

# Units of Measure: Ice Melting Lab

---



Percent of Ice Melted:

Recalculate Water Level

CALCULATE

The image shows the Statue of Liberty with a semi-transparent blue overlay representing ice. The overlay covers the statue and extends to the water level. The background is a clear blue sky with some clouds. The simulation interface includes a text input field for 'Percent of Ice Melted' containing the value '100', an orange button labeled 'Recalculate Water Level', and a vertical black bar on the right side with the word 'CALCULATE' written vertically in white.

**Contents**

Overview ..... 3

Starting Out..... 4

## Overview

Mathematicians and scientists, in addition to everyone who reads almost any account of the events of nature or the economy encounter units of measure. One of the primary challenges of chemistry and physics is to become comfortable with such terms as joule, hertz, mole, watt, and so on. Such terms represent units of measure.

This application explores units of measure by simulating one of the most important findings of current environmental science: climate changes. NASA has confirmed that the ice caps are melting. What will this mean in terms of the level of seawater in the oceans?

## Application

This application uses images of a popular landmark to show how the landscape might change with rising sea levels. This application shows the total projected rise as 74 meters (m), but it is designed to show gradations also. For example, what would 10%, 20% or 40% look like?

## Starting Out

1. Create a new directory called **IceMelting**. Download Starter\_Lab and copy the contents in the folder that you just created.
2. Open **StarterIceMelting.fla** in Adobe Flash CS4 and save it as **IceMelting\_01.fla** in your main project directory.

## Preparing the Artwork

The first thing to do is to personalize the graphics of the application. The first view you have of the application shows a picture of the Statue of Liberty. Take a minute to look through the items in the library. The main piece that we will be updating is the *Calculations* movie clip. We will also adding a number of items to the stage, like a button and a text field.

## Stage Graphics

1. Add 2 text fields (the symbol for text fields is T on the toolbar) to the stage and arrange them as shown in the following figure:
2. The first text field is a static field that tells the user to enter a percent.
3. The second text field is a input field that allows the user to input a percent. Name this text field *txtPercentMelted*.
4. Next we need to create a button that will make the water level adjust to show how much ice melted.

## Button

Buttons are one of the most important features in Flash applications. They are used in website navigation, game menus, simulation controls and lots of other places on the web. This section will teach you how to create a button.

1. In the library, right click and choose *New Symbol...*
2. Name the symbol *Recalculate*
3. Choose Button for the Type
4. Click the check box in front of "Export for ActionScript." If you do not see this option, click the Advanced Button to get more options. This allows you to access this button in the ActionScript code which we will be writing in a few more steps. The name that you use to access the symbol is the one that we specified in step 2.
5. Once you click OK, the main window will be blank. In the upper left corner you will see there is a light blue arrow, then the text *Scene 1*, followed by *Recalculate*. Clicking the blue arrow will

take you back to *Scene 1* where we started. You can get back into editing mode of the button by clicking on *Recalculate* in the library on the right side of the screen.

6. In the window at the bottom of the application click the *Timeline* tab. If you do not see this you can open it from the menu at *Window -> Timeline* or by using the keyboard shortcut *CTRL + ALT + T*
7. In the timeline you will see that there are four frames: **Up, Over, Down, Hit**. These represent the four states a button can be in. When the button is not being interacted with (no mouse over, or mouse down on it) it is in the **Up** frame. When the mouse is over the button it is in the **Over** frame. When the user clicks on the button, it is in the **Down** frame. Finally the **Hit** frame is used to tell flash what can be clicked on. In most situations we want this to be the same graphic as the rest of the button, so that any part that is clicked on fires the clicked event.
8. The easiest way to start is to select Frame 1 (**Up**), in the timeline and draw a shape. I like to use the **Rectangle Primitive Tool** or the **Circle Primitive Tool**. These allow me to adjust the inside color and edge color after initially drawing the shape and also let me resize the shape as needed.
9. If you use the **Rectangle Primitive Tool** then you can also adjust the roundness of the corners by clicking on the *Properties* tab and the moving the slider at the bottom in the *Rectangle Options* section. You can also adjust the color, stroke and a few other properties of the shape.
10. Once you have the graphic completed for the first frame we want right click in Frame 2 (**Over**) and choose *Insert Keyframe*. This will copy over the shape from the previous frame in the same location so that you can modify it. If you would rather start a new shape, choose the *Insert Blank Keyframe* option on the right click menu.
11. Next we will repeat the previous step for Frame 3 (**Down**)
12. When all this is finished we want to insert a keyframe in Frame 4 (**Hit**). We will leave this the way it is so that the button can be clicked correctly.
13. Next we want to add text to the button, so create a new layer using the icon of a page turning in the lower left corner of the screen in the *Timeline* tab. It will automatically add the frames for in the timeline. You can add a new static text field to the shape and put in text indicating that this button is used to recalculate the sea level.
14. If you want the text to change in any way (color, size, ...) you can add keyframes into the other frames and adjust the text fields properties.
15. Now that the button graphics are finished we want to return the stage. Do this by clicking on *Scene 1* or the blue arrow.

