BadmintonMan.js

Deborah Chang

Overview

- Goal: Animation of the *Clear*
  - a difficult, but elementary badminton move

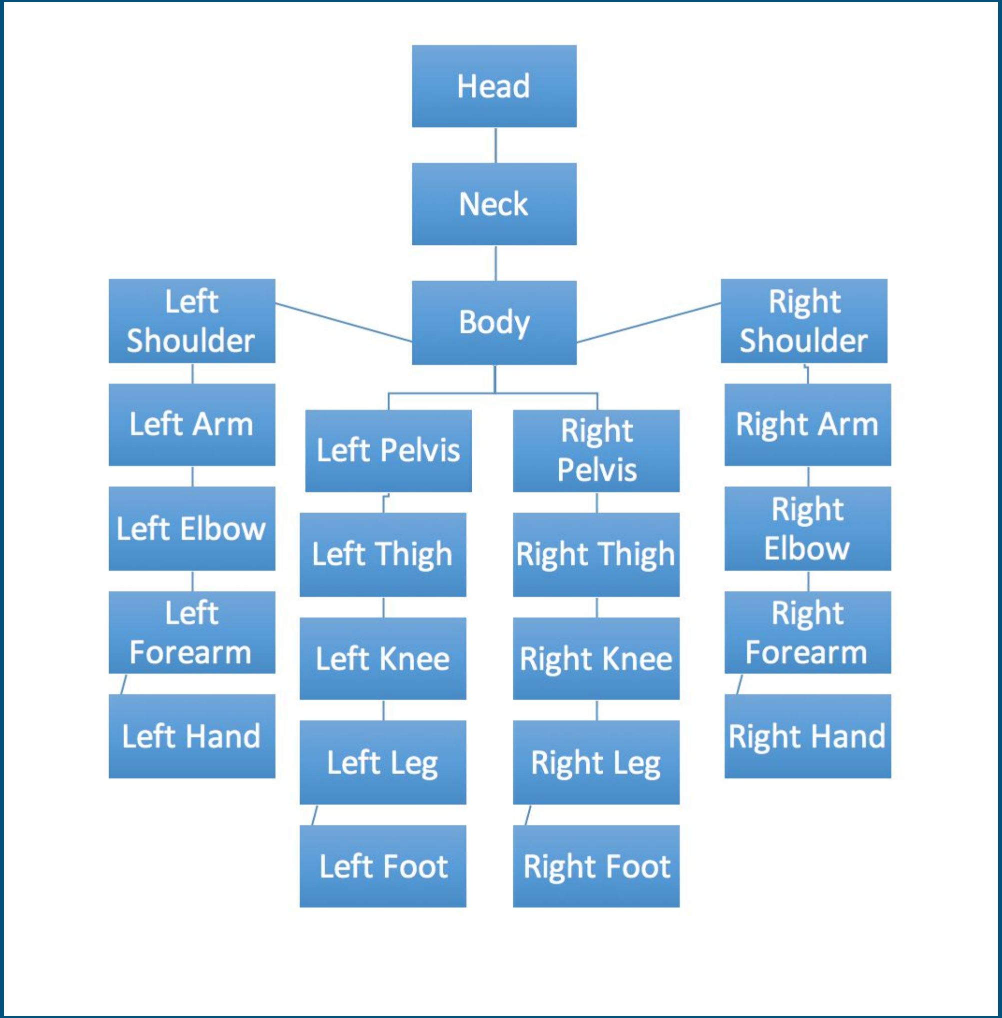
- Javascript

The Clear

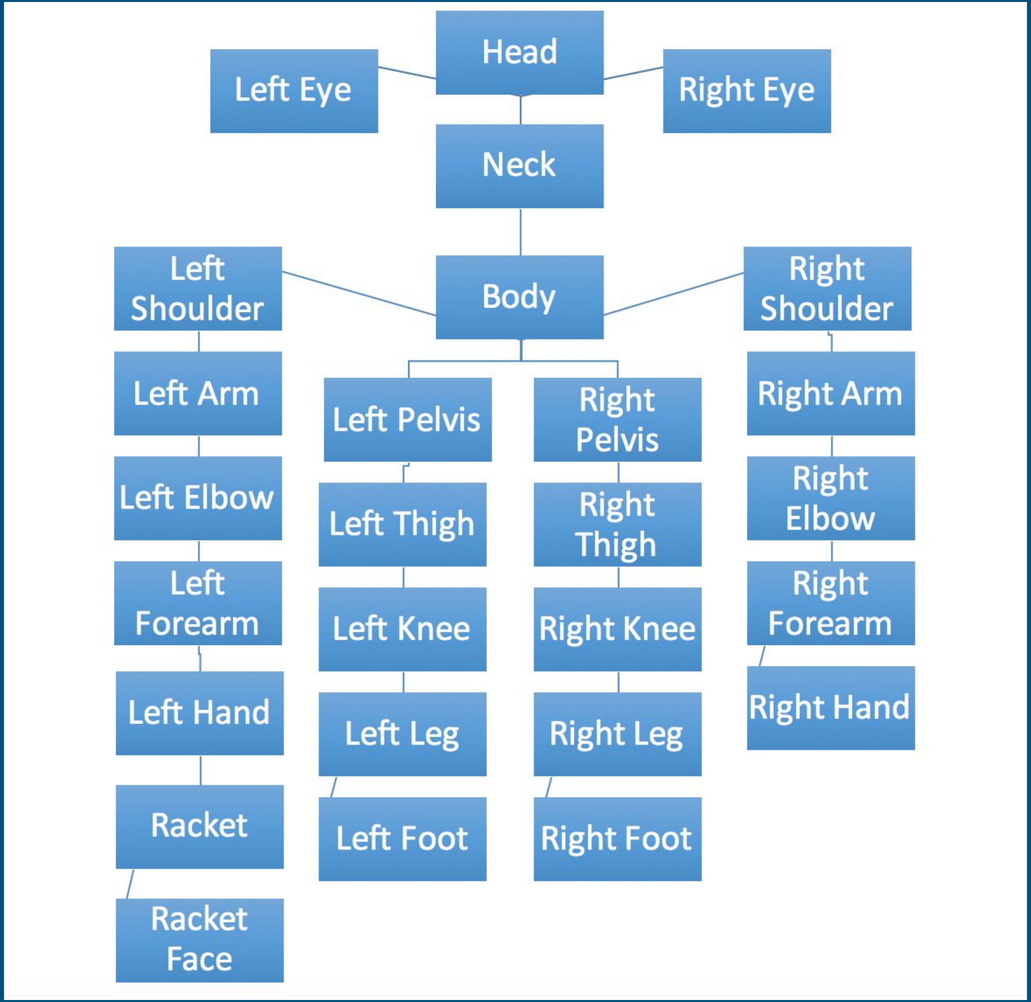
- Basic move

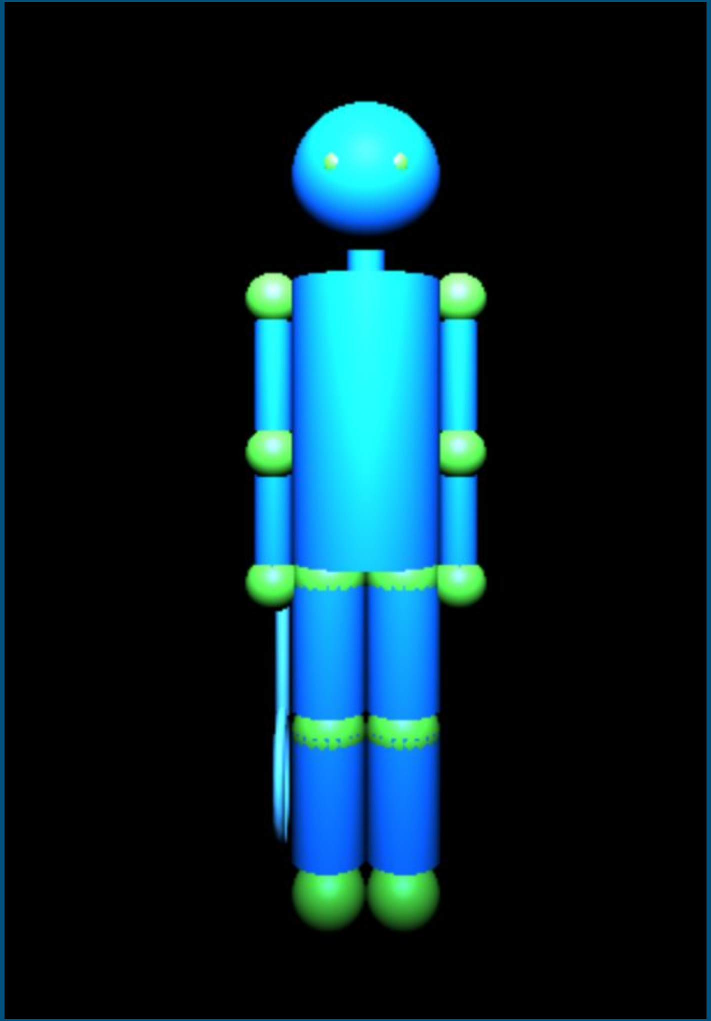
BadmintonMan

