helper_sequences.hps

Helper functions for creating sequences of events

Sequence_Begin

```
void Sequence_Begin(const tString &in asFunction,
                    cSequenceStatesData &aData)
```

Begins to define a sequence of events. Note that the entire specified function will be run once for every sequence step.

- **asFunction**: the name of the function that Sequence_Begin is called from. Syntax: void Func(const tString &in asSequence)
- **aData**: object containing the state of the sequence.

Sequence_End

```
void Sequence_End()
```

Ends the definition of a sequence.

Sequence_Stop

```
void Sequence_Stop()
```

Stops the current sequence immediately.

Sequence_DoStepAndWait

```
bool Sequence_DoStepAndWait(float afTime)
```

Use as an if statement condition to define a single step in a sequence. A step will only be run once per sequence, and delays execution of the next step by a specified time.

- **afTime**: the time before the next step is called.

Sequence_DoStepWaitAndRepeat

```
bool Sequence_DoStepWaitAndRepeat(int alNumOfRepetitions,
                                  float afWaitTime)
```

Sequence_DoStepAndContinue

```
bool Sequence_DoStepAndContinue()
```

Use as an if statement condition to define a single step in a sequence. A step will only be run once per sequence. This one does not wait and just jumps to the next step

---

Sequence_DoStepAndPause

```
bool Sequence_DoStepAndPause(float afTime=)
```

Use as an if statement condition to define a single step in a sequence. Works similar to Sequence_DoStepAndWait, but pauses the sequence when the step has been executed. To resume, use SequenceStates_Resume("FunctionName")

- **afTime**: the time before the next step is called after the sequence has been resumed, 0 by default.

Sequence_Wait

```
void Sequence_Wait(float afTime)
```

This is just like an if statement with DoStepAndWait only it does nothing, ie if(Sequence_DoStepAndWait(x)){} Add it to just get a wait before the next step is run

- **afTime**: the time before the next step is called after the sequence has been resumed, 0 by default.

Sequence_Pause

```
void Sequence_Pause()
```

This is just like an if statement with Sequence_DoStepAndPause only it does nothing, ie if(Sequence_DoStepAndPause(x)){} Add it to just get a pause

Sequence_SkipNextSteps

```
void Sequence_SkipNextSteps(int alStepsToSkip)
```

Skips a number of steps of the sequence.

- **alStepsToSkip**: the number of steps to skip.

Sequence_SkipNextStep

```
void Sequence_SkipNextStep()
```

Skips the next step of the sequence.

SequenceStates_Pause

```
void SequenceStates_Pause(tString &in asName)
```

SequenceStates_Resume

```
void SequenceStates_Resume(tString &in asName)
```

SequenceStates_Stop

```
void SequenceStates_Stop(tString &in asName)
```

SequenceStates_IsActive

```
bool SequenceStates_IsActive(tString &in asName)
```
