

James Khazar Alchemy Flash Project Scripts

Movie Scripts

```

// TimeLapse Clip Scripts
//-----
// march 21, 2006
// james khazar
//
// Scripts used by the Flash movies, attached to the timeline, which
// control how the time lapse clock widget in the lower right
// corner behaves.
//
// Init variables
var theDate = new Date();
var timeToggle;
var rate = 1;
// Set the clock on startup
setTimeLapse = function (newDate) {
    var minutePos = newDate.getMinutes();
    var hourPos = newDate.getHours();
    var monthPos = newDate.getMonth() + 1;
    var datePos = newDate.getDate();
    var yearPos = newDate.getFullYear();
    this.myMinuteHand._rotation = minutePos * 6;
    this.myHourHand._rotation = (hourPos * 30) + (minutePos * (6/360));
    this.timeVar = hourPos + ":" + minutePos;
    this.dateVar = monthPos + "/" + datePos + "/" + yearPos;
    timeToggle = false;
    theDate = newDate;
}
//
setTimeLapse(theDate);
//
// Update the celestial events in timelapse fashion
// Toggled on and off by clicking on the clock's face (myClockFace)
this.onEnterFrame = function () {
    if (timeToggle) {
        // Advance the date based on the current rate
        // rate is set by myPlusButton and myMinusButton
        theMinutes = theDate.getMinutes();
        theDate.setMinutes(theMinutes + rate);
        minutePos = theDate.getMinutes();
        // Set the text displays, but first
        // deal with that pesky zero hour between
        // 12 and 1 for the text display
        if (theDate.getHours() == 0) {
            hourPos = 24;
        } else {
            hourPos = theDate.getHours();
        }
        monthPos = theDate.getMonth() + 1; // Offset for Jan = 0
        datePos = theDate.getDate();
        yearPos = theDate.getFullYear();
        timeVar = hourPos + ":" + minutePos;
        dateVar = monthPos + "/" + datePos + "/" + yearPos;
        // Set the clock hands
        this.myMinuteHand._rotation = minutePos * 6;
        this.myHourHand._rotation = (theDate.getHours() * 30) + (minutePos * 0.4);
        // Update the clips
        this._parent.mySunClip.setRotation(theDate);
        this._parent.myMoonClip.setRotation(theDate);
        this._parent.myMoonClip.setPhase(theDate);
        this._parent.mySkyClip.calcSky(theDate);
        this._parent.myHorizonClip1.setAlpha(theDate);
        this._parent.myHorizonClip2.setAlpha(theDate);
        this._parent.myStarsClip.setAlpha(theDate);
        this._parent.water1Clip.setTide(theDate);
        this._parent.water2Clip.setTide(theDate);
        this._parent.water3Clip.setTide(theDate);
        this._parent.water4Clip.setTide(theDate);
        this._parent.myTreeClip.setSeason(theDate);
    }
}
//
// Interface controls
myClockFace.onPress = function () {
    timeToggle = !timeToggle;
}
myClockFace.onRollOver = function() {

```

```

        helpClip.gotoAndStop(1);
        helpClip._x= -2;
    }
    myClockFace.onRollOut = function() {
        helpClip._x= 100;
    }
    //
    myDirButton.onPress = function() {
        rate = -rate;
        rateVar = rate;
    }
    myDirButton.onRollOver = function() {
        helpClip.gotoAndStop(4);
        helpClip._x= -2;
    }
    myDirButton.onRollOut = function() {
        helpClip._x= 100;
    }
    //
    myPlusButton.onPress = function () {
        rate = rate*2;
        rateVar = rate;
    }
    myPlusButton.onRollOver = function() {
        helpClip.gotoAndStop(2);
        helpClip._x= -2;
    }
    myPlusButton.onRollOut = function() {
        helpClip._x= 100;
    }
    //
    myMinusButton.onPress = function () {
        rate = rate/2;
        rateVar = rate;
    }
    myMinusButton.onRollOver = function() {
        helpClip.gotoAndStop(3);
        helpClip._x= -2;
    }
    myMinusButton.onRollOut = function() {
        helpClip._x= 100;
    }
}

