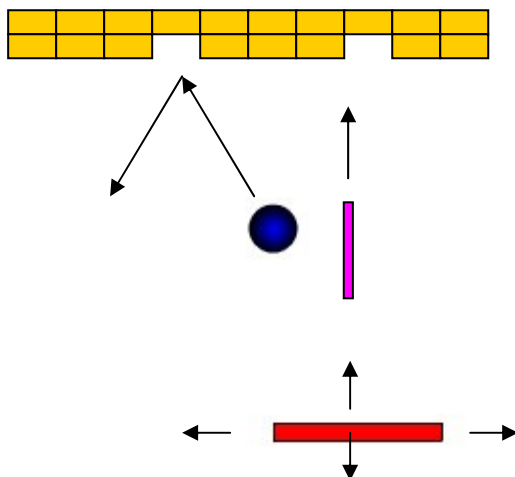


Games Scripting and Programming Workshop

Using action scripting as a basis for Object Oriented Programming 2

Flash Actionscript 2.0



Allan Morrison
Senior Project Officer
Museum Magnet Schools
Queensland Museum
P.O. Box 3300, South Bank, Qld. 4101,
Australia.
Web: <http://www.mms.qld.edu.au>
Email: allan.morrison@qm.qld.gov.au or
mms@qm.qld.gov.au
Ph. 07 3840 7611
Fax 07 3840 7607



Introduction to Object Oriented Programming 2 (OOP)

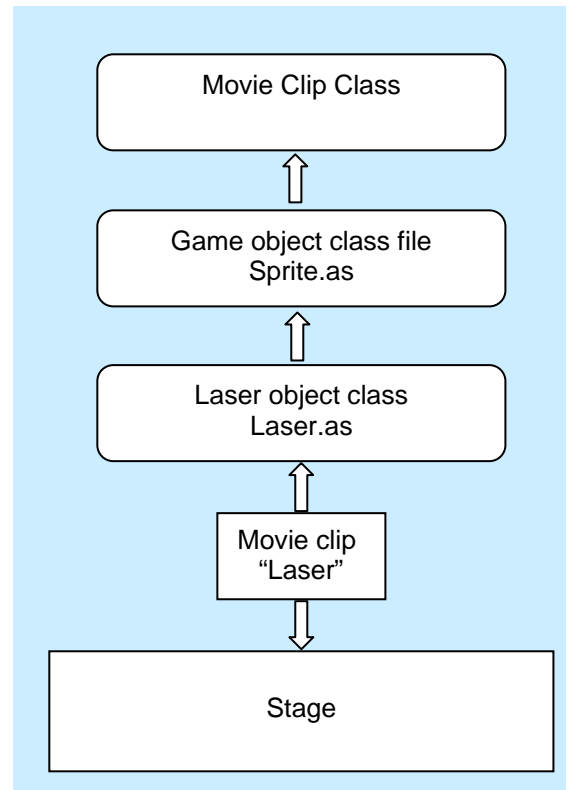
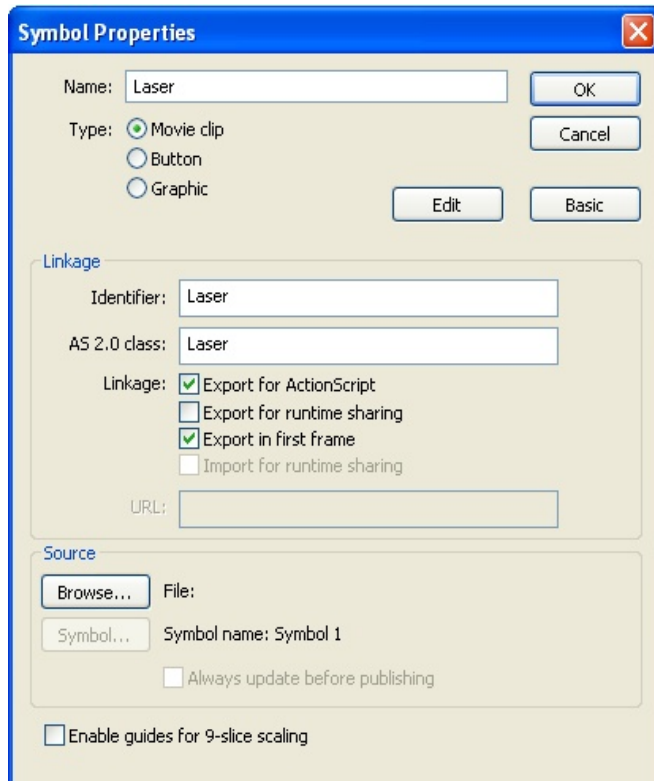
Flash Actionscript 2.0

