

CGT 353: Principles of Interactive and Dynamic Media Text in Flash

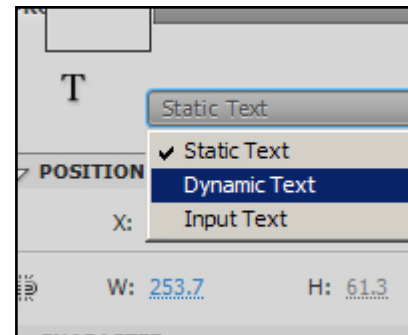
Text in Flash:

- Obviously, many of the methods used to incorporate and modify text in Flash are universal to computer applications (aligning, kerning, etc).
- As such, it is up to you to learn these nuances by reviewing the materials and practicing with Flash.



Types of Text in Flash:

1. **Static text** - type where content does not change and is subdivided into **labels** (single line) and **blocks** (multiline)
2. **Dynamic Text** - text that changes based on actions and variables in the movie
3. **Input text** - text objects that can change like dynamic text fields but can also be manipulated by the end user at runtime

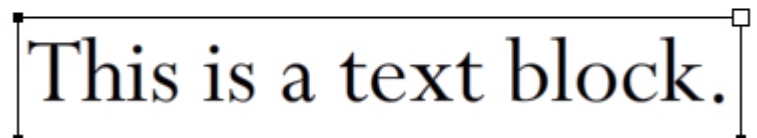
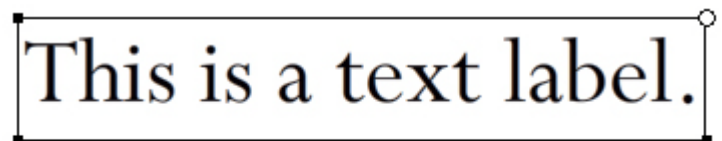


Using the Text Tool:

- The properties panel permits you the most control for formatting various text objects
- The basics of formatting text (aligning, kerning, etc) will be left up to the student to learn, although you will still be responsible for this information

Entering Text Labels and Blocks:

- **Text labels are indicated by a circle, where blocks are indicated by a square.**
- Labels made by just clicking, where blocks are made by clicking, then dragging.



- Besides being single or multiline, the biggest difference between labels and blocks is in accessibility.
- **Note:** If a nonvisual browser or screen reader is viewing the Flash movie, it will "read" a label and assume it is an indicator of an input field or something else to that effect
- Text blocks are not read in this manner.

Modifying Labels and Blocks:

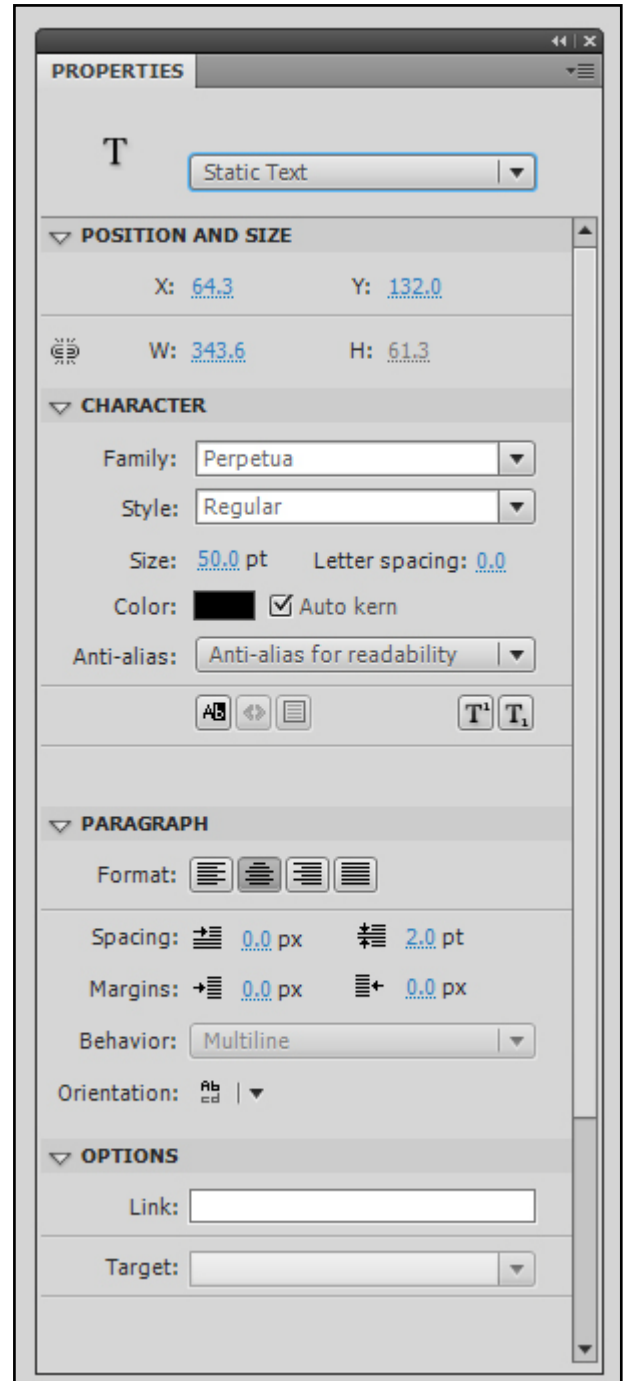
- Most parameters in the lower part of the property inspector only apply to the block level.
- For example, you can only assign a URL to an entire block, not specific text within the block.