```

```

// Set Longitude and Latitude Clip Scripts
//-----
// March 21, 2006
// James Khazar
//
// Stop the clip from playing (prevents unwanted
// animation and is considered good form.
stop();
// Init variables
this._y = Stage.height;
this.openMe._visible = 1;
this.closeMe._visible = 0;
this.popupTextClip._visible = 0;
//
// Misc. button functions
openMe.onRollOver = function() {
    this._parent.popupTextClip._visible = 1;
}
openMe.onRollOut = function() {
    this._parent.popupTextClip._visible = 0;
}
openMe.onPress = function() {
    this._parent._y = Stage.height - _root.myUserDataClip._height;
    this._visible = 0;
    this._parent.popupTextClip._visible = 0;
    this._parent.closeMe._visible = 1;
}
closeMe.onPress = function() {
    this._parent._y = Stage.height;
    this._parent.openMe._visible = 1;
    this._visible = 0;
}
//
// Set variables to defaults
var theLatitude:Number = 36.9998;
var theLongitude:Number = 122.0624;

```

```

var theEastwest:String = "West";
var theNorthsouth:String = "North";
//
// Setup Fields & restrict the input types to numbers
myLatitude.text = theLatitude;
myLatitude.restrict = "0-9 .";
myLongitude.text = theLongitude;
myLongitude.restrict = "0-9 .";
//
// Listen, Respond to focus to remove directive text
//
// Longitude
var latitudeListener:Object = new Object();
latitudeListener.handleEvent = function(event_obj:Object) {
    testMe = event_obj.type;
    if (testMe == "focusIn" && myLatitude.text == "36.9998") {
        myLatitude.text = "";
    }
}
myLatitude.addEventListener("focusIn", latitudeListener);
myLatitude.addEventListener("keyDown", latitudeListener);
//
// Latitude
var longitudeListener:Object = new Object();
longitudeListener.handleEvent = function(event_obj:Object) {
    testMe = event_obj.type;
    if (testMe == "focusIn" && myLongitude.text == "122.0624") {
        myLongitude.text = "";
    }
}
myLongitude.addEventListener("focusIn", longitudeListener);
myLongitude.addEventListener("keyDown", longitudeListener);
//
// east/west
var eastwestListener:Object = new Object();
eastwestListener.handleEvent = function(event_obj:Object) {
    theEastwest = eastwestGroup.selectedData;
}
eastwestGroup.addEventListener("click", eastwestListener);
//
// north/south
var northsouthListener:Object = new Object();
northsouthListener.handleEvent = function(event_obj:Object) {
    theNorthsouth = northsouthGroup.selectedData;
}
northsouthGroup.addEventListener("click", northsouthListener);
//
// The SET BUTTON
mySetButton.label = "Set";
mySetButton.setStyle("fontFamily", "_sans");
mySetButton.useHandCursor = true;
//
var setButtonListener:Object = new Object();
setButtonListener.handleEvent = function(event_obj:Object) {
    testMe = event_obj.type;
    if (testMe == "click") {
        doSet();
    }
}
mySetButton.addEventListener("click", setButtonListener);
//
// Store the longitude and latitude on the local hard drive
doSet = function() {
    UserData.setLongLat(myLongitude.text, theEastWest.text, myLatitude.text, theNorthSouth.text);
    this._y = Stage.height;
    this.openMe._visible = 1;
    this.closeMe._visible = 0;
}
}

```

Classes used by the movie

```

// Horizon Class
//-----
// March 15, 2006
// James Khazar
//
// Class to calculate the opacity of the horizon sunset/sunrise colors,
// which is a transparent overlay in front of the sky and clouds.

