Week 06 MEL for animation

- The currentTime and setKeyframe commands.
- The Expression Editor
- Different between MEL script and expression
- The sin() function

MEL for animation: introduction

So far the MEL scripts we have written are all "run once": we execute it, the script does something, and then finish. In such a way we cannot create any animation effects – something related to the time line.

To create animation, we can have 2 different ways:

1. Use the **`currentTime`** and **`setKeyframe`** MEL command:
   - The **`currentTime`** command can be used to set the current frame. The **`setKeyframe`** command is used to set a key frame for an attribute.

   Therefore, by repeatedly calling the **`currentTime`** and **`setKeyframe`**, we can create key frame animation using MEL.

2. Use the **Expression Editor**:
   - Expressions are just **scripts stored in a scene**, and **will be executed at every frame**. We can create animation effect by writing expressions.
   - Expressions are also useful for **linking attributes between different objects** - where a change in one attribute alters the behavior of the other.

currentTime and setKeyframe

currentTime

To set the current frame to frame 5, just write down:

```
currentTime 5;
```

setKeyframe

To set a key frame (x=3.8, y=2.7, z=5) for the **current selected object**, you can use:

```
setKeyframe -value 3.8 -attribute "translateX";
setKeyframe -value 2.7 -attribute "translateY";
setKeyframe -value 5   -attribute "translateZ";
```

Other attributes can be set in the same way.

In order to create the keyframe animation using MEL, the pattern is:

1. At frame 1, set the attribute
```
currentTime 1;

setKeyframe –value ?? –attribute “??”;
setKeyframe –value ?? –attribute “??”;
```

2. At frame 5, says, set the attribute:
```
currentTime 5;

setKeyframe –value ?? –attribute “??”;
setKeyframe –value ?? –attribute “??”;
```

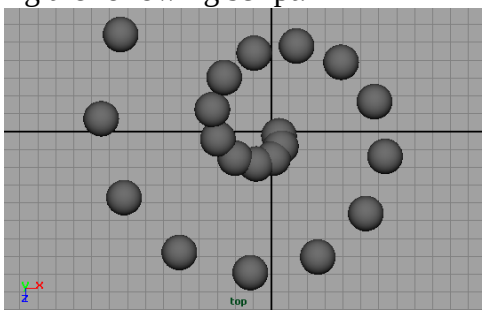
3. Go to **step 2.** and repeat until you are satisfied with the result.

4.

Class exercise: sphere moving along spiral path

In Week 2 we created a "spiral spheres" model using the following script:

```
float $y = 0;
float $angle = 0;
while ( $y < 10 ) {
    $y = $y + 0.5;
    $angle = $angle + 0.5;
    float $x = $y * cos( $angle );
    float $z = $y * sin( $angle );
    sphere -pivot $x $y $z;
}
```

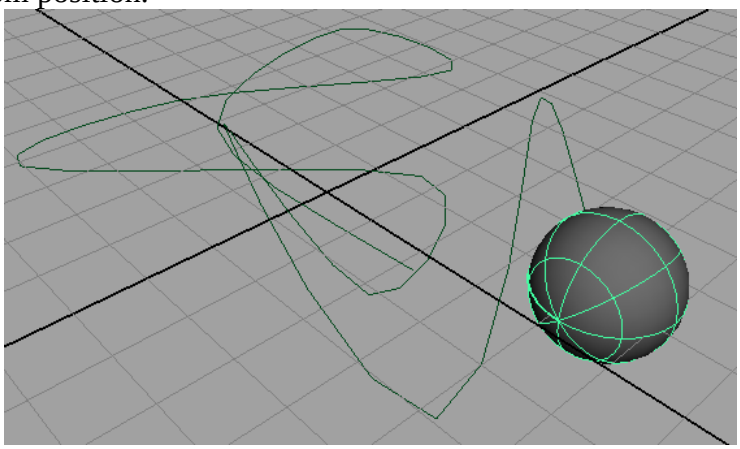


Modify this script, so that instead of creating a lot of sphere, now **you have one sphere moving along the spiral path**.

