

# Drawing Shapes and Skinning Programmatically

---

## Import the project

1. In Flex Builder, select **File > Import > Flex Project**.
2. In the dialog window, select **Archive File** and browse to where **Ex18\_Starter.zip** is located in your local file system.
3. Click **Finish**.

Note: Alternatively you can start with the code you completed from exercise 17.

## Create a component

4. Right click on the **components** folder and select **New > MXML Component**.
5. Create the component file using the following properties:
  - Filename : GoldStar
  - Based on : Canvas
  - Width : remove default value
  - Height : remove default value
6. Click Finish.
7. To the beginning Canvas tag add a `creationComplete` event with a value of `drawStar()`.

## Draw a star

8. After the beginning Canvas tag, create a Script block.
9. Within the Script block, create a private function named `drawStar()` that takes no parameters and returns `void`.

```
private function drawStar():void {  
  
}
```

10. Within the function, refer to the `graphics` property of this HBox component and call the `clear()` method. This code removes any existing graphics from the stage.

```
this.graphics.clear();
```

11. Call the `beginFill()` method of the `graphics` property passing `0xFFD700` and `1` as parameters. This code sets the color of the graphic.

```
this.graphics.beginFill(0xFFD700, 1);
```

12. Call the `lineStyle()` method of the `graphics` property passing the following parameters: `2`, `0xFFD700`, `1`, `false`, `"normal"`, `"square"`, `"miter"`, `8`. This code sets the line and corner display.

```
this.graphics.lineStyle(2, 0xFFD700, 1, false, "normal", "square", "miter", 8);
```

Your code should appear as follows:

```
private function drawStar():void
{
    this.graphics.clear();
    this.graphics.beginFill(0xFFD700, 1);
    this.graphics.lineStyle(2, 0xFFD700, 1, false, "normal", "square", "miter", 8);
}
```

13. Still within the function, use the `moveTo()` method of the `graphics` property and pass `9` and `2` as the first and second parameters. This code sets the starting point for the line.

```
this.graphics.moveTo(9, 2);
```

14. Call the `lineTo()` method of the `graphics` property for each of the following parameters:

- `x: 5, y: 14`
- `x: 16, y: 7`
- `x: 3, y: 7`
- `x: 14, y: 14`
- `x: 9, y: 2`

This code sets all the points in the graphic.

Your code should appear as follows:

```

private function drawStar():void
{
    this.graphics.clear();
    this.graphics.beginFill(0xFFD700,1);
    this.graphics.lineStyle(2, 0xFFD700, 1, false,
"normal", "square", "miter", 8);
    this.graphics.moveTo(9, 2);
    this.graphics.lineTo(5, 14);
    this.graphics.lineTo(16, 7);
    this.graphics.lineTo(3, 7);
    this.graphics.lineTo(14, 14);
    this.graphics.lineTo(9, 2);
}

```

15. Save the file.
16. Open **RequestTicket.mxml** from the **components** folder.
17. Around line 82, locate the following code:

```

<mx:Text x="0" y="351"
text="Max Rectangle Tables:
{selectedRoom.maxsquaretables}" fontSize="10"/>

```

18. After the Text component, add an instance of the GoldStar component from the comp namespace.
19. Add the following properties and values:

- id: star1
- width: 15
- height: 15
- x: 0
- y: 371

```

<comp:GoldStar id="star1"
width="15" height="15"
x="0" y="371"/>