Overview

This tutorial adds the functionality of firing lasers from the bat to destroy the brick wall. To achieve this, a laser class is added that inherits all the sprite class methods but not all of its properties. Some additional methods and properties are added to the Sprite class so that all “Sprites” can access these methods. The idea is to keep all general or shared properties in the sprite class while all specifically laser properties are contained in the laser class. This is an important concept of OOP known as **inheritance**.

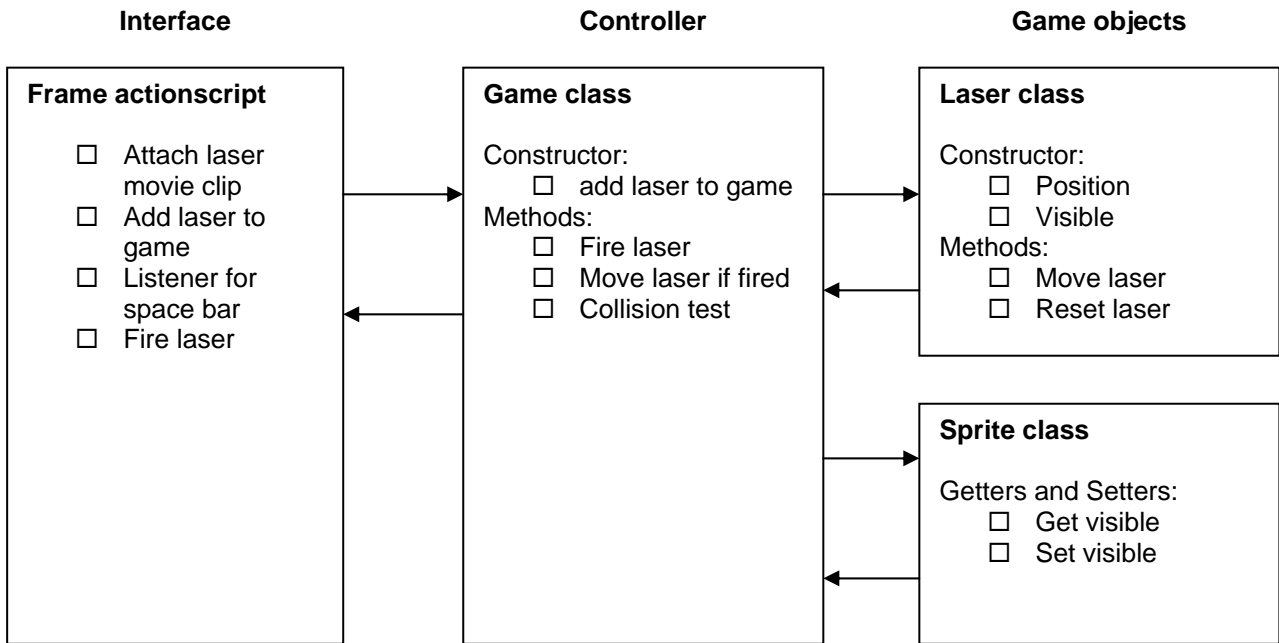
Getting started

1. A laser movie clip is created and given the following properties:



The process from here involves:

2. Frame actionscript code is added to:
 - Attach movie to stage
 - Add laser to game
 - Listen for the space bar to be pressed to fire the laser
 - Move the laser if fired
3. Game code is modified to:
 - Add laser to game
 - Fire laser
 - Move laser
 - Check for laser collisions with bricks
4. Laser class file is created to:
 - Set initial properties of the laser
 - Define method of movement
5. Sprite class modified to:
 - Add visible getter and setter



There is quite a lot to do. The key to success is to **build a solution** with small steps that are tested often.

