

Day 1 - Flash PacMan – Create a dot Movie Clip

Question of the Day:

- Open **pacMan-yourName fla**
- Why would you want to **embed code in a movie clip** in a Flash movie?
- List the steps to embed code in a movie clip.
- Summarize what the following code does: `_parent.removeMovieClip()`

Objectives:

- Answer Question of the Day.
- Understand: 1) the reason to embed code in symbols in Flash 2) steps to embed code in a movie clip.
- **Demo:** use ActionScript to embed code in Symbols.
Ex. = Pac-man-Final fla

Assignment:

- Open **pac-man-yourName fla**
- Create a **dot movie clip** in the library. Embed ActionScript in the dot that will remove the dot from the screen when Pac-Man hits the dot.
- **Hint:** follow the steps below to complete this assignment.

Answer to Question of the Day:

- Why would you want to **embed code in a movie clip** in Flash?

The primary reason to embed code in Flash Symbols (like a movie clip) is so **code can be reused**. For example, create one **dot symbol** in a Pac-Man program; embed the code in it and then use the same dot throughout the program. This makes the program more efficient because you do not need to add code to every dot that appears on the screen.

Steps to embed code in a movie clip:

1. Draw a dot on the stage.
2. Use **Modify > Convert to Symbol** to turn the dot shape into a movie clip, provide a descriptive name (Ex: **mc_dot**) and click “**Export for ActionScript**”.
3. Double-click on the movie clip so that you are **editing the timeline of the movie clip**. Turn the graphic in the movie clip’s timeline into a movie clip, provide a descriptive name that indicates the movie clip isn’t used at the root level (Ex: **mc_dot_layer2**) and click “**Export for ActionScript**”.
4. Select **mc_dot_layer2** and add code in the Actions window. The following code makes a dot disappear when Pac-Man collides with it.

```
onClipEvent(enterFrame) {  
    if (this.hitTest(_root.pac)) {  
        trace("pac hit a dot")  
        _parent.removeMovieClip()  
    } // end hitTest(_root.pac)  
} // end onClipEvent(enterFrame)
```

`_parent.removeMovieClip()` removes the current instance of a movie clip. In order for `removeMovieClip()` to function, the movie clip *must be added to the stage* with `attachMovie()`.

Day 2 & 3 - Flash PacMan – Create walls and build a maze

Question of the Day:

- What is `_root` in ActionScript?
- How can we create a “**right wall**” with code embedded inside that will keep PacMan from exiting the stage as he moves from left to right?

Objectives:

- Answer Question of the Day.
- **Understand:** how to create walls with code embedded inside that will keep PacMan from passing through the wall.
- **Assignment:** create 4 walls with code embedded inside. Use the walls to create a maze for the PacMan game. Ex. = [Pac-man-Final.fla](#)

Answer to Question of the Day:

- **_root:** The **_root property** specifies or returns a reference to the root movie Timeline. If the currently executing movie has multiple levels, *the current level is the _root reference*. In the example code below, `_root.Pac` is telling the wall to go back to the **main stage** and look for an instance named **Pac**.

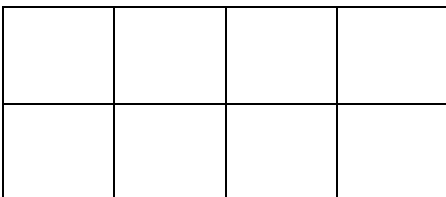
Demo / Answer to Question of the Day:

- Use the rectangle drawing tool to create a vertical wall on the right side of the stage.
- Use **Modify > Convert to Symbol** to turn the wall shape into a movie clip, provide a descriptive name (Ex: `mc_rightWall`) and click “**Export for ActionScript**”.
- Double-click on the movie clip so that you are **editing the timeline of the movie clip**. Turn the graphic in the movie clip’s timeline into a movie clip, provide a descriptive name that indicates the movie clip isn’t used at the root level (Ex: `mc_rightWall_layer2`) and click “**Export for ActionScript**”.
- Select `mc_rightWall_layer2` and add code in the Actions window. The following code makes PacMan bounce off the right wall when he collides with it.
- **Example code:** to make Pac-Man bounce away from a right, vertical wall. The code is placed on a movie clip inside a movie clip.

```
// bounce Pac left away from the right wall
onClipEvent(enterFrame) {
    if(this.hitTest(_root.Pac)){
        // 'bounce' Pac away from the wall
        _root.Pac._x -= 10
        // Stop moving Pac
        _root.xmove = 0
    } // hitTest(_root.Pac)
} //enterFrame
```