```

```

//
// Public methods: none
// Private methods: setAlpha
//
// Calls on: Sun Class
//
class Horizon extends MovieClip {
    //Declare class variables
    private var now:Date;           // An arbitrary date
    private var mySun:Object;
    //
    //-----
    // Horizon constructor
    //-----
    function Horizon(){
        now = new Date();
        mySun = new Sun();
        setAlpha(now);
    }
    //
    //-----
    // Calculate and set the Horizon's opacity
    //-----
    private function setAlpha(theDate):Void{
        var horizonOffset:Number = 15.0;
        var alpha:Number;           // Alpha is the transparency of the horizon MovieClip.
        alpha = 0;                   // 0 alpha is invisible, 100 is fully opaque.
        var theRotation = mySun.getRotation(theDate);
        if (theRotation >= 360 - horizonOffset) {
            //AM before sunrise
            alpha = 1 - ((360 - theRotation) / horizonOffset);
        }
        if (theRotation >= 0 && theRotation <= horizonOffset) {
            //AM after sunrise
            alpha = 1 - (theRotation / horizonOffset);
        }
        if (theRotation >= 180-horizonOffset && theRotation < 180) {
            //PM before sunset
            alpha = 1 - ((180 - theRotation) / horizonOffset);
        }
        if (theRotation >= 180 && theRotation <= 180+horizonOffset) {
            //PM after sunset
            alpha = (180 + horizonOffset - theRotation) / horizonOffset;
        }
        alpha = alpha * 100;
        this._alpha = alpha; // _alpha is Flash's movieclip property
                               // for transparency/opacity
    }
}

```

```

// Moon Class
//-----
// March 15, 2006
// James Khazar
//
// Class to calculate the phase of the moon based on the date and set a
// movieClip's frame number to match the phase.
//
// Public methods: getPhase
// Private methods: setPhase, setRotation, calcPhase
//
// Referenced by: no one yet, but getPhase is available...
// Calls on: Sun Class
//
class Moon extends MovieClip {
    //Declare class variables
    private var now:Date;           // An arbitrary date
    private var majorPhase:Number; // The position of the edge of the shadow on the moon
    private var minorPhase:Number; // The waxing or waning status of the moon
                                     // (positive values are waxing, negative waning)
    private var phaseArray:Array;  // An array that holds the names of the phases
                                     // moon, New, Waxing Crescent, etc.
    private var moonsSun:Object;
    //
    //-----
    // Moon constructor
    //-----
    function Moon(){
        now = new Date();
    }
}

```

```

        moonsSun = new Sun();
        setPhase(now);
        setRotation(now);
    }
    //
    //-----
    // Calculate the moon's phase
    //-----
    //
    // Allow others to get the phase frame number,
    // which is 1 to 1000 relative to new moon to new moon.
    public function getPhase(theDate:Date):Number {
        return(calcPhase(theDate));
    }
    // Go to the appropriate frame number of this movieclip
    // Frame 1 is New Moon, frame 250 is First Quarter, frame 500 is Full Moon
    // frame 750 is Last Quarter and frame 1000 is the New Moon again.
    private function setPhase(theDate:Date):Void {
        this.gotoAndStop(calcPhase(theDate));
    }
    // Get the rotation in degrees relative to sunrise and sunset from the
    // Sun class and use it to set the rotation of the Moon at 180° away from it.
    private function setRotation(dateToTest:Date):Void{
        var radPerDeg = Math.PI/180.0;
        var radius:Number = 282;
        var xOffset:Number = 400;
        var yOffset:Number =300;
        var alpha:Number = (moonsSun.getRotation(dateToTest))*radPerDeg;
        this._x = radius * Math.cos(alpha) + xOffset;
        this._y = radius * Math.sin(alpha)*0.95 + yOffset;
    }
    public function calcPhase(theDate:Date):Number {
        now = theDate;
        var Pi:Number = Math.PI;
        // This code borrowed from http://biology.clc.uc.edu/steincarter/moon/moon%20code.htm
        // By J. Stein Carter, carterjs@uc.edu Copyright © 1998. All rights reserved.
        // File moon code.htm was last modified on 14 Mar 2001.
        // Based on calculations given in the book
        // Duffett-Smith, Peter. 1988. Practical Astronomy with Your Calculator, 3rd Ed.
        // Cambridge Univ. Press.
        //
        // in min so convert to hours
        var tzone:Number = now.getTimezoneOffset()/60;
        // this gives msec
        var moonmsec:Number = now.getTime();
        // GMT in msec
        var GMtime:Number = moonmsec+(tzone*60*60*1000);
        // equivalent of 0 Jan 90
        var startDate:Date = new Date(89, 11, 31, 00, 00, 00);
        var startMsec:Number = startDate.getTime();
        var DMsec = GMtime-startMsec;
        var D:Number = (((DMsec/1000)/60)/60)/24;
        var n:Number = D*(360/365.242191);
        if (n>0) {
            n = n-Math.floor(Math.abs(n/360))*360;
        } else {
            n = n+(360+Math.floor(Math.abs(n/360))*360);
        }
        var Mo:Number = n+279.403303-282.768422;
        if (Mo<0) {
            Mo = Mo+360;
        }
        var Ec:Number = 360*.016713*Math.sin(Mo*Pi/180)/Pi;
        var lamda:Number = n+Ec+279.403303;
        if (lamda>360) {
            lamda = lamda-360;
        }
        var l:Number = 13.1763966*D+318.351648;
        if (l>0) {
            l = l-Math.floor(Math.abs(l/360))*360;
        } else {
            l = l+(360+Math.floor(Math.abs(l/360))*360);
        }
        var Mm:Number = l-.1114041*D-36.34041;
        if (Mm>0) {
            Mm = Mm-Math.floor(Math.abs(Mm/360))*360;
        } else {
            Mm = Mm+(360+Math.floor(Math.abs(Mm/360))*360);
        }
        var N65:Number = 318.510107-.0529539*D;
        if (N65>0) {
            N65 = N65-Math.floor(Math.abs(N65/360))*360;
        } else {
            N65 = N65+(360+Math.floor(Math.abs(N65/360))*360);
        }
        var Ev:Number = 1.2739*Math.sin((2*(1-lamda)-Mm)*Pi/180);
        var Ae:Number = .1858*Math.sin(Mo*Pi/180);
    }