Hints: inside the while-loop, instead of calling **`sphere,`** use the **`currentTime`** and **`setKeyframe`** commands to set key frames for the sphere.

Class exercise: sphere moving at random position

Create a sphere, and use **`currentTime`** and **`setKeyframe`** commands to make the sphere moving at random position.



Note that moving the sphere at random position **at every frame** is not a good idea. A better way is to put the key-frames further apart, and then let Maya to do the in-between.

The Expression Editor

Create an expression

Another way to create animation using MEL is to use the Expression Editor. Let's try the following simple example:

Step 1) Create a sphere, and rename it as "**`ball`**".

Step 2) Open the Expression Editor, by choosing menu item:
Window > Animation Editors > Expression Editor…

Step 3) At the "Expression Name" entry, assign a name to your expression (say, *test1*).

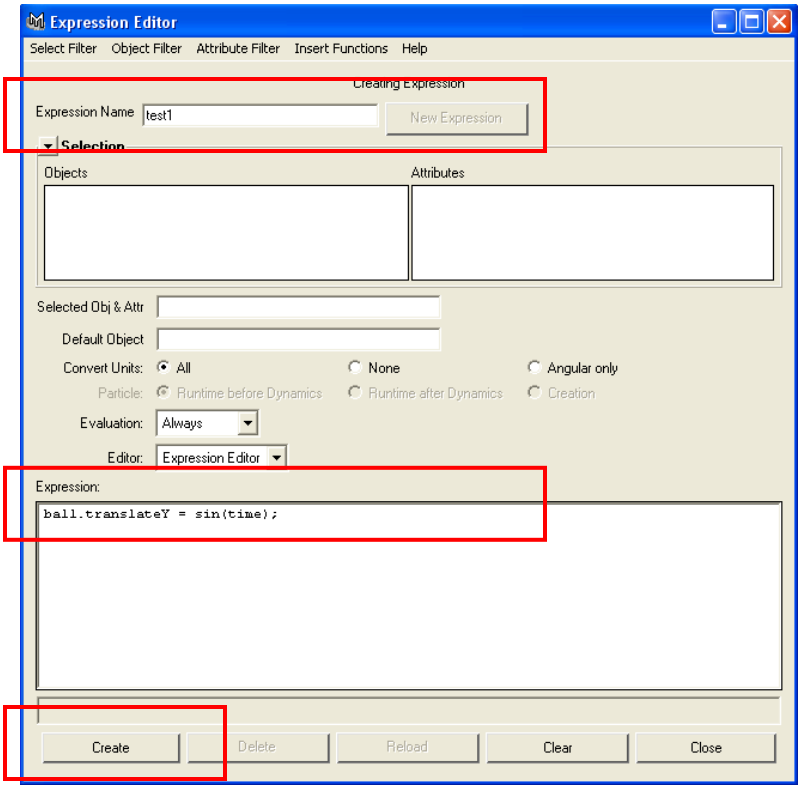
Step 4) At the lower part under "expression", type the following:

```
ball.translateY = sin(frame);
```

Step 5) Press the "Create" button.

Now an expression is created and stored in a node called *test1*. (From the Hypergraph Window you can find the node.) The expression will also be saved together with the scene.

Play the scene to see what happen.



Modify an expression

If you want to modify an existing expression, you can:
Step 1) Open the Expression Editor.
Step 2) From the menu, choose **Select Filter > By Expression Name**.
Step 3) Under the section "**Selection"**, find your expression name and click on it.
Step 4) The original expression will now be shown at the "**Expression**" section. Modify your expression, and click the "Edit" button once you have finished. For example, try to modify your expression as:

```
ball.translateY = sin(time);
```

Play the scene to see what happen. You may want to extend the range of your timeline. I will explain the meaning of the expression soon.

Different between MEL script and expression

There are only two differences between expression and MEL syntax:

1. Expression will be executed at **every frame**, while MEL is usually "run once".
2. In expression you can direct access the object attributes. For example:
   ```
   ball.translateX = 5.7;
   ```
3. In expression you can have the pre-defined **`time`** and **`frame`** variables.