To break the task down into manageable steps, an approach might be:

1. Add the laser to the stage
2. Add the laser to the game
3. Listen for the space bar
4. Fire the laser
5. Move the laser if fired
6. Check for laser collisions, remove bricks and reset status



Phase 1

Add a laser to the stage

1. Create a class file for a laser that defines its properties and methods.
2. Create a laser movie clip.
3. Set linkage between the laser object and the class file
4. Add actionscript to the flash game file to initiate the action

1. Class file for a laser

```
class Laser extends Sprite {
//
//===== Declare variables=====
//
var spY:Number;
var isActive:Boolean = false;
//
//===== Constructor =====
//
public function Laser() {
    // set starting speed
    spY = -10;
    // set starting visibility
    this._visible = isActive;
}
//
//===== Methods =====
//
// move Laser method

//
// ===== Getters =====
//
public function getActive():Boolean {
    return isActive;
}
//
// ===== Setters =====
//
public function setActive(_isActive):Void {
    isActive = _isActive;
    //trace(isActive);
    this._visible = isActive;
}
}
```

2. Example Flash file

- Open the game flash file Game fla.
- Create a laser object and convert to the symbol Laser saved in the library.
- Remove the laser from the stage.

3. Set the properties of the Laser movie clip as above in the getting started section.

4. In frame 1 of layer 1 of the flash file, add the following actionscript:

- Add laser to stage

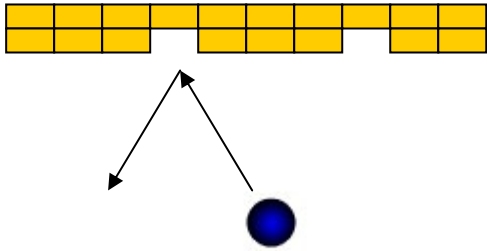
```
// attach movie to stage  
attachMovie("Laser", "laser", 400);
```

This script creates a new instance of the Laser movieclip and places it on the stage in layer 400 according to the initial values in the constructor.



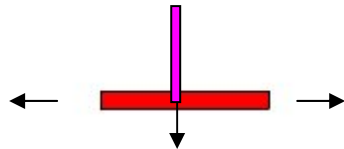
Checkpoint 1

Test your project now



Functionality achieved:

- Laser appears on stage in a given starting position



Next phase of development:

- add the laser to the game

Notes:



Phase 2

Add the laser to the game

This task is straight forward. Code only needs to be added in two places:

Frame actionscript:

```
// instantiate new game
//setup
var myGame:Game = new Game(ball, bat, brickArray, laser);
```

Game class:

```
class Game {
    //===== declare variables =====
    private var ball:Sprite;
    private var bat:Sprite;
    //private var brick:Sprite;
    private var brickArray:Array;
    private var laser:Laser;
    private var dir:String;

    //===== constructor =====
    public function Game (_ball:Sprite, _bat:Sprite, _brickArray:Array, _laser:Laser) {
        ball=_ball;
        bat=_bat;
        laser=_laser;
        brickArray=_brickArray;
    }
}
```



Test your project now

Notes:



Phase 3

Listen for the space bar

Control of on screen objects is achieved through the use of listeners. Listeners respond to keystrokes or mouse action events. The listener is placed in the actionscript of the first frame of the flash file as this is an interface layer event.

1. Keyboard listener

A keyboard listener can used to respond to the space bar for firing the laser.

Frame actionscript:

```
// add keyboard listener
// instantiate new keyboard listener object
var keyListener:Object = new Object();
var dir:String;
var isFired:Boolean = false;
keyListener.onKeyDown = function() {
    //direction keys
    if (Key.isDown(Key.RIGHT)) {
        dir = "right";
    } else if (Key.isDown(Key.LEFT)) {
        dir = "left";
    } else if (Key.isDown(Key.UP)) {
        dir = "up";
    } else if (Key.isDown(Key.DOWN)) {
        dir = "down";
    }
    //space bar
    if (Key.isDown(Key.SPACE)) {
        isFired = true;
    }
};
keyListener.onKeyUp = function() {
    dir = "stop";
    isFired = false;
};
// add key listener method to key properties
Key.addListener(keyListener);
```

The *isFired* variable stores the firing status if the space bar is pushed. Once the space bar is released, the *isFired* value is reset. This ensures the laser doesn't continually fire once started.