```

```

var A3:Number = .37*Math.sin(Mo*Pi/180);
var Mmp:Number = Mm+Ev-Ae-A3;
var Ec:Number = 6.2886*Math.sin(Mmp*Pi/180);
var A4:Number = .214*Math.sin((2*Mmp)*Pi/180);
var lp:Number = l+Ev+Ec-Ae+A4;
var V:Number = .6583*Math.sin((2*(lp-lamda))*Pi/180);
var lpp:Number = lp+V;
var D67:Number = lpp-lamda;
majorPhase = .5*(1-Math.cos(D67*Pi/180));
minorPhase = (Math.sin(D67*Pi/180));
// Convert data to a number from 1-1000
majorPhase = Math.floor(majorPhase*500);
// If minorPhase is a negative number, then the moon is waning.
// So invert majorPhase from (1 to 500) to (501 to 1000)
if (minorPhase < 0) {
    majorPhase = (majorPhase-1000)*-1;
}
// Set the clip's frame number to correspond to the phase
return(majorPhase);
    }
}

```

```

// Seasons Class
//-----
// March 15, 2006
// James Khazar
//
// Class to change the frame of a movieClip to correspond with the day of
// the year. Stub for later version which will get the true solistice and equinoxes
// for a given year.
//
// Public methods: none
// Private methods: setSeason, getDayOfYear
//
// Stands alone, no calls or outside references.
//
class Seasons extends MovieClip {
    //Declare class variables
    private var now:Date;           // An arbitrary date
    //
    //-----
    // Seasons constructor
    //-----
    function Seasons(){
        now = new Date();
        setSeason(now);
    }
    //
    //-----
    // Calculate and set the season
    //-----
    // The seasons are done as a frame in the movieClip corresponding
    // to a day of the year.
    private function setSeason(theDate):Void{
        var days = getDayOfYear(theDate);
        this.gotoAndStop(days);
    }
    //
    private function getDayOfYear(theDate):Number {
        var monthLengthArray:Array = new Array(31,28,31,30,31,30,31,31,30,31,30,31);
        var totalMonthDays:Number = 0;
        for (var n = 0; n <= theDate.getMonth(); n++) {
            totalMonthDays = totalMonthDays + monthLengthArray[n];
        }
        var totalDays:Number
        totalDays = totalMonthDays + theDate.getDate();
        return(totalDays);
    }
}