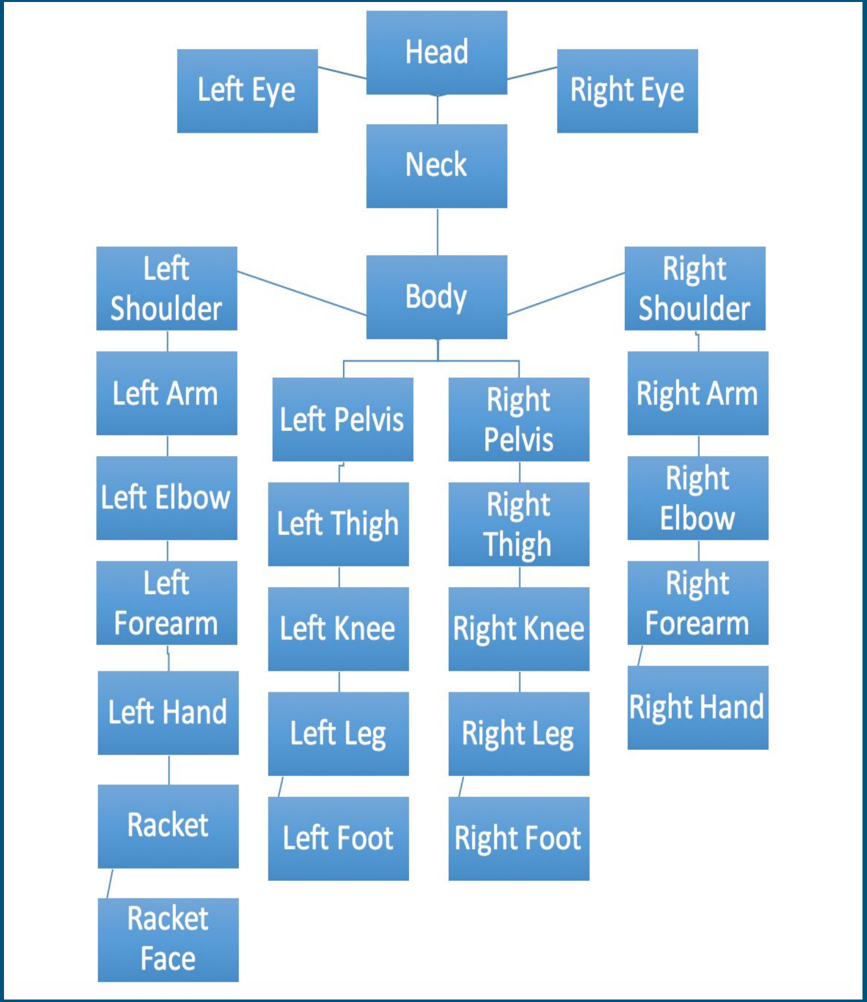
Hierarchy


Head
Neck
Body
Left Shoulder
Right Shoulder
Left Arm
Left Pelvis
Right Pelvis
Right Arm
Left Elbow
Left Thigh
Right Thigh
Right Elbow
Left Forearm
Left Knee
Right Knee
Right Forearm
Left Hand
Left Leg
Right Leg
Right Hand
Left Foot
Right Foot

Hierarchy