Direct access of object attributes

In expression, you can write down something like this:

```
ball.translateX = 5.7;
```

In MEL syntax, you have to use **`setAttr`**:

```
setAttr( "ball.translateX", 5.7 );
```

Similar idea applied to the **`getAttr`** command. In expression, you can write down:

```
float $x = ball.translateX;
```

or something like:

```
cube1.scaleZ = ball.translateX * 5 + 10;
```

But in MEL, you have to use:

```
float $x = getAttr( "ball.translateX" );
```

The use of the `time` and `frame` variables

In expression, you have two pre-defined variables: **`time`** and **`frame`**.

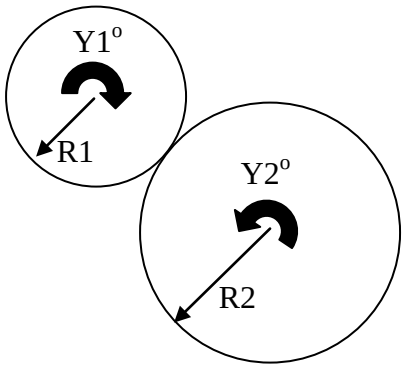
`time`: the current time, in second

`frame`: the current frame

For example, you can have an expression like this (guess what will happen):

```
if (frame == 1) {
    ball.translateY = 0;
} else {
    ball.translateY = ball.translateY + time;
}
```

Class exercise: gears

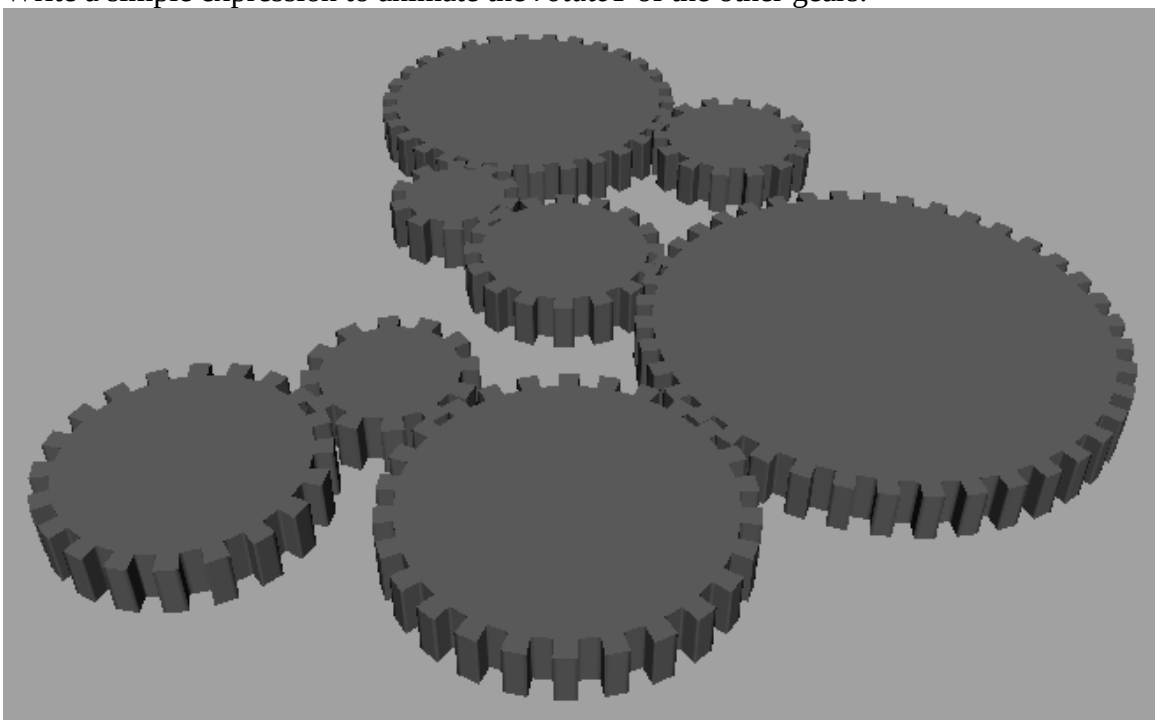


As shown in the above figure, *gear1* (with radius R1) touches *gear2* (with radius R2). If *gear1* rotated Y1 degree, then *gear2* will rotate Y2 degree, where

Y2 = -1 * Y1 * R1 / R2

You are given the scene file **J:\SM3122\Week06-gear.mb**. It contains 8 gears with different radius. Each gear has an extra attribute called ***radius***. Moreover, the gear1's *rotateY* is key-framed.