```

```

// Sky Class

```

```

//-----
// March 15, 2006
// James Khazar
//
// Class to set the color of the sky based on the sunset and sunrise
//
// Public methods: calcSky
// Private methods: rgbToHex, getLuminance
//
// Referenced by: Stars Class
// Calls on: Sun Class
//
class Sky extends MovieClip {
    //Declare class variables
    private var now:Date;           // An arbitrary date
    private var mySun:Object;
    private var skyColor:Color;    // Flash uses a bizarro color object
    //
    //
    //
    //
    //
    //-----
    // Sky constructor
    //-----
    function Sky(){
        now = new Date();
        mySun = new Sun();
        skyColor = new Color(this);
        calcSky(now);
    }
    //
    //-----
    // Calculate and set the sky's color
    //-----
    //
    // Get the rotation in degrees relative to sunrise and sunset from the
    // Sun class and use it to set the rotation of the Moon at 180° away from it.
    // Returns the angle of rotation.
    public function calcSky(theDate):Number{
        var colorStep:Number;
        var r:Number;
        var g:Number;
        var b:Number;
        var theRotation = mySun.getRotation(theDate);
        if (theRotation >= 270) {
            // AM night r @ 0, g @ 0, b from 32 to 255
            theRotation = theRotation - 270;
            if (theRotation >= 45) {
                colorStep = 1 - getLuminance(theRotation);
                r = 0;
                g = 0;
                b = (223*colorStep)+32;
                skyColor.setRGB(rgbToHex(r, g, b));
            }else{
                colorStep = 0;
                skyColor.setRGB(rgbToHex(0,0,32));
            }
            return(1 - colorStep);
        } else if (theRotation >= 180) {
            // PM night r @ 0, g @ 0, b from 255 to 32
            theRotation = theRotation - 180;
            if (theRotation <= 45) {
                colorStep = 1 - getLuminance(theRotation);
                r = 0;
                g = 0;
                b = (223*colorStep)+32;
                skyColor.setRGB(rgbToHex(r, g, b));
            }else{
                colorStep = 0;
                skyColor.setRGB(rgbToHex(0,0,32));
            }
            return(1 - colorStep);
        } else if (theRotation >= 90) {
            // PM day r from 212 to 0, g from 250 to 0, b @ 255
            theRotation = theRotation - 90;
            if (theRotation >= 45) {
                colorStep = getLuminance(theRotation);
                r = 212 * colorStep;
                g = 250 * colorStep;
                b = 255;
                skyColor.setRGB(rgbToHex(r, g, b));
            }else{
                colorStep = 0;
                skyColor.setRGB(rgbToHex(212, 250, 255));
            }
        }
    }
}

```

```

        return(0);
    } else {
        // AM day r from 0 to 212, g from 0 to 250, b @ 255
        if (theRotation <= 45) {
            colorStep = getLuminance(theRotation);
            r = 212 * colorStep;
            g = 250 * colorStep;
            b = 255;
            skyColor.setRGB(rgbToHex(r, g, b));
        }else{
            colorStep = 0
            skyColor.setRGB(rgbToHex(212, 250, 255));
        }
        return(0);
    }
}
// RGB manipulations
private function rgbToHex(r:Number,g:Number,b:Number):Number {
return << 16 | g << 8 | b };
}
// Calculate a parabola
private function getLuminance(position:Number):Number {
var luminance:Number;
luminance = ( (-1/2025) * position*position ) + ( (2/45) * position );
return(luminance);
}
}
}

```

```

// Stars Class
//-----
// March 15, 2006
// James Khazar
//
// Class to fade in and rotate the star field behind the moon. Stub for
// something more accurate in the future.
//
// Public methods: none
// Private methods: setAlpha
//
// Calls on: Sun Class, Sky Class
//
class Stars extends MovieClip {
//Declare class variables
private var now:Date; // An arbitrary date
private var mySun:Object;
private var mySky:Object;
//
//-----
// Stars constructor
//-----
function Stars(){
now = new Date();
mySky = new Sky();
mySun = new Sun();
setAlpha(now);
}
//
//-----
// Calculate and set the stars opacity
//-----
//
private function setAlpha(theDate):Void{
this._alpha = mySky.calcSky(theDate)*100;
this._rotation = mySun.getRotation(theDate)/2;
}
}
}