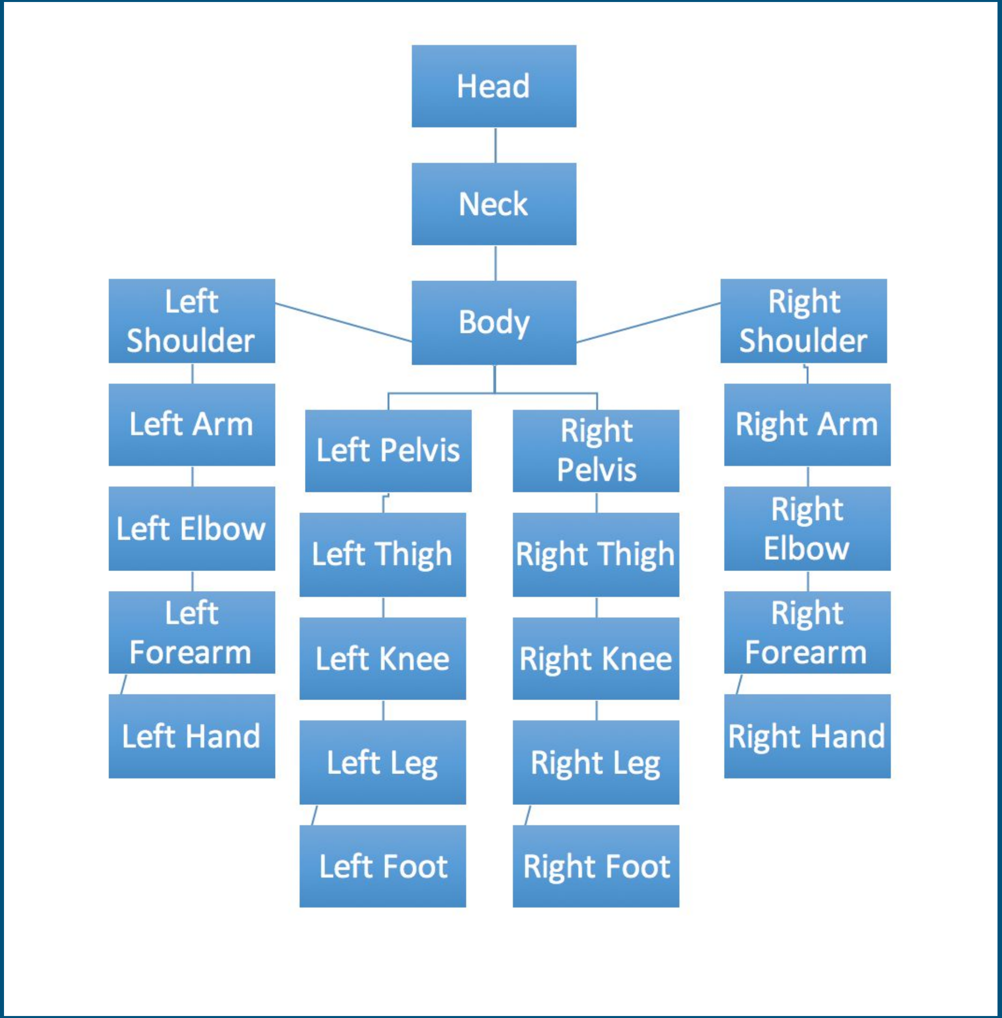

Head
Left Eye
Right Eye
Neck
Body
Left Shoulder
Right Shoulder
Left Arm
Left Pelvis
Right Pelvis
Right Arm
Left Elbow
Left Thigh
Right Thigh
Right Elbow
Left Forearm
Left Knee
Right Knee
Right Forearm
Left Hand
Left Leg
Right Leg
Right Hand
Racket
Left Foot
Right Foot
Racket Face