Test your project now

To test your project, add a trace into the game loop to check that the space bar is being detected. Values should alternate between true and false as the space bar is pushed.

```
// loop
this.onEnterFrame = function() {
    trace(isFired);
    ball.moveSprite();
    bat.moveDir(dir);
}
// loop
this.onEnterFrame = function() {
    ball.moveSprite();
    bat.moveDir(dir);
```



Summary of actionscript code on timeline:

```
//declare variables
var brickArray:Array = new Array(20);
// attach movie to stage
attachMovie("Ball", "ball", 100);
attachMovie("Bat", "bat", 200);
//attach movie and place bricks on stage
for (var i = 0; i<20; i++) {
    attachMovie("Brick", "brick"+i, 300+i);
    //load bricks into array
    brickArray[i] = _root["brick"+i];
    //place bricks on stage screen
    //change width
    brickArray[i].setWidth(Stage.width/10);
    //change height
    brickArray[i].setHeight(15);
    //adjust values dependent on size of brick
    if (i<10) {
        brickArray[i].setX(5+brick0.getWidth()*i);
        brickArray[i].setY(10);
    } else {
        brickArray[i].setX(5+brick0.getWidth()*(19-i));
        brickArray[i].setY(10+brick0.getHeight());
    }
}
// add laser movie
attachMovie("Laser", "laser", 400);

// instantiate new game
//setup
var myGame:Game = new Game(ball, bat, brickArray, laser);
// add mouse listener
// instantiate new mouse listener object
var mouseListener:Object = new Object();
var isDragging:Boolean = false;
// define mouse drag event
mouseListener.onMouseDown = function() {
    this.isDragging = true;
};
mouseListener.onMouseMove = function() {
    if (this.isDragging) {
        bat.setX(_xmouse-bat.getWidth()/2);
        bat.setY(_ymouse-bat.getHeight()/2);
    }
};
mouseListener.onMouseUp = function() {
    this.isDragging = false;
};
// add mouse listener method to mouse properties
Mouse.addListener(mouseListener);
// add keyboard listener
// instantiate new keyboard listener object
var keyListener:Object = new Object();
var dir:String;
var isFired:Boolean = false;
keyListener.onKeyDown = function() {
    //direction keys
    if (Key.isDown(Key.RIGHT)) {
        dir = "right";
    } else if (Key.isDown(Key.LEFT)) {
        dir = "left";
    }
};
```



```
} else if (Key.isDown(Key.UP)) {
    dir = "up";
} else if (Key.isDown(Key.DOWN)) {
    dir = "down";
}
//space bar
if (Key.isDown(Key.SPACE)) {
    isFired = true;
}
};
keyListener.onKeyUp = function() {
    dir = "stop";
    isFired = false;
};
// add key listener method to key properties
Key.addListener(keyListener);
// loop
this.onEnterFrame = function() {
    //trace(isFired);
    //myGame.fireLaser(isFired);
    myGame.moveSprites(dir);
    myGame.checkCollisions();
};
```

Notes:



Phase 4

Fire the laser.

To fire the laser, a fire laser method must be added to the game class that receives the *isFired* status from the game loop. Essentially, each game loop the status is passed to the Game controller that determines if the laser has been fired, then acts accordingly.

1. Class file: Game.as

- Add fire laser method

```
class Game {
    //===== declare variables =====
    private var ball:Sprite;
    private var bat:Sprite;
    //private var brick:Sprite;
    private var brickArray:Array;
    private var laser:Laser;
    private var dir:String;
    private var isFired:Boolean;
    private var laserActive:Boolean = false;

    //===== constructor =====
    public function Game (_ball:Sprite, _bat:Sprite, _brickArray:Array, _laser:Laser) {
        ball=_ball;
        bat=_bat;
        //brick=_brick;
        laser=_laser;
        brickArray=_brickArray;
    }
    //===== public methods =====
    // fire laser
    public function fireLaser(_isFired:Boolean) {
        isFired = _isFired;
        if(isFired) {
            //make active
            laserActive = true;
            laser.setActive(laserActive);
            // set starting point
            laser.setX(bat.getX()+bat.getWidth()/2);
            laser.setY(bat.getY()-laser.getHeight()/2);
        }
    }
}
```

The fire laser method makes the laser active if it has been fired. The laser is then set to being visible with starting position at the centre of the bat.