```

```

// Sun Class
//-----

```

```

// March 15, 2006
// James Khazar
//
// Class to locate the sun movieClip according to the longitude, latitude
// and local time.
//
// Public methods: setRotation, getRotation, getSunriseSunset
// Private methods: getDayNumber, siderealTime, rev, sinDegree, cosDegree,
//                 tanDegree, aSinDegree, aCosDegree, atanDegree
//
// Referenced by: Stars Class, Sky Class
// Calls on: UserData class
//
class Sun extends MovieClip {
    var theDate:Date; // An arbitrary date
    var theLongitude:Number;
    var theLatitude:Number;
    var myUserData:UserData; // UserData is a custom kind class which stores data
    // on the user's hard drive.
    //
    //-----
    // Sun constructor
    //-----
    function Sun() {
        myUserData = new UserData;
        theLongitude = UserData.getLongitude();
        theLatitude = UserData.getLatitude();
        theDate = new Date();
        setRotation(theDate);
    }
    //
    //-----
    // Determine and set the rotational angle of the sun
    //-----
    // Get the rotational angle and set the sun's position.
    // Called on by timelapse movieClip to simulate accelerated time.
    public function setRotation(dateToTest:Date):Void {
        var radius:Number = 282;
        var xOffset:Number = 400;
        var yOffset:Number = 300;
        var alpha:Number = (getRotation(dateToTest)-180)*RadPerDeg;
        this._x = radius * Math.cos(alpha) + xOffset;
        this._y = radius * Math.sin(alpha)*0.85 + yOffset;
    }
    // Calculate the rotational angle
    // Used by this and other classes:
    public function getRotation(dateToTest:Date):Number{
        //Set the rotation of the clip this controls.
        var sunriseSunsetArray:Array = getSunriseSunset(dateToTest);
        var sunrise:Number = sunriseSunsetArray[0];
        var sunset:Number = sunriseSunsetArray[1];
        var dayLength:Number = sunriseSunsetArray[2];
        var nightLength:Number = 1440 - dayLength;
        // current time in minutes since midnight
        var currentTime:Number = (dateToTest.getHours()*60) + dateToTest.getMinutes();
        /*
        // UNCOMMENT TO CHECK VARS
        trace("sunrise: " + sunrise);
        trace("sunset: " + sunset);
        trace("dayLength: " + dayLength);
        trace("currentTime: " + currentTime);
        */
        var currentRot:Number;
        if (currentTime >= sunrise && currentTime <= sunset) {
            // It's daytime
            currentRot = (180*(currentTime - sunrise))/dayLength;
        }else{
            // It's nighttime
            if (currentTime > sunset) {
                // It's before midnight
                var SStoMid = 1440 - sunset;
                currentRot = ((90*(currentTime - 1440)) / SStoMid) + 270;
            } else {
                // It's after midnight
                currentRot = ((90*(currentTime - sunrise))/ sunrise) + 360;
            }
        }
        return(currentRot);
    }
    //-----
    // Solar calculations
    //-----
    // JavaScript by Peter Hayes http://www.peter - hayes.freemove.co.uk/
    // Copyright 2001 - 2003
    // This code is made freely available but please keep this notice.
    // The calculations are approximate but should be good enough for general use,
    // Mr Hayes accepts no responsibility for errors in astronomy or coding.