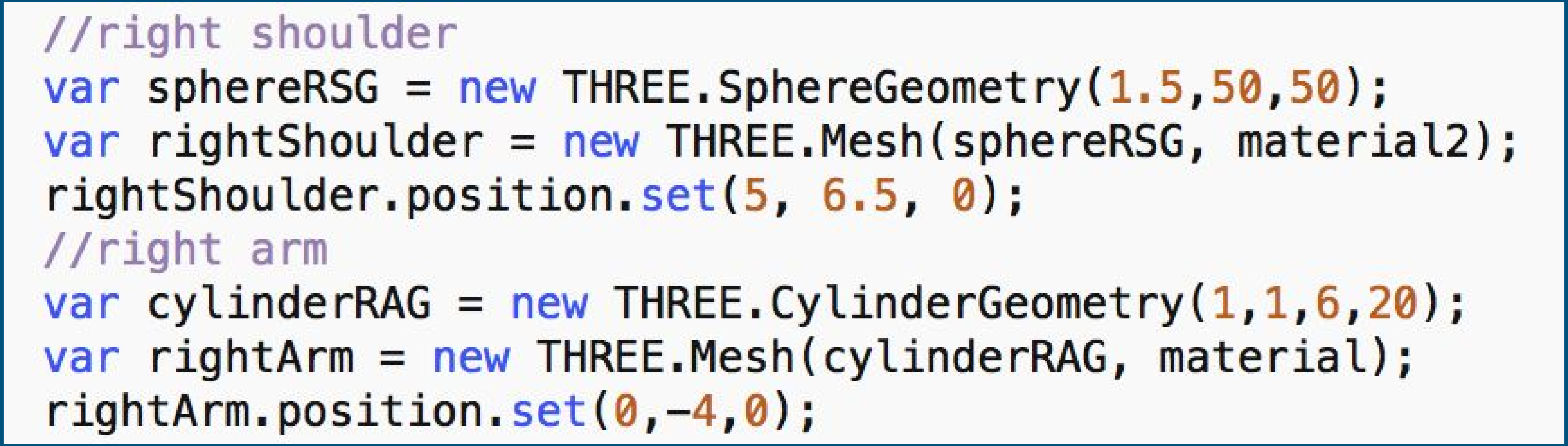

Head
Left Eye
Right Eye
Neck
Left Shoulder
Body
Right Shoulder
Left Arm
Left Pelvis
Right Pelvis
Right Arm
Left Elbow
Left Thigh
Right Thigh
Right Elbow
Left Forearm
Left Knee
Right Knee
Right Forearm
Left Hand
Left Leg
Right Leg
Right Hand
Racket
Left Foot
Right Foot
Racket Face

Hierarchy Ex.

```
leftLeg.add(leftFoot);

leftKnee.add(leftLeg);

leftThigh.add(leftKnee);

leftPelvis.add(leftThigh);

body.add(leftPelvis);
```

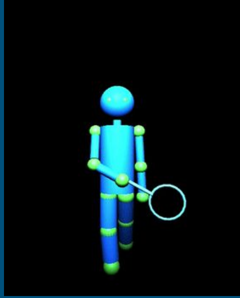


Creating Objects



```
//right shoulder
var sphereRSG = new THREE.SphereGeometry(1.5,50,50);
var rightShoulder = new THREE.Mesh(sphereRSG, material2);
rightShoulder.position.set(5, 6.5, 0);
//right arm
var cylinderRAG = new THREE.CylinderGeometry(1,1,6,20);
var rightArm = new THREE.Mesh(cylinderRAG, material);
rightArm.position.set(0,-4,0);
```