Write a simple expression to animate the *rotateY* of the other gears.



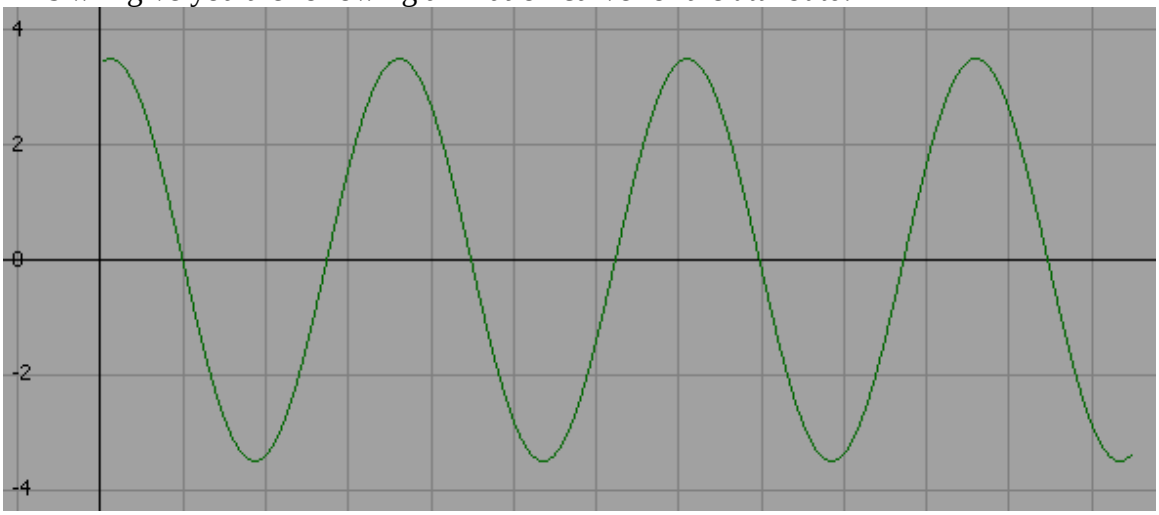
The *sin()* function

Though you can create an expression to animate attributes for any purpose, they are ideal for attributes that change incrementally, randomly, or rhythmically over time.

The *sin*() function is a commonly used function for creating a **cyclic** motion. The basic syntax of a general *sin*() function is:

```
attribute = magnitude * sin( (time + offset) * speed );
```

This will give you the following animation curve for the *attribute*:



In the expression, there are 3 variables that you can fine-tune:

- The *speed* is used to control the speed of the motion.
- The *offset* is used to control the amount of "horizontal-shift" of the curve.
- The *magnitude* is used to control the height of the curve, i.e. the magnitude of the motion.

For example, you can write down the following expression:

```
ball.translateY = 3.5 * sin( (time + 0.75) * 1.8 );
```

Class exercise: a waving flag

Step 1) Create a curve with 5 CVs. (You don't need to use MEL script. Just use the Maya interface.)