Assignment: → **DUE at the end of class on day 3.**

- Open `pac-man-yourName.fla`
- Create walls and embed code in the wall so that Pac-Man cannot go through the wall. **Hint:** it will probably work best to create 4 wall types: **RightWall, LeftWall, TopWall, BottomWall**.
- Consider your stage to be a grid. To see it as a grid and change grid size, use the following features:
View > Grid > Show Grid
View > Grid > Edit Grid
- Use the walls to **create a maze** by dragging wall movie clips from the library and placing them on the stage.



Day 4&5 - Flash PacMan – Place dots in the Maze

Question of the Day:

- Why do we need to use ActionScript code to place the dots on the stage?
- What is **attachMovie()** used for in ActionScript?
- What is **_parent** in ActionScript?
- How can we use loops to place our dots on the stage?

Objectives:

- Answer Question of the Day.
- Understand:
 - The reason we need to use Actionscript code to place dots on the stage.
 - What **attachMovie()** is for and how to use it.
 - What **_root** and **_parent** are use for in ActionScript.
- **Demo:** use **attachMovie()** and nested loops to place multiple instances of a symbol on the Flash stage.
[Ex. = Pac-man-Final fla](#)
- **Assignment:** write a Flash function that will place dots on the stage between the walls.

Assignment:

- Open **pac-man-yourName fla**
- Write a Flash function that will place dots on the stage between the walls using the **attachMovie()** method.
- **Hint:** use the **Example Code** below to help complete this assignment.

Answer to Question of the Day:

- Why do we need to use Actionscript code to place the dots on the stage? → In ActionScript, the **removeMovieClip()** method requires that symbols *must be added to the stage* with **attachMovie()**. In other words, if we drag symbols from the library and place them on the stage, the **removeMovieClip()** method will not remove them from the stage.
- **attachMovie()** is used to attach movie clip instances at run-time. The **attachMovie()** method attaches an instance of a movie clip symbol in the SWF file's library to the stage. Before **removeMovieClip()** can be used, the instances must be added to the stage with **attachMovie()**.
- **_parent:** The **_parent property** specifies or returns a reference to the movie clip or object that **contains** the current movie clip or object. The current object is the object containing the ActionScript code that references **_parent**. Use **_parent** to specify a relative path to movie clips or objects that **are above** the current movie clip or object.

Example Code: (placed in frame 1 of Actions layer)

```
_root.onLoad = function() {  
    LoadDots() // calls the function that we write to place dots on the stage  
} // end _root.onLoad
```

// the following user defined function is used to

// fill the game board with dots for Pac to eat

```
function LoadDots() {  
    dotNum = 100 // counter to see how many dots are in  
                // the level and control the depth  
  
    // row 1  
    for(a=0; a<11; a++) {  
        name = "dot" + dotNum  
        _root.attachMovie("mc_dot", name, dotNum)  
        _root[name]._x = a*50 + 25  
        _root[name]._y = 25  
        dotNum +=1  
    } // end row 1  
}
```

```

// column 1
for(a=1; a<7; a++) {
    name = "dot" + dotNum
    _root.attachMovie("mc_dot", name, dotNum)
    _root[name]._x = 25
    _root[name]._y = a*50 + 25
    dotNum +=1
} // end column 1
// column 11
for(a=1; a<7; a++) {
    name = "dot" + dotNum
    _root.attachMovie("mc_dot", name, dotNum)
    _root[name]._x = 525
    _root[name]._y = a*50 + 25
    dotNum +=1
} // end column 11
// row 2 (place 1 dot)
name = "dot" + dotNum
_root.attachMovie("mc_dot", name, dotNum)
_root[name]._x = 275
_root[name]._y = 75
dotNum +=1
for(a=7; a<9; a++) {
    name = "dot" + dotNum
    _root.attachMovie("mc_dot", name, dotNum)
    _root[name]._x = a*50 + 25
    _root[name]._y = 75
    dotNum +=1
} // end row 2
} // function LoadDots()

```

Notes to Code:

- For Loops start with the column or row value of the first cell to fill and end with the last cell to fill. The expression **for(a=7; a<9; a++)** would fill cells 7, 8, and 9 of the row or column.
- The code above assumes a grid size of 50 by 50, and places a dot in the middle of each cell.