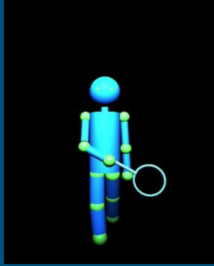
Linear Interpolation



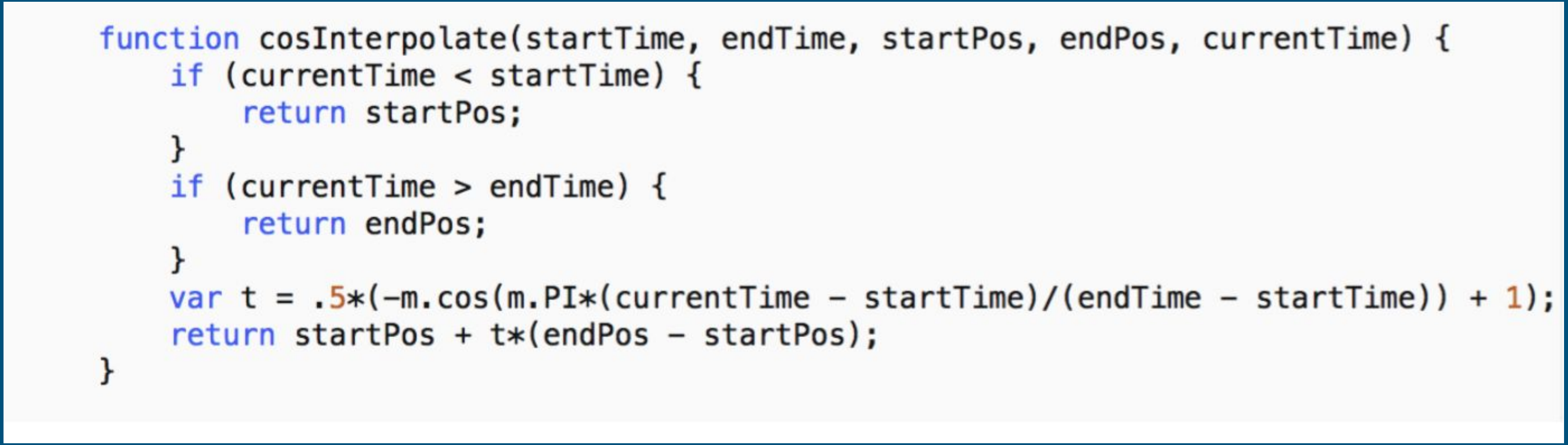
$$f(t_0, t_1, p_0, p_1, t) = (1 - \frac{t - t_0}{t_1 - t_0})p_0 + \frac{t - t_0}{t_1 - t_0}p_1$$

```
function linearInterpolate (startTime, endTime, startPos, endPos, currentTime) {
    if (currentTime < startTime) {
        return startPos;
    }
    if (currentTime > endTime) {
        return endPos;
    }
    var t = (currentTime - startTime)/(endTime - startTime);
    return (1-t)*startPos + t*endPos;
}
```

Sinusoidal Interpolation

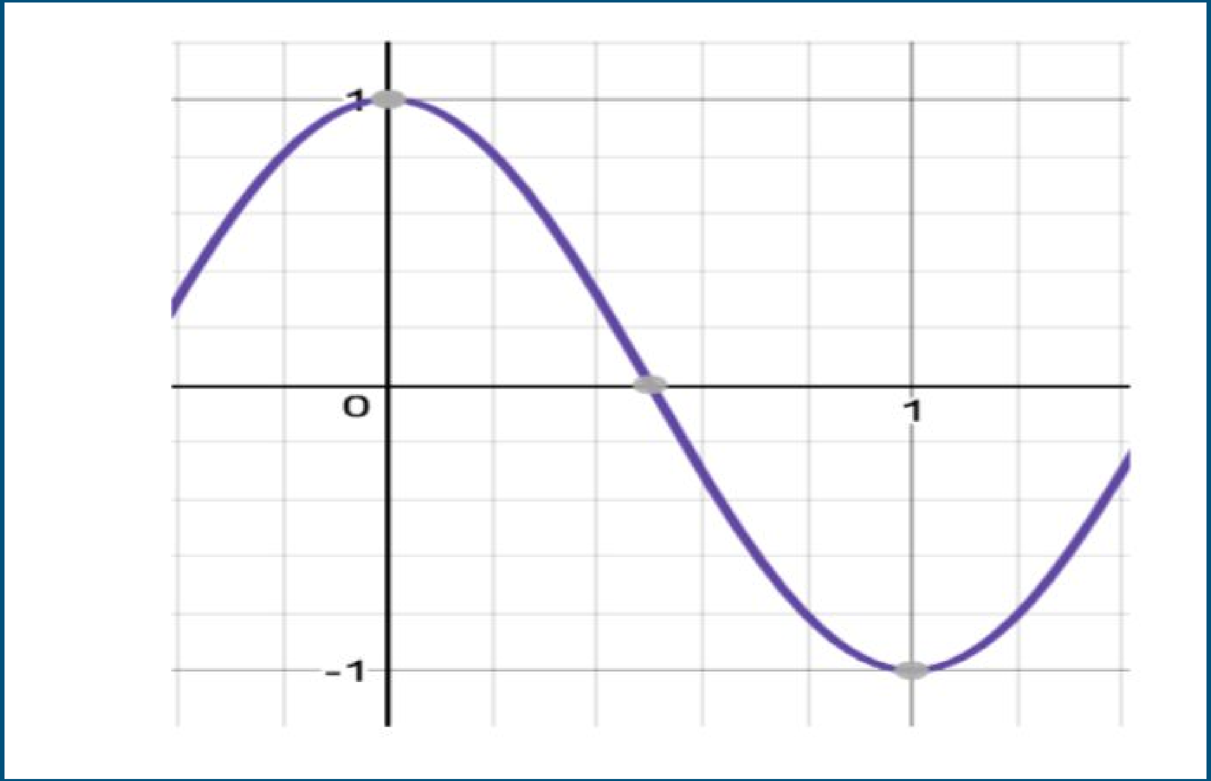


$$f(t_0, t_1, p_0, p_1, t) = p_0 + (0.5(-\cos(\pi(\frac{t - t_0}{t_1 - t_0})) + 1))(p_1 - p_0)$$

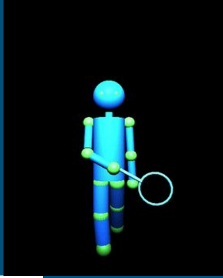


```
function cosInterpolate(startTime, endTime, startPos, endPos, currentTime) {
    if (currentTime < startTime) {
        return startPos;
    }
    if (currentTime > endTime) {
        return endPos;
    }
    var t = .5*(-m.cos(m.PI*(currentTime - startTime)/(endTime - startTime)) + 1);
    return startPos + t*(endPos - startPos);
}
```