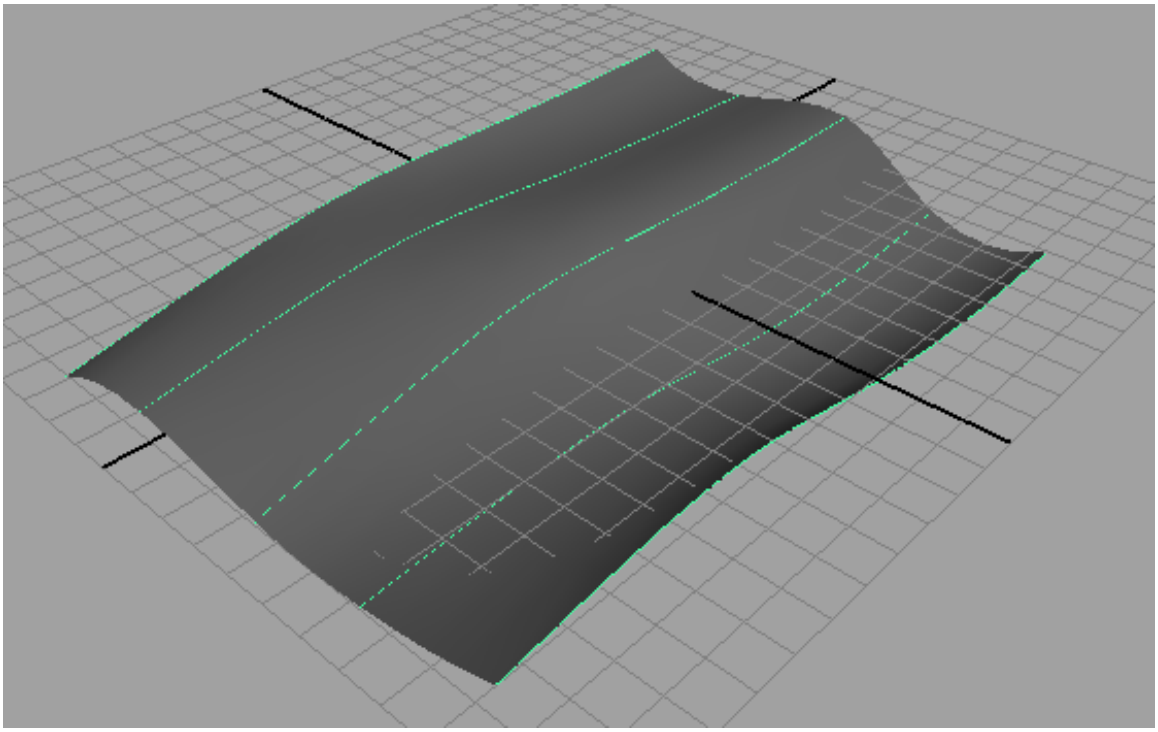
Step 2) Create an expression to control the 5 CVs. The *y position* of each CV's will follow the *sin*() function, with different *offset*. (Hints: Use **`select`** and **`move`** command to move a CV).

Step 3) Create 4 more curves, each has 5 CVs.

Step 4) Modify your expression to control all the CVs. Again, the *y position* of all the CV's will follow the *sin*() function, with different *offset*.

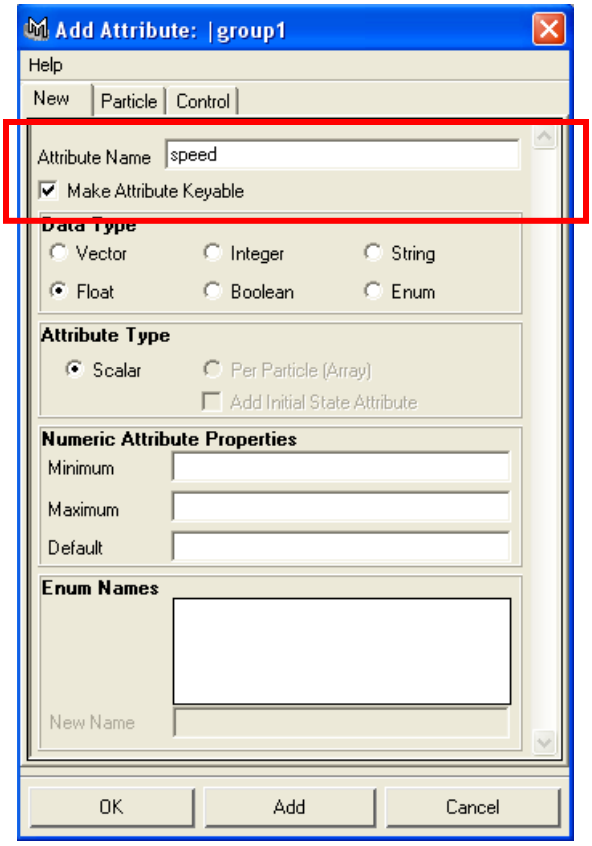
Step 5) Select all the curves, and use **Surfaces > Loft** to "loft" a surface using the 5 curves (keeping the construction history).