Day 6 - Flash PacMan – Add Scoring

Question of the Day:

- How can we add scoring capability to our Pac-Man game?

Objectives:

- Answer Question of the Day and prior day questions.
- **Understand:** How to use variables to create a scoreboard.
- **Assignment:** add a scoreboard to the PacMan game, add to the score every time PacMan eats a dot.
[Ex. = Pac-man-Final fla](#)

Assignment:

- Open **pac-man-yourName fla**
- Add a scoreboard to the bottom of the screen.
- Add code to **mc_dot** so that each time Pac-Man eats a dot, 10 points are added to a variable named **score** and the scoreboard is updated.
- **Extra:** Add a timer in addition to the scoreboard.

Answer to Question of the Day:

- How can we add **scoring** capability to our Pac-Man game?
 1. Create a variable named **score** and set its initial value to **0**.
In the **Actions** layer, add the following to **_root.onLoad** function:
score = 0
 2. Create a *static text box* that says “**Score**”.
 3. Create a *dynamic text box*, in the **Var:** property, enter **score**.
 4. Test your project, the display should show:
Score: 0
 5. Now, add the following code to the **hitTest** on the movie clip inside **mc_dot**. Access this by editing **mc_dot**, and highlight the movie clip in the frame:

```
if (this.hitTest(_root.pac)) {  
    _root.dotNum -= 1  
    _root.score += 10  
} // end hitTest
```
 6. Follow the procedure from steps 2 & 3 to display how many dots remain on the screen.

Day 7 - Flash PacMan – Adding Sound

Question of the Day:

- Review prior questions.
- How can we add sound to our Pac-Man game? We want a sound to play each time a dot is eaten.
- Copy “[eatpill1.wav](#)” to your H:\ drive from
<\\tjhsnt01\Shared\MrFornstrom\ClassResources\PacManSounds>

Objectives:

- Answer Question of the Day.
- **Understand:** adding and playing sound files in a Flash game.
- **Assignment:** add sound to the PacMan game.
Ex. = Pac-man-Final fla

Assignment:

- Open **pac-man-yourName fla**
- Follow the instructions below so that each time Pac-Man eats a dot, the eatPill.wav sound is played.

Answer to Question of the Day:

How can we add sound to our Pac-Man game? We want a sound to play each time a dot is eaten.

1. Import sound [eatpill1.wav](#) into the library. After importing the sound file, right-click the file in the Library, choose **linkage** and check “**Export for ActionScript**”
2. Add the following to the **Actions frame** of the main timeline.

```
// create a new movie clip and name it mcSound_pacEat
_root.createEmptyMovieClip("mcSound_pacEat", 109)
// create a Sound object with the name mySoundEat
// associate mySoundEat object with " mcSound_pacEat " movie clip
mySoundEat = new Sound(mcSound_pacEat)
// Load "eatpill1.wav" from the library into the mySoundEat object
mySoundEat.attachSound("eatpill1.wav")
```

3. Add the following code to the **hitTest** on the movie clip inside **mc_dot**. Access this by editing **mc_dot**, and highlight the movie clip in the frame:

```
if (this.hitTest(_root.pac)) { → this line is not needed, just the line below:
    _root.mySoundEat.start()
}
```

Day 8 – Flash PacMan – Adding Open Screen

Question of the Day:

- **How can we add an opening screen to the Pac-Man game?** The screen should include game instructions, creator information, and information about when and where the project was created.

Objectives:

- Answer Question of the Day and review prior questions.
- Work on Assignment. [pac-man-yourName fla](#)

Assignment:

- Open [pac-man-yourName fla](#)
- Add an opening screen to the game.

Answer to Question of the Day:

How can we add an opening screen to the Pac-Man game?