```

```

//
public function getSunriseSunset(dateToTest:Date):Array {
    if (theLongitude == undefined) {
        theLongitude = UserData.getLongitude();
        theLatitude = UserData.getLatitude();
    }
    // Based on method in sci.astro FAQ by Paul Schlyter
    // returns an array holding rise and set times in UTC hours
    // NOT accurate at high latitudes
    //
    // Calculate the Suns position at noon local zone time
    var dayNumber = getDayNumber(theDate, theLongitude );
    var oblecl = 23.4393 - (3.563E-7) * dayNumber;
    var w = 282.9404 + (4.70935E-5) * dayNumber;
    var M = 356.0470 + 0.9856002585 * dayNumber;
    var e = 0.016709 - (1.151E-9) * dayNumber;
    var E = M + e * (180 / Math.PI) * sinDegree(M) * (1.0 + e * cosDegree(M));
    var A = cosDegree(E) - e;
    var B = Math.sqrt(1 - e * e) * sinDegree(E);
    var slon = w + atanDegree(B, A);
    var sRA = atanDegree(sinDegree(slon) * cosDegree(oblecl), cosDegree(slon));
    while (sRA<0) {
        sRA += 360;
    }
    while (sRA>360) {
        sRA -= 360;
    }
    sRA = sRA/15;
    var sDec = aSinDegree(sinDegree(oblecl) * sinDegree(slon));
    // The time when the sun is on the meridian
    var lst = siderealTime(theDate, 12 - theLongitude/15, theLongitude);
    var MT = 12.0-theLongitude/15+sRA-lst;
    while (MT<0) {
        MT += 24;
    }
    while (MT>24) {
        MT -= 24;
    }
    // The hour angle
    var cHA0 = ( sinDegree( - 0.8333333) - sinDegree(theLatitude) * sinDegree(sDec) )
              / ( cosDegree(theLatitude) * cosDegree(sDec) );
    var HA0 = aCosDegree(cHA0);
    HA0 = rev(HA0)/15;
    // The sunset and sunrise times in minutes from midnight
    var sunriseMinutes = ((MT - HA0)*60) + theDate.getTimezoneOffset();
    var sunsetMinutes = ((MT + HA0)*60) + theDate.getTimezoneOffset();
    var dayLength = sunsetMinutes - sunriseMinutes;
    var nightLength = sunriseMinutes + (1440 - sunsetMinutes);
    /*
    // UNCOMMENT TO CHECK VARS
    trace("dayNumber: " + dayNumber );
    trace("oblecl: " + oblecl );
    trace("w: " + w );
    trace("M: " + M );
    trace("e: " + e );
    trace("E: " + E );
    trace("A: " + A );
    trace("B: " + B );
    trace("slon: " + slon );
    trace("sRA: " + sRA );
    trace("lst: " + lst );
    trace("MT: " + MT );
    trace("cHA0: " + cHA0);
    trace("HA0: " + HA0);
    trace("sunriseMinutes: " + sunriseMinutes);
    trace("sunsetMinutes: " + sunsetMinutes);
    */
    // Return rise, set and day length times in minutes from midnight
    return new Array(sunriseMinutes, sunsetMinutes, dayLength, nightLength );
}
//
//-----
// Utilities
//-----
private function getDayNumber(theDate, hours) {
    // Day number is a modified Julian date, day 0 is 2000 January 0.0
    // which corresponds to a Julian date of 2451543.5
    var d = 367 * theDate.getFullYear()
          - Math.floor(7 * (theDate.getFullYear() + Math.floor(((theDate.getMonth()+1) + 9)/12))/4)
          + Math.floor((275 * (theDate.getMonth()+1))/9)
          + theDate.getDate() - 730530 + hours/24;

    return d;
}
// Radian to Degree to Radian converters
private var RadPerDeg = Math.PI/180.0;
private var DegPerRad = 180.0/Math.PI;
//

```

```

private function siderealTime(theDate, hours, lon) {
    // Compute local sidereal time in degrees
    // year, month, day and hours are the Greenwich date and time
    // lon is the observers longitude
    var d = getDayNumber(theDate, hours);
    var lst = (98.9818 + 0.985647352 * d + hours * 15 + lon);
    return rev(lst)/15;
}
// Various math functions
private function rev(deg):Number {
    return deg - Math.floor(deg/360.0) * 360.0;
}
private function sinDegree(deg):Number {
    return Math.sin(deg * RadPerDeg);
}
private function cosDegree(deg):Number {
    return Math.cos(deg * RadPerDeg);
}
private function tanDegree(deg):Number {
    return Math.tan(deg * RadPerDeg);
}
private function aSinDegree(c):Number {
    return Math.asin(c) * DegPerRad;
}
private function aCosDegree(c):Number {
    return Math.acos(c) * DegPerRad;
}
private function atanDegree(y, x):Number {
    return Math.atan(y/x) * DegPerRad - 180.0 * (x<0);
}
}