```

20. Create four additional instances of the GoldStar component using the following values:

```

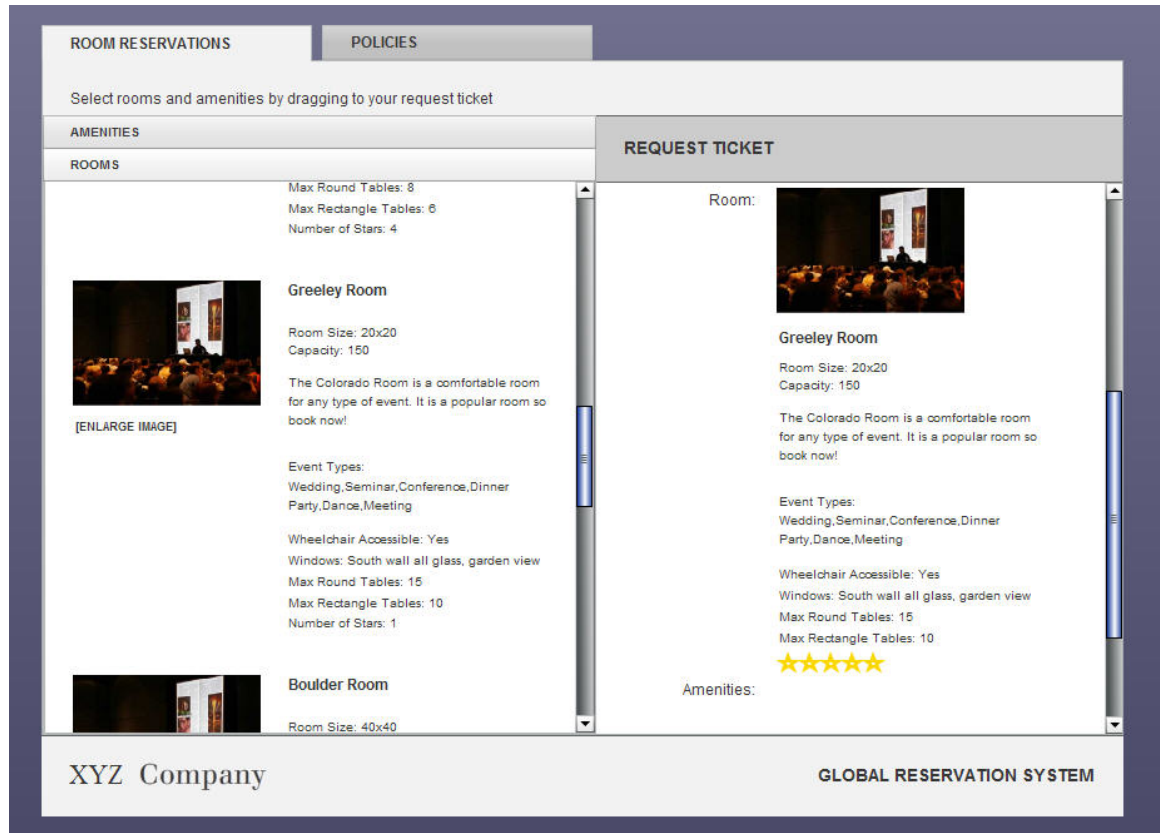
id: star2, width: 15, height: 15, x: 17, y: 371
id: star3, width: 15, height: 15, x: 34, y: 371
id: star4, width: 15, height: 15, x: 51, y: 371
id: star5, width: 15, height: 15, x: 68, y: 371

```

Your code should appear as follows:

```
<comp:GoldStar id="star1"
    width="15" height="15"
    x="0" y="371"/>
<comp:GoldStar id="star2"
    width="15" height="15"
    x="17" y="371"/>
<comp:GoldStar id="star3"
    width="15" height="15"
    x="34" y="371" />
<comp:GoldStar id="star4"
    width="15" height="15"
    x="51" y="371" />
<comp:GoldStar id="star5"
    width="15" height="15"
    x="68" y="371"/>
```

21. Save the file and run the application.
22. In the **Rooms** panel locate the **Greeley Room**.
23. Click and drag it over to the **Request Ticket** and release your mouse.  
You should see the room information appear in the form. There are 5 stars.



## Using a conditional statement in a binding

24. Return to Flex Builder and locate the **star1** GoldStar component instance.
25. Add a `visible` property with a value bound to a conditional statement of `selectedRoom.stars > 0 ? true : false`.

```
<comp:GoldStar id="star1"
  width="15" height="15"
  x="0" y="371"
  visible="{selectedRoom.stars > 0? true: false}"/>
```

26. Add the `visible` property to each instance of the GoldStar components incrementing the value in the conditional statement.

Your code should appear as follows:

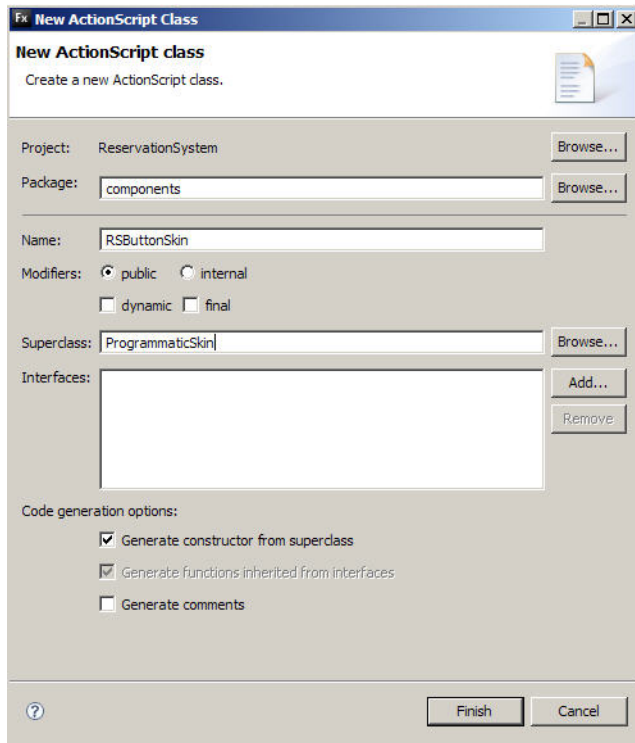
```
<comp:GoldStar id="star1"
  width="15" height="15"
  x="0" y="371"
  visible="{selectedRoom.stars > 0? true: false}"/>
```

```
<comp:GoldStar id="star2"
    width="15" height="15"
    x="17" y="371"
    visible="{selectedRoom.stars > 1? true: false}"/>
<comp:GoldStar id="star3"
    width="15" height="15"
    x="34" y="371"
    visible="{selectedRoom.stars > 2? true: false}"/>
<comp:GoldStar id="star4"
    width="15" height="15"
    x="51" y="371"
    visible="{selectedRoom.stars > 3? true: false}"/>
<comp:GoldStar id="star5"
    width="15" height="15"
    x="68" y="371"
    visible="{selectedRoom.stars > 4? true: false}"/>
```

27. Save the file and run the application.
28. Drag the **Greeley Room** from the **Rooms** panel to the **Request Ticket** form.  
There is one star.
29. Click and drag any other room.  
Notice the number of stars.