16. Switch to the *Library* tab and drag a copy of the Recalculate button to the stage. Position it next to the text fields we created earlier.
17. With the button selected, switch to the *Properties* tab. There is a text field that currently says *<Instance Name>*. Change this to *btnRecalculate* (using a prefix before a variable name can be really helpful later when using the variable to know what type it is)

## Save as Version 2

1. Run the game by pressing *CTRL + Enter*, or from the menu under *Control->Play*
2. Save your work as *IceMelting\_02.fla*

## Working on the ActionScript Code

Now that the graphics have been developed, we can work on the code that is behind the application.

1. In the main fla file, frame one, select the actions window
2. Copy in the following code:

```
import fl.transitions.*;
import fl.transitions.easing.*;

// global variables
var baseHeight:int = 3;           // Number of meters from the ocean
                                   // to the base of the statue
var libertyPixels:int = 332;     // Height in pixels
var libertyHeight:int = 93;     // Height in meters (inc. base)

var myTween:Tween;

var waterLevel:Water = new Water();
waterLevel.y = this.height - 12;

addChildAt(waterLevel, 2);

// Add a listener to recalculate the height of the ocean
btnRecalculate.addEventListener(MouseEvent.CLICK, recalculateHeight);

function recalculateHeight(evt:MouseEvent) : void {

    trace("Recalculate Water Level");
```

```
}
```

3. The first two import statements are used to access the Tween class, which allows us to move objects on the stage smoothly as if we were animating them in the timeline.

The global variables are used to later in the calculations for the height of the water given the amount of ice melted. We need the actual height of the statue of liberty (93 m) and the height in pixels (332 px) so that we can scale the height of the water correctly.

The `waterLevel` variable defines an instance of the class `Water`, which we will not be explaining in this project. If you would like to learn more about how it works, look in the `Water.as`, `Wave.as`, and `Dot.as` files.

Finally the `AddChildAt()` adds the water to the stage so that it is visible.

4. We now want to add an event listener to the button so that when the user clicks it, the water level will be recalculated. Add the following line of code after the `addChildAt()`. The first parameter is the type of event to listen for, in this case, the mouse clicked event. The second parameter is the function that will be called when the mouse clicks on the button.

```
// Add a listener to recalculate the height of the ocean  
btnRecalculate.addEventListener(MouseEvent.CLICK, recalculateHeight);
```

5. Next we need to add code for what will happen when the mouse clicks on the object. Add the following code:

```
function recalculateHeight(evt:MouseEvent) : void {  
  
    trace("Recalculate Water Level");  
  
    var percent:int = int(txtPercentMelted.text);  
  
    // Make sure that the given percent is valid  
    if (percent < 0 || percent > 100)  
        return;  
  
    // Calculate the height of the water given the percentage of ice  
    // melted and the base height.  
    var newHeight:Number = ((74.0 * percent / 100.0) / libertyHeight) * libertyPixels +  
baseHeight;  
  
    myTween = new Tween(waterLevel, "y", Regular.easeOut, waterLevel.y, 400 -  
newHeight, 5, true);  
}
```

6. First we read in the percent from the text filed using the *txtPercentMelted.text* property. To see what properties an object has you can visit the [ActionScript Live Docs](#) online and look up the object.

We calculate the new height of the water by getting the sea level change from the percent of 74 meters, which is the predicted change if 100% of the ice melts. We divide this by the Statue of Liberty height, to find out what the percent of the Statue of Liberty that would be covered if the ice melted. Once we multiply this by the Height of the Statue of Liberty in pixels we know how much the height of the water would change in pixels. We add an offset because the Statue of Liberty base is not located at the bottom of the stage.

Next we create a tween that will change the height of the `waterLevel` object over 5 seconds using an easing rule. This means that the motion will start fast, but slow down as it gets to the desired value.

7. Test the application to see how it works.

## Further Explorations

For those who are interested in understanding this simulation further can take a look at the `Water.as`, `Wave.as`, and `Dot.as` files. These contain some interesting functions for modeling water by using a number of points where the “waves” are and calculating the heights of these points based on the motion of the water.

## Code Expansion

Here are some tasks you can try to expand the code if you are interested. They are put in order of increasing difficulty.

1. Adjust the speed in which the water level moves to its new height.
2. Change the color of the water
3. Change the background graphic and refactor the equations so that they accurately show how high the water level would be with respect to the new objects.