Range Between (0) and (1)



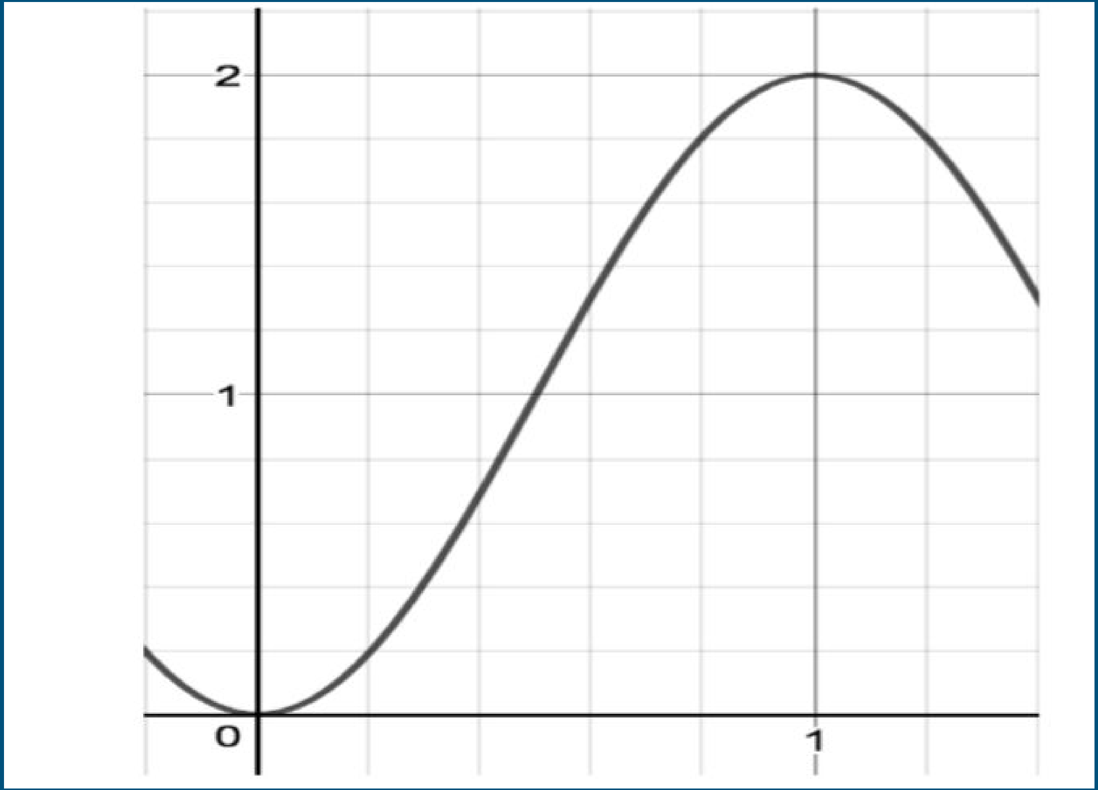
Sinusoidal Interpolation



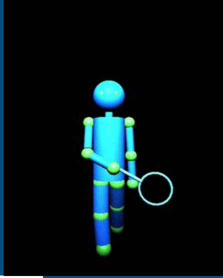
$$f(t_0, t_1, p_0, p_1, t) = p_0 + (0.5(-\cos(\pi(\frac{t - t_0}{t_1 - t_0})) + 1))(p_1 - p_0)$$

```
function cosInterpolate(startTime, endTime, startPos, endPos, currentTime) {
    if (currentTime < startTime) {
        return startPos;
    }
    if (currentTime > endTime) {
        return endPos;
    }
    var t = .5*(-m.cos(m.PI*(currentTime - startTime)/(endTime - startTime)) + 1);
    return startPos + t*(endPos - startPos);
}
```

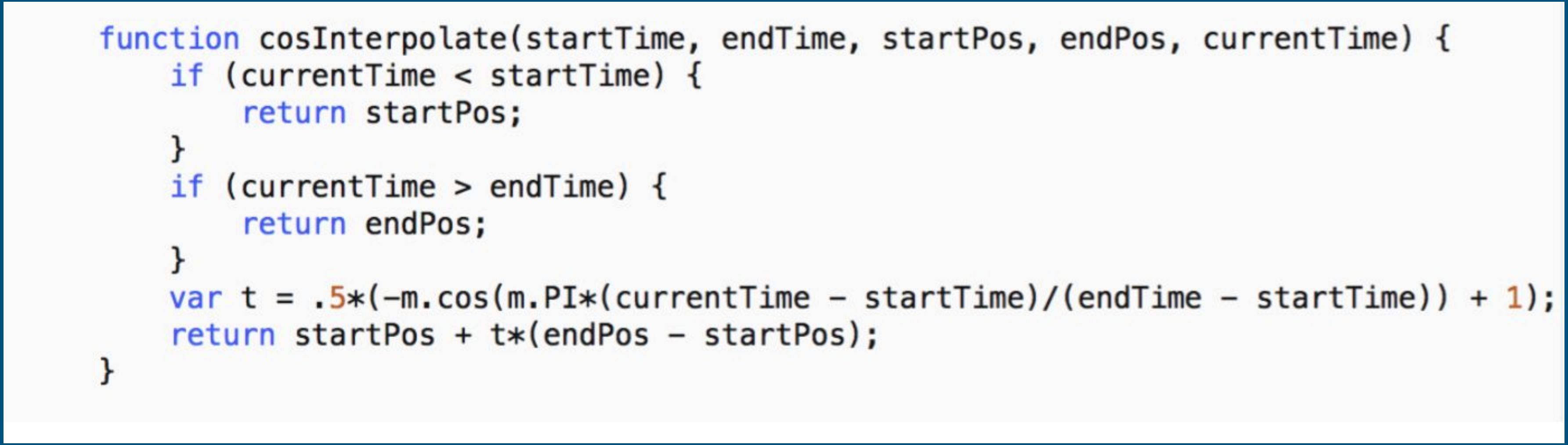
Multiply By (-1) & Add (1)