2. Frame action script

- Call game check fire laser function

```
// loop
this.onEnterFrame = function() {
    //trace(isFired);
    myGame.fireLaser(isFired);
    myGame.moveSprites(dir);
    myGame.checkCollisions();
};
```





Test your project now

When the space bar is pressed, the laser should appear in the position of the bat and remain stationary. We haven't told it to move yet!

Phase 5

Move the laser if fired

The laser should only move forward after it has been fired. Firing the laser sets the values of *isActive* and *visible* to true. These states can be used as flags for conditionally moving the laser.

1. Laser class methods

- Add the move laser function

```
//  
//===== Methods =====  
//  
// move Laser method  
public function moveLaser():Void {  
    //move forward at set speed  
    this._y += spY;  
    //check if top boundary reached  
    if (this._y<=5) {  
        //reset y  
        reSet();  
    }  
}  
public function reSet():Void {  
    //  
    this._visible = false;  
}
```

This method moves the laser without condition. When the upper boundary is reached the laser is not visible.

2. Game class methods

- Add move laser method

```
// move all sprites  
public function moveSprites(_dir:String):Void{  
    dir = _dir;  
    moveLasers();  
    moveBall();  
    moveBat(dir);  
}  
  
// ===== private methods =====  
//move lasers  
private function moveLasers() :Void {  
    // move only the active lasers  
    if(laser.getActive()) {  
        laser.moveLaser();  
    }  
}
```

If the laser is active, then move it forward.





Checkpoint 5

Lasers should now fire from the bat and move up until they disappear off the top of the screen. Bricks will not interact until a collision test has been added.

Phase 6

Check for laser collisions with bricks

```

//check collisions
public function checkCollisions():Void {
    if(isCollision(bat,ball)) {
        // do something
        ball.reBoundY();
    }
    // loop to test all bricks
    for (var i=0; i<20; i++) {
        // brick and ball
        if(isCollision(brickArray[i],ball)) {
            // do something
            ball.reBoundY();
            brickArray[i].reMove();
        }
        //brick and laser
        if((isCollision(brickArray[i], laser))&&(laser.getActive())) {
            laserActive = false;
            laser.setActive(laserActive);
            brickArray[i].reMove();
        }
    }
}

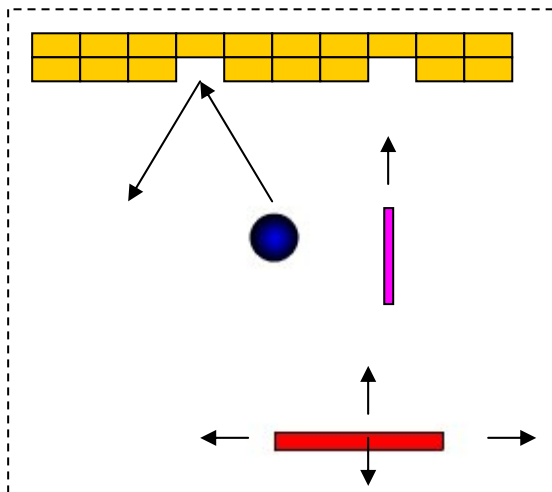
```

Collisions are only detected if the laser is active. Once the collision is detected, the laser is set to inactive and invisible. The brick is removed from the stage. Note that all the laser code added is easily extended to multiple lasers.



Checkpoint 6

Test your project now



Functionality achieved:

- Lasers fire
- Move towards bricks
- Collision with bricks
- Laser and brick are removed
- Laser resets after collision or passing boundary.

Next phase of development:

- Multiple lasers
- Smart lasers?
- ???