```

```

// Tide Class
//-----
// March 15, 2006
// James Khazar
//
// Stub class to move the tide's up and down. Linked to multiple movieClips,
// it uses periods of 400 minutes for the tidal ebb and flow and moves the
// movieClips 33 pixels up and down accordingly.
//
// Public methods: none
// Private methods: setTide, getJulianMinutes
//
// Stands alone, no calls or outside references.
//
class Tide extends MovieClip {
    //Declare class variables
    private var now:Date;           // An arbitrary date
    private var myY:Number;
    private var tideDirection:Boolean;
    private var tideOffset:Number;
    private var oldOffset:Number;
    //-----
    // Tide constructor
    //-----
    function Tide(){
        now = new Date();
        myY = this._y;
        tideDirection = true;
        oldOffset = 0;
        tideOffset = 0;
        setTide(now);
    }
    // Stub animation control moves tide up and down 33 pixels every 400 minutes.
    // Called on by timelapse movieClip to simulate accelerated time.
    private function setTide(theDate:Date) {
        var jNow = getJulianMinutes(theDate);
        oldOffset = tideOffset
        tideOffset = ( ( jNow/400) - Math.floor(jNow/400) ) * 100 ) / 3 ;
        if (tideOffset < oldOffset) {
            tideDirection = !tideDirection;
        }
        if (tideDirection) {
            this._y = myY + tideOffset;
        } else {
            this._y = myY + 33 - tideOffset;
        }
    }
}

```

```

    }
    // Returns the number of minutes from midnight, January 1st, 2000.
    private function getJulianMinutes(theDate) {
        // Day number is a modified Julian date, day 0 is 2000 January 0.0
        // which corresponds to a Julian date of 2451543.5
        // JavaScript borrowed from Peter Hayes http://www.peter-hayes.freereserve.co.uk/
        // Copyright 2001 - 2003 (use permitted with attribution).
        var hours = theDate.getHours();
        var minutes = (hours * 60) + theDate.getMinutes()
        var julianMinutes = ((367 * theDate.getFullYear()
            - Math.floor(7 * (theDate.getFullYear() + Math.floor((theDate.getMonth()+1) + 9)/12))/4)
            + Math.floor((275 * (theDate.getMonth()+1))/9)
            + theDate.getDate() - 730530) * 1440) + minutes;

        return julianMinutes;
    }
}

```

```

// UserData Class
//-----
// March 15, 2006
// James Khazar
//
// Class to store user data about their longitude and latitude
// Referenced by a movieClip interface with fields to capture entered data
// on longitude and latitude, which it stores to the user's hard drive.
//
// Public methods: setLongLat, getLongitude, getLatitude
//
// Referenced by: Sun class
//
class UserData {
    // SharedObject is a Flash thingy that stores simple variables on the
    // local user's hard drive.
    static var longLat:SharedObject;
    //
    //-----
    // Constructor
    //-----
    function UserData(){
        longLat = SharedObject.getLocal("userData");
        // Set default Longitude and Latitude to Santa Cruz coords
        // if longLat has never been set before
        if (longLat.data.theLongitude == undefined || longLat.data.theLatitude == undefined) {
            longLat.data.theLongitude = new Number();
            longLat.data.theLongitude = 122.0624;
            longLat.data.theLatitude = new Number();
            longLat.data.theLatitude = 36.9998;
        }
    }
    //
    // Set data function
    public static function setLongLat(myLong:Number, myEastWest:String, myLat:Number, myNorthSouth:String):Void {
        // Set the incoming longitude and latitude to a positive number in case
        // the user has put in a minus sign. It should only be done by setting
        // the east/west north/south radio buttons
        myLong = Math.abs(myLong);
        myLat = Math.abs(myLat);
        // Send the location data to the SharedObject, "longLat."
        if (myEastWest == "East") {
            myLong = -myLong;
        }
        longLat.data.theLongitude = new Number();
        longLat.data.theLongitude = myLong;
        if (myNorthSouth == "South") {
            myLat = -myLat;
        }
        longLat.data.theLatitude = new Number();
        longLat.data.theLatitude = myLat;
    }
    //
    // Retrieve data functions
    public static function getLongitude():Number {
        return(longLat.data.theLongitude);
    }
    public static function getLatitude():Number {
        return(longLat.data.theLatitude);
    }
}

```

```
}
```