Sinusoidal Interpolation

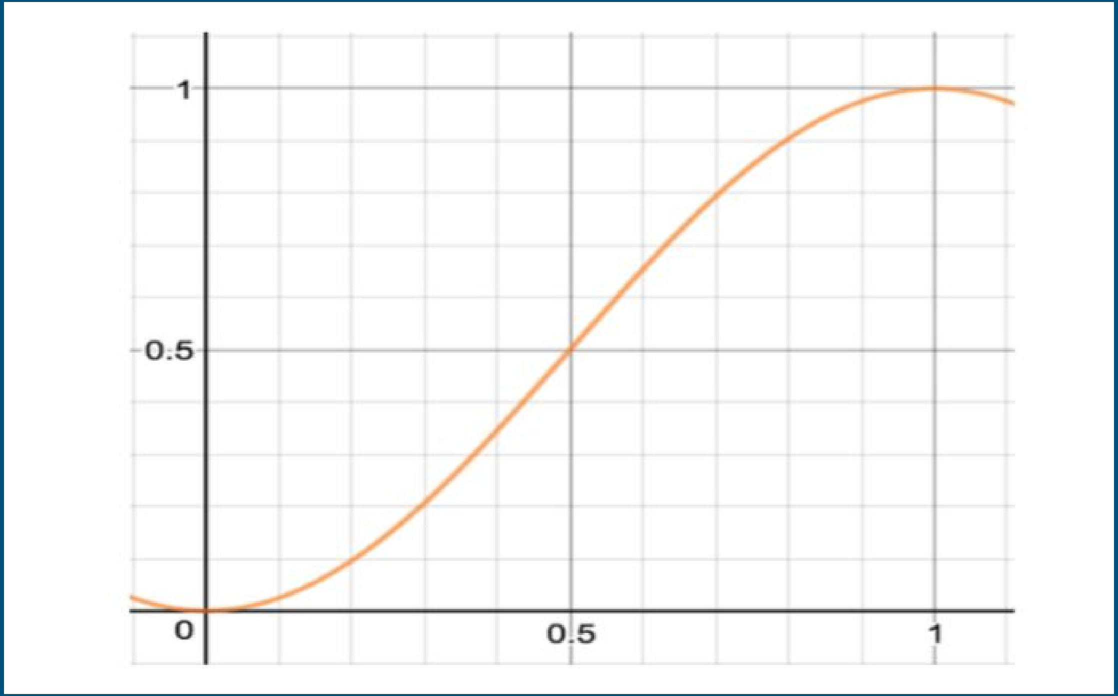


$$f(t_0, t_1, p_0, p_1, t) = p_0 + (0.5(-\cos(\pi(\frac{t - t_0}{t_1 - t_0})) + 1))(p_1 - p_0)$$

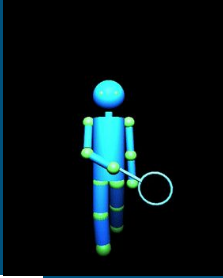


```
function cosInterpolate(startTime, endTime, startPos, endPos, currentTime) {
    if (currentTime < startTime) {
        return startPos;
    }
    if (currentTime > endTime) {
        return endPos;
    }
    var t = .5*(-m.cos(m.PI*(currentTime - startTime)/(endTime - startTime)) + 1);
    return startPos + t*(endPos - startPos);
}
```

Multiply by (0.5)