Assigning a Hyperlink:

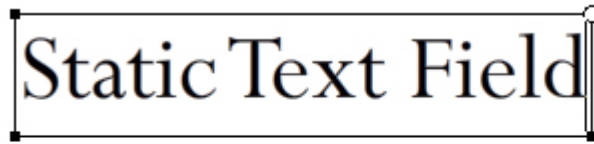
- Puts a dashed line beneath the text in .fla file, but doesn't display typical link formatting as you would see with HTML links.
- **Note:** If you were to put a linked object inside a button, the link will not work, so most times you should just script the intended link with ActionScript.
- Always put the full URL in the link.

Dynamic Text:

- Dynamic text fields are essentially just variable containers and function almost the exact same way as form elements in HTML.
- Can receive information from a variety of elements, both inside and external to the Flash movie



- Properties are essentially the same as static text, although the circle/ square appears at the bottom right instead of top right.
- Differences in property inspector are indicative of dynamic text, including instance name, variable name, and multiline button



Static Text Field

The diagram shows a rectangular box with the text "Static Text Field" inside. A small circle is located at the top right corner of the box, indicating a property inspector handle.



Dynamic Text Field

The diagram shows a rectangular box with the text "Dynamic Text Field" inside. A small square is located at the bottom right corner of the box, indicating a multiline button.

Input Text:

- Again, difference in this type of text is that the end user can modify it
- Input text cannot also have a URL assigned to it.

Manipulating Text Objects:

- Can manipulate and transform just like any other object
- Breaking text apart (already demonstrated) significantly increases file size, especially when you convert the text to fill objects

Common Tasks for Working with Text/ TextField Class:

- Modifying text field contents
- Using HTML in text fields
- Using images in text fields
- Selecting text and working with user-selected text
- Capturing text input
- Restricting text input
- Applying formatting and CSS styles to text
- Fine-tuning text display with sharpness, thickness, and anti-aliasing
- Accessing and working with static text fields from ActionScript
- The following common tasks are covered for the Flash Text Engine classes:
- Creating and displaying text
- Handling events
- Formatting text
- Working with fonts
- Controlling text

Modifying Contents:

Assign String Directly:

```
myTextField.text = "Hello World";
```

or

Assign the text property a value from a variable defined in your script:

```
package
{
    import flash.display.Sprite;
    import flash.text.*;

    public class TextWithImage extends Sprite
    {
        private var myTextBox:TextField = new TextField();
        private var myText:String = "Hello World";

        public function TextWithImage()
        {
            addChild(myTextBox);
            myTextBox.text = myText;
        }
    }
}
```

You can also assign the text property a value from a remote variable, for which you have three options in 3.0:

- **flash.net.URLLoader/ flash.net.URLRequest** classes - load variables for the text from a local or remote location.
- **FlashVars** attribute - embedded in the HTML page hosting the SWF file and can contain values for text variables.
- **flash.net.SharedObject** class - manages persistent storage of values.

HTML Text Support:

- `flash.text.TextField` class has an `htmlText` property used to identify your text string as one containing HTML tags for formatting the content.

```
var myText:String = "<p>This is <b>some</b> content to <i>render</i> as  
<u>HTML</u> text.</p>";  
myTextBox.htmlText = myText;
```

- **Flash can support a limited number of HTML tags, including:**
 - `<a>` - anchor
 - `` - bold
 - ``
 - `<i>` - italic
 - `<p>` - paragraph (must close)
 - `<u>` - underline

Using Images in Text Fields:

- Another advantage to displaying your content as HTML text is that you can include images in the text field.
- You can reference an image, local or remote, using the `img` tag and have it appear within the associated text field.
- The `img` tag supports JPEG, GIF, PNG, and SWF files.

```
package  
{  
    import flash.display.Sprite;  
    import flash.text.*;  
  
    public class TextWithImage extends Sprite  
    {  
        private var myTextBox:TextField;  
        private var myText:String = "<p>This is <b>some</b> content to  
<i>test</i> and <i>see</i></p><p><img src='eye.jpg' width='20'  
height='20'></p><p>what can be rendered.</p><p>You should see an eye image and  
some <u>HTML</u> text.</p>";
```

```

public function TextWithImage()
{
    myTextBox.width = 200;
    myTextBox.height = 200;
    myTextBox.multiline = true;
    myTextBox.wordWrap = true;
    myTextBox.border = true;

    addChild(myTextBox);
    myTextBox.htmlText = myText;
}
}
}

```

Scrolling Text:

Can use the scroll-related properties of the `flash.text.TextField` class to manage lengthy content, either vertically or horizontally.

