACTORS=1
! /GENERATE_MEAN_AND_STDDEV=TRUE
```

```
! /APPEND_TO_EXISTING_VALUES=FALSE
! /GENERATE_VECTOR_LENGTH_METRIC=FALSE
! /RETAIN_NO_DATA_VALUES=FALSE
;

Metric_Signal_Value_At_Event
/SIGNAL_TYPES= LINK_MODEL_BASED
! /SIGNAL_FOLDER=ORIGINAL
/SIGNAL_NAMES= Left_Knee_Angle
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME= Left_Knee_Angle_at_MidSwing
! /APPLY_AS_SUFFIX_TO_SIGNAL_NAME=FALSE
! /SIGNAL_COMPONENTS=
! /COMPONENT_SEQUENCE=
/EVENT_NAME= MidSwing
! /EVENT_INSTANCE=0
! /SCALE_FACTORS=1
! /GENERATE_MEAN_AND_STDDEV=TRUE
! /APPEND_TO_EXISTING_VALUES=FALSE
! /GENERATE_VECTOR_LENGTH_METRIC=FALSE
! /RETAIN_NO_DATA_VALUES=FALSE
;
```

Following this, the Event_Threshold command is used to create an event named **Max_Knee_Flexion** when the **Left_Knee_Angle** exceeds 45 degrees, identifying the point of maximum knee flexion during the swing phase.

```
Event_Threshold
/RESULT_EVENT_NAME= Max_Knee_Flexion
/SIGNAL_TYPES= LINK_MODEL_BASED
! /SIGNAL_FOLDER=ORIGINAL
/SIGNAL_NAMES= Left_Knee_Angle
! /SIGNAL_COMPONENTS=
! /FRAME_OFFSET=0
! /TIME_OFFSET=
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /EVENT_SEQUENCE_INSTANCE=0
! /EVENT_SUBSEQUENCE=
! /SUBSEQUENCE_EXCLUDE_EVENTS=
! /EVENT_SUBSEQUENCE_INSTANCE=0
! /EVENT_INSTANCE=0
/THRESHOLD= 45
! /ON_ASCENT=
! /ON_DESCENT=
! /FRAME_WINDOW=8
! /ENSURE_FRAMES_BEFORE=FALSE
! /ENSURE_FRAMES_AFTER=FALSE
;
```