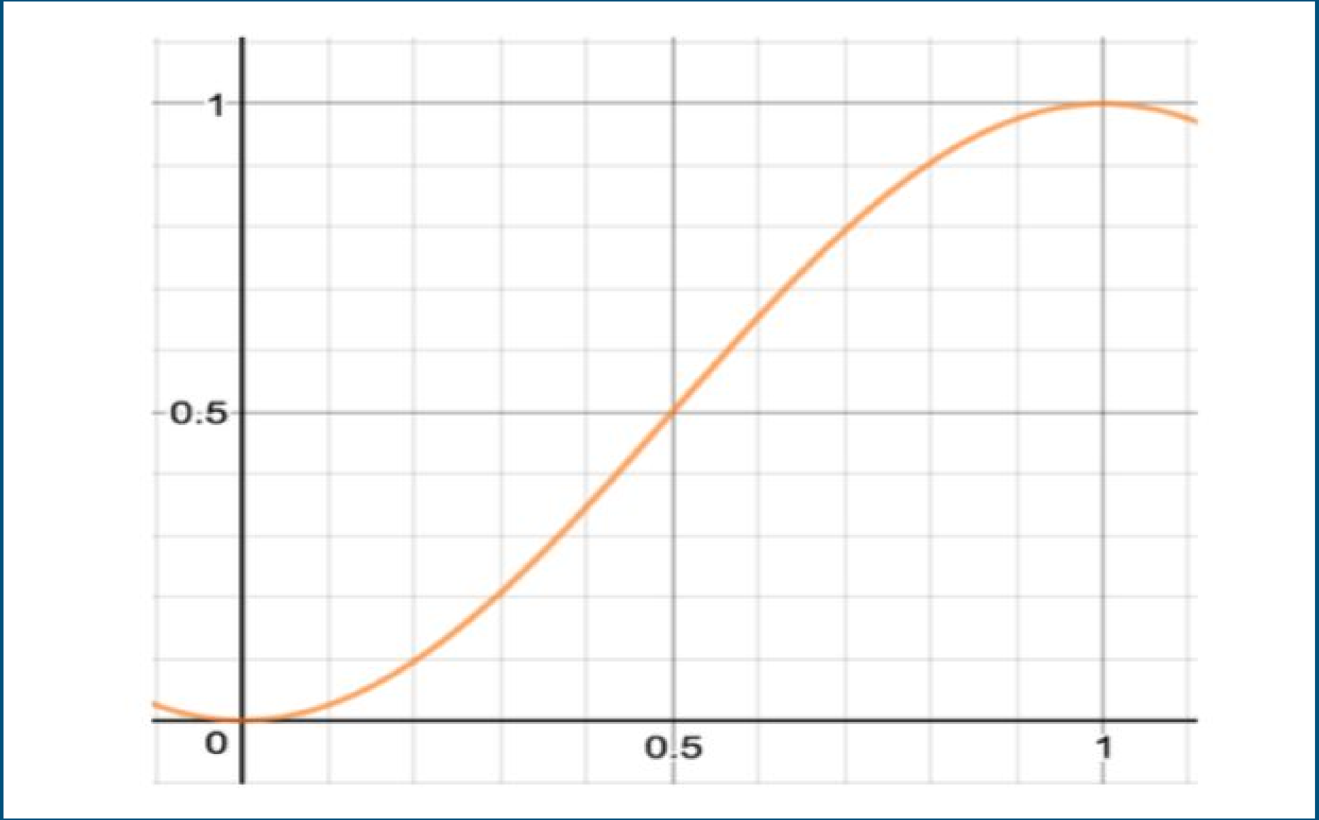


Sinusoidal Interpolation

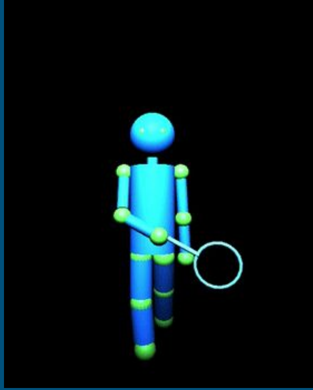


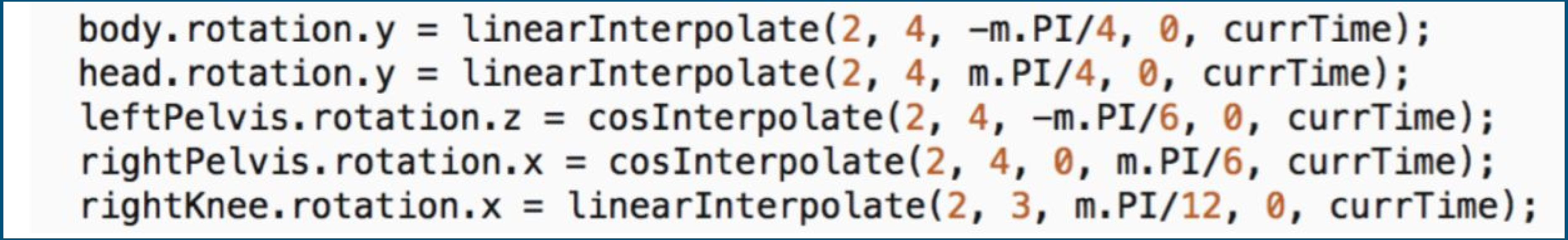
$$f(t_0, t_1, p_0, p_1, t) = p_0 + (0.5(-\cos(\pi(\frac{t - t_0}{t_1 - t_0})) + 1))(p_1 - p_0)$$

```
function cosInterpolate(startTime, endTime, startPos, endPos, currentTime) {
    if (currentTime < startTime) {
        return startPos;
    }
    if (currentTime > endTime) {
        return endPos;
    }
    var t = .5*(-m.cos(m.PI*(currentTime - startTime)/(endTime - startTime)) + 1);
    return startPos + t*(endPos - startPos);
}
```


1
0.5
0
0.5
1

Calling Interpolation Functions





```
body.rotation.y = linearInterpolate(2, 4, -m.PI/4, 0, currTime);
head.rotation.y = linearInterpolate(2, 4, m.PI/4, 0, currTime);
leftPelvis.rotation.z = cosInterpolate(2, 4, -m.PI/6, 0, currTime);
rightPelvis.rotation.x = cosInterpolate(2, 4, 0, m.PI/6, currTime);
rightKnee.rotation.x = linearInterpolate(2, 3, m.PI/12, 0, currTime);
```

Future Expansion

User-Interactive Features

Other Badminton Moves

Second Player

Game