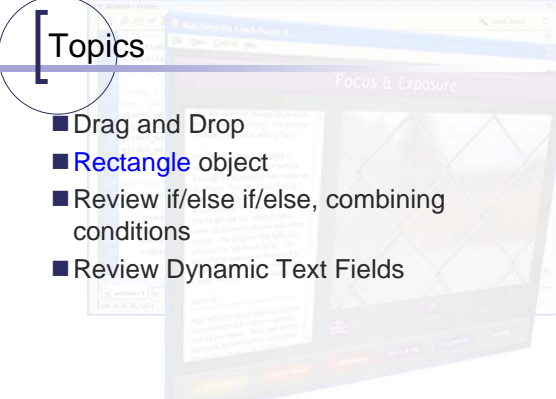# Drag and Drop, Custom Functions
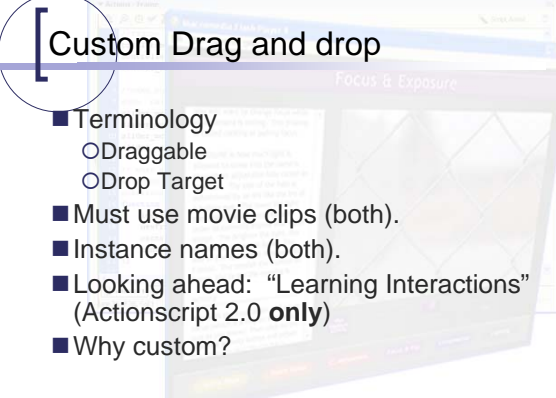
Utah State

Topics

- Drag and Drop
- Rectangle object
- Review if/else if/else, combining conditions
- Review Dynamic Text Fields

Utah State

Custom Drag and drop

- Terminology
  - Draggable
  - Drop Target
- Must use movie clips (both).
- Instance names (both).
- Looking ahead: "Learning Interactions" (Actionscript 2.0 **only**)
- Why custom?

Utah State

Nesting code blocks

```
if(car == "out of gas" || car == "broken" || pool = "closed") {
    // Can you spot the logic error?
    // If the car is out of gas OR the car is broken OR the
    // pool is closed, then no swimming lesson.
} else {
    if(rachel == "chicken") {
        // Went to swimming lessons, but Rachel
        // is chicken, so she is not swimming.
    } else {
        // Went to swimming lessons, Rachel is willing
        // to swim.
    }
}
```

Dynamic text fields

- Text can be altered using actionScript
- Syntax

```
objectPath.textFieldName.text = "newTextToShow";
```

- Example

```
(parent as MovieClip).feedbackText.text = "Wrong!
    Don't quit your day job.";
```

Getting Movie Clips to Move

- Use the startDrag() function

- Example:

```
function clickDrag0_2perc(evt:MouseEvent):void {
    this.drag0_2perc.startDrag();
}
```

Arguments for startDrag()

- Can lock center (boolean)
- Can set up boundaries (Rectangle)
- Syntax for rectangle:

var variableName:Rectangle = new Rectangle(*x*, *y*, *width*, *height*);

- Example:

var dragBounds:Rectangle = new Rectangle(40, 40, 704, 378);

- Revised Function Call:

this.drag0_2perc.startDrag(true, dragBounds);

Utah State

Detecting the drop zone

- dropTarget property
- References the displayObject your draggable is over
- Weird pathing issue:

this.drag0_2perc.dropTarget.parent.name

- Unless you created your object at run time, then:

this.drag0_2perc.dropTarget.name

Utah State

Other important properties

- Object coordinates (both read and write).
- .x; // the x coordinate of an object
- .y; // the y coordinate of an object

Utah State

3

Custom Functions

- We've done this before
- New concept: Passing arguments (data)
- Syntax (familiar)

(*variableName*:variableType, *nextName*:nextType)

- Example:

function calculateScore(nCorrect:int, nWrong:int):void

Once you have the arguments

- Call them up just like a variable.
- For example:

```
function showScore(nCorrect:int, nWrong:int) {
    var score:Number;
    score = nCorrect / (nCorrect + nWrong);
    trace(score);
}
```

- Function Call:

showScore(8, 2);