Step 6) Play the scene to see a "waving flag".



Additional note – adding extra attribute

When you know how to write simple expression, you may want to add extra attribute for your object. This extra attribute will be a "linkage" between the CHANNEL BOX and your expression: i.e. you can control your expression's variable through the CHANNEL BOX.



Use the exercise "waving flag" as an example. I can group the 5 curves into a group, add an extra attribute called *speed*, and use this *speed* attribute in my expression:

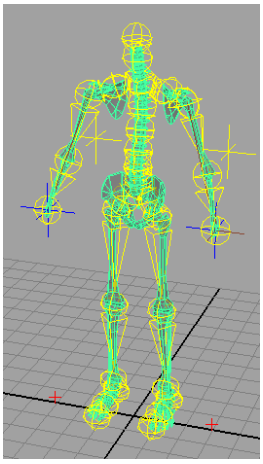
Step 1) Group the 5 curves into a group called group1.
Step 2) Select the group1, and choose
Modify > Add Attribute…
Step 3) In Attribute Name, type speed. Then press OK.
Step 4) Modify the expression we have in the following way:

```
select curve1.cv[0];
move -y (sin((time + 0.5) * group1.speed));
select curve1.cv[1];
move -y (sin((time + 1.0) * group1.speed));
……
……
```

Now, in the CHANNEL BOX, select group1 and you can find the attribute *speed*. Change this attribute can change the moving speed of our waving-flag.

Class exercise: expression driven "walk cycle"

(Need to hand in at the end of the class)



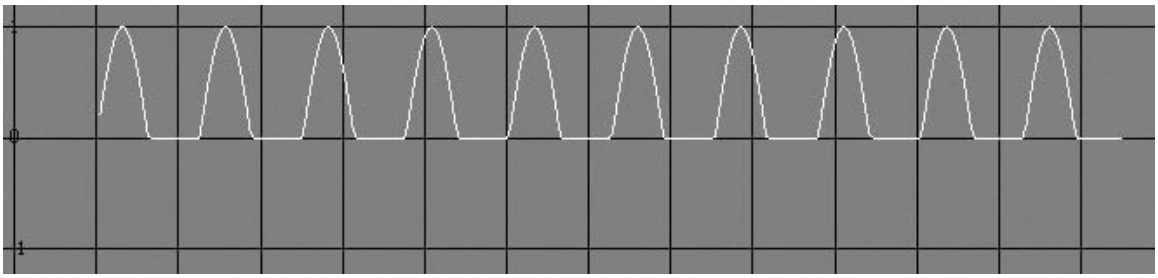
Given the scene file **J:\SM3122\mrBlah.mb**, try to do the following:

Create a basic "walk cycle"

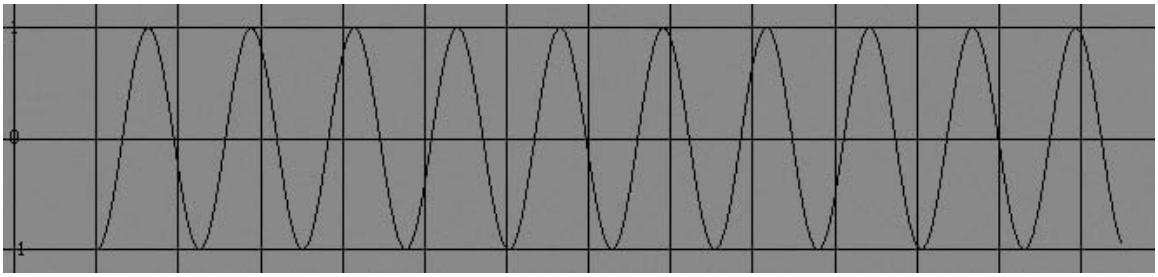
Create a new expression, and use *sin*() to control the *translateY* and *translateZ* of the node *l_foot* and *r_foot*.
(Your result may looks like **J:\SM3122\ mrBlah-demo.mb.)**

Hints: the animation curves of *l_foot* and *r_foot* should be something like the following. How to create these motions using *sin*()?