The scroll-related properties include **`TextField.scrollV`** , **`TextField.scrollH`** and **`maxScrollV`** and **`maxScrollH`** . Use these properties to respond to events, like a mouse click or a keypress.

```

package
{
    import flash.display.Sprite;
    import flash.text.*;
    import flash.events.MouseEvent;

    public class TextScrollExample extends Sprite
    {
        private var myTextBox:TextField = new TextField();
        private var myText:String = "Hello world and welcome to the show. It's
really nice to meet you. Take your coat off and stay a while. OK, show is
over. Hope you had fun. You can go home now. Don't forget to tip your waiter.
There are mints in the bowl by the door. Thank you. Please come again.";
    }
}

```

```
public function TextScrollExample()
{
    myTextBox.text = myText;
    myTextBox.width = 200;
    myTextBox.height = 50;
    myTextBox.multiline = true;
    myTextBox.wordWrap = true;
    myTextBox.background = true;
    myTextBox.border = true;

    var format:TextFormat = new TextFormat();
    format.font = "Verdana";
    format.color = 0xFF0000;
    format.size = 10;

    myTextBox.defaultTextFormat = format;
    addChild(myTextBox);
    myTextBox.addEventListener(MouseEvent.MOUSE_DOWN,
mouseDownScroll);
}

public function mouseDownScroll(event:MouseEvent):void
{
    myTextBox.scrollV++;
}
}
```

Cascading Style Sheets - In Code Example:

```

var style:StyleSheet = new StyleSheet();

var styleObj:Object = new Object();
styleObj.fontSize = "bold";
styleObj.color = "#FF0000";
style.setStyle(".darkRed", styleObj);

var tf:TextField = new TextField();
tf.styleSheet = style;
tf.htmlText = "<span class = 'darkRed'>Red</span> apple";

addChild(tf);

```

Cascading Style Sheets – Loading External CSS:

```

package
{
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.net.URLLoader;
    import flash.net.URLRequest;
    import flash.text.StyleSheet;
    import flash.text.TextField;
    import flash.text.TextFieldAutoSize;

    public class CSSFormattingExample extends Sprite
    {
        var loader:URLLoader;
        var field:TextField;
        var exampleText:String = "<h1>This is a headline</h1>" +
            "<p>This is a line of text. <span class='bluetext'>" +
            "This line of text is colored blue.</span></p>";

        public function CSSFormattingExample():void
        {
            field = new TextField();
            field.width = 300;
            field.autoSize = TextFieldAutoSize.LEFT;

```



```

        field.wordWrap = true;
        addChild(field);

        var req:URLRequest = new URLRequest("example.css");

        loader = new URLLoader();
        loader.addEventListener(Event.COMPLETE, onCSSFileLoaded);
        loader.load(req);
    }

    public function onCSSFileLoaded(event:Event):void
    {
        var sheet:StyleSheet = new StyleSheet();
        sheet.parseCSS(loader.data);
        field.styleSheet = sheet;
        field.htmlText = exampleText;
    }
}
}

```

Fonts and Font Mapping - Embedded and Device Fonts:

- Flash files embed fonts into the .swf by default.
- **Note:** Embedded fonts increase file size variable depending on the particular font, so be cautious.
- "Use device font" prevents embedding in static text.
- "Character" button does similar in input and dynamic text.
- If the end user does not have a particular font, you are taking a risk if you do not embed them.
- Flash will make a substitution font if necessary but this is not reliable.
- Certain fonts cannot be embedded - if Flash cannot antialias a font and it appears "jaggy", you should discard the font.

- When you do not have a particular font on your machine, you can temporarily reassign the font (Edit/ Font Mapping)

Creating a Font Library Item:

1. Open the library to add a font symbol to.
2. Select New Font from the Library Panel menu.
3. Enter a name for the font item in the Name text field.
4. Select a font from the Font menu or enter the name of a font in the Font text field.
5. (Optional) Select Bold or Italic.
6. (Optional) To embed the font information as bitmap data rather than vector outline data, select the Bitmap Text option, and enter a font size in the Size text field. (Bitmap fonts cannot use anti-aliasing. You must choose Bitmap as the anti-aliasing option in the Property inspector for text that uses this font.)