## Create a button skin programmatically

30. Return to Flex Builder and right-click on the components folder and select **New > ActionScript Class**.
31. Use the following values in the dialog window:
  - Package: components
  - Name: RSButtonSkin
  - Modifiers: public
  - Superclass: ProgrammaticSkin
  - Generate constructor from superclass: checked



32. Click **Finish**.
33. After the class definition and before the constructor, create a public variable named `backgroundColor` datatyped as `Number`.
34. Create another public variable named `lineThickness` datatyped as `Number`.

```
public var backgroundColor:Number;  
public var lineThickness:Number;
```

35. After the constructor override the protected function `updateDisplayList`.
36. Pass the two parameters `unscaledWidth` variable and the `unscaledHeight` variable datatyped as `Number` types. The function returns `void`.

```
override protected function updateDisplayList  
(unscaledWidth:Number, unscaledHeight:Number):void  
{  
}
```

37. Create a switch conditional statement testing for the value of the `name` variable.

```
switch (name) {  
}
```

38. Within the switch curly braces, add a case statement for the upSkin state.  
39. Assign 1 to lineThickness and 0xFFFFFFFF to backgroundColor then break.

```
switch (name) {  
    case "upSkin":  
        lineThickness = 1;  
        backgroundColor = 0xFFFFFFFF;  
        break;  
}
```

40. Add a case statement for the overSkin state.  
41. Assign 2 to lineThickness and 0xF2F2F2 to backgroundColor then break.

```
case "overSkin":  
    lineThickness = 2;  
    backgroundColor = 0xF2F2F2;  
    break;
```

42. Add a case statement for the downSkin state.  
43. Assign 1 to lineThickness and 0xCCCCCC to backgroundColor then break.

```
case "downSkin":  
    lineThickness = 1;  
    backgroundColor = 0xCCCCCC;  
    break;
```

44. Add a case statement for the disabledSkin state.  
45. Assign 1 to lineThickness and 0xCCCCCC to backgroundColor then break.

Your code should appear as follows:

```
switch (name) {  
    case "upSkin":  
        lineThickness = 1;  
        backgroundColor = 0xFFFFFFFF;  
        break;
```

```

        case "overSkin":
            lineThickness = 2;
            backgroundColor = 0xF2F2F2;
            break;
        case "downSkin":
            lineThickness = 1;
            backgroundColor = 0xCCCCCC;
            break;
        case "disabledSkin":
            lineThickness = 1;
            backgroundColor = 0xCCCCCC;
            break;
    }

```

46. After the conditional statements but still within the function, create a local variable named `g` datatyped as `Graphics`.
47. Assign the `graphics` object to `g`. The `graphics` property of the `ProgrammaticSkin` class allows you to create drawings.

```
var g:Graphics = graphics;
```

48. Next call the `clear()` method of `g`.
49. Call the `beginFill()` method and pass `backgroundColor` and `1.0` as the first and second parameters.
50. Call the `lineStyle()` method and pass `lineThickness` and `0x363636` as the first and second parameters.
51. Call the `drawEllipse()` method and pass `0`, `-3`, `70` and `30` as the parameters.

Your code should appear as follows:

```

var g:Graphics = graphics;
g.clear();
g.beginFill(backgroundColor, 1.0);
g.lineStyle(lineThickness, 0x363636);
g.drawEllipse(0, -3, 70, 30);

```

52. Locate the `import` statement at the top of the file.
53. After the `import` statement, import the `Graphics` class from the `flash.display` package.

```
import flash.display.Graphics;
```

54. Save the file.

## Apply a programmatic skin

55. Open **styles.css** from the **assets** folder.

56. At the top of the file, create a `Button` selector.

57. Within the selector, create an property named `overSkin` with a value of `ClassReference("components.RSButtonSkin")`.

```
Button {
    overSkin:
    ClassReference("components.RSButtonSkin");
}
```

58. Create an `upSkin` property with the value

`ClassReference("components.RSButtonSkin")`.

59. Create a `downSkin` property with the value

`ClassReference("components.RSButtonSkin")`.

Your code should appear as follows:

```
Button {
    overSkin: ClassReference("components.RSButtonSkin");
    upSkin: ClassReference("components.RSButtonSkin");
    downSkin: ClassReference("components.RSButtonSkin");
}
```

60. Save the file and run the application.

You should see the login button has changed to an oval but functions as a normal button.

**USER LOGIN**

UserName:

Password:

**SUBMIT** 

XYZ Company

GLOBAL RESERVATION SYSTEM