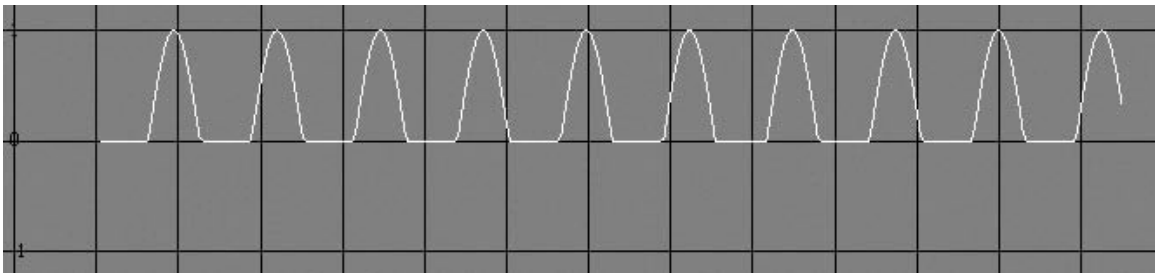
l_foot translateY



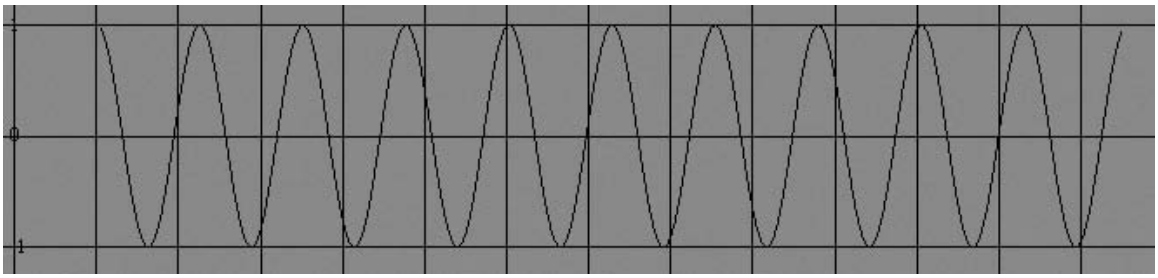
l_foot translateZ



r_foot translateY



r_foot translateZ



Bonus parts

Try also controlling the other parts of the skeleton, so that your walk-cycle looks better. (Example: **J:\SM3122\ mrBlah-bonus.mb).** Try to add the followings as much as you can:

Make the pelvis up and down, and also side-shift

Modify the statements in your expression, so that the pelvis will also move up and down, and a little bit side-shift during the motion.
Hints: use statements to control the *translateX* and *translateY* of *m_pelvisRot*.

Twist the hip

Add more statements to your expression, so that the hip will also twist during the motion. Moreover, the hip should also rotate side-way a little bit to counter-balance the movement.
Hints: use statements to control the *rotateX* and *rotateY* of *m_backRoot*.

Twist the shoulder

Add more statements to your expression, so that the shoulder will also twist during the motion.
Hints: use statements to control the *rotateX* and *rotateY* of *m_shoulders*.

Swing the hands

Add more statements to your expression, so that the hands will also swing during the motion.
Hints: use statements to control the *translateZ* of *l_wristLocator* and *r_wristLocator*.

Add randomness

Modify the statements in your expression, so that the motion of *l_foot* and *r_foot*'s will have a little bit randomness.

Save the finished scene (note: your expression will also be saved together), and use your student number as the file name. Put the file into the folder J:\SM3122\submit\Week06\ at the end of this class.

**** Week 06 End ****