1. Open the Scene window – Window > Other Panels > Scene
2. Use the “+” button to create a new scene, double-click on the name, and type “Open”. You now have a new scene named “Open”. Double-click on the original scene name, and name it “Game”.
3. Use text boxes and graphics to create an attractive Open screen. Include instructions, creator information, and information about when and where the project was created.
4. In the **Scene** window, drag **Open** to above the game scene. This will make the Open scene play when the program begins.
5. Copy the **LoadDots()** function from the main game window to the Open screen window.
6. Add a button for the Open screen. Place ActionScript in the frame of the Actions layer. Code for the button (instance name = **playGame**).

```
stop()
```

```
playGame.onRelease = function() {  
    score = 0  
    lives = 0  
    LoadDots()  
    _root.createEmptyMovieClip("mcSound_gameStart", 1111)  
    mySoundGameStart = new Sound(mcSound_gameStart)  
    mySoundGameStart.attachSound("newgame.wav")  
    mySoundGameStart.start()  
    gotoAndPlay("Game", 1)  
}
```

Day 9 – Flash PacMan – Making Ghosts Move & Bounce off Walls

Question of the Day:

- What code can be placed on the ghosts to make them move?
- What code can be placed in the walls to make a ghost bounce off the wall in a random direction?

Objectives:

- Answer Question of the Day.
- **Understand:** adding movement code to ghosts and making ghosts bounce off walls in a random direction.
- **Assignment:** adding ghosts to the PacMan game.
Ex. = [Pac-man-Final.fla](#)

Assignment:

- Open **pac-man-yourName.fla**
- Add an animated ghost to the stage.
- Add code to the ghost so it moves.
- Add code to the walls so that when the ghost contacts a wall it bounces in a random direction.
- **Extra:** add code to the ghosts so that when they touch Pac-Man, something happens (Ex: lose a life).

Answer to Question of the Day:

1. What code can be placed on the ghosts to make them move?

// Set up variables to track ghost movement

```
onClipEvent(load) {  
    this.xmove = 3  
    this.ymove = 0  
}
```

// Make the ghost move on each enterFrame

```
onClipEvent(enterFrame) {  
    this._x += this.xmove  
    this._y += this.ymove  
}
```

2. What code can be placed in the walls to make a ghost bounce off the wall in a random direction?

Code on **mc_RightWall_layer2** in the **onClipEvent(enterFrame)** event to make a Ghost bounce off a wall and move a direction besides right:

```
if(this.hitTest(_root.pGhost)) {  
    _root.pGhost._x -= 10  
    // the following randomly returns 0, 1, 2  
    randNum = Math.round(2*Math.random())  
    if (randNum == 0) { // reverse direction, move left  
        _root.pGhost.xmove = -3  
    } else if (randNum == 1) { // move up  
        _root.pGhost.ymove = -3  
        _root.pGhost.xmove = 0  
    } else { // move down  
        _root.pGhost.ymove = 3  
        _root.pGhost.xmove = 0  
    }  
} // hitTest(_root.pGhost)
```

Rubric for Flash Pac-Man game (100 points):

Name:		File Location/Name:	
-------	--	---------------------	--

- Intro Screen – name, date, play instructions. (10 pts)
- Design – attractive, complete, no errors, follows accepted design principles, good color combinations (10 pts)
- Pac-Man - animated and is controlled by the arrow keys. (10 pts)
- Walls – movie clips are used for the walls, Pac-Man cannot pass through the walls. (15 pts)
- Maze – the walls are used to create a complete, challenging, and interesting maze for Pac-Man to navigate. (10 pts)
- Dots – spaced evenly in the corridor of the maze, no dots placed outside of the corridors. When Pac-Man contacts a dot, the dot is removed and points are added to the score. (15 pts)
- Sound – 1) Add background music or sounds 2) Play a sound each time Pac-Man eats a dot. (10 pts)
- Extra features – to receive the full points, add a feature not mentioned in the above list. (20 pts) **Ideas:**
 - Timer that ends the game after a certain time has elapsed
 - Lives – remove a life when a condition occurs, such as hitting a ghost, hitting a special wall, or running out of time
 - Ghosts – that move randomly around the maze and chase Pac-Man
 - Levels – provide additional levels after the player completes the initial maze.

Rubric for Flash Pac-Man game (100 points):

Name:		File Location/Name:	
-------	--	---------------------	--

- Intro Screen – name, date, play instructions. (10 pts)
- Design – attractive, complete, no errors, follows accepted design principles, good color combinations (10 pts)
- Pac-Man - animated and is controlled by the arrow keys. (10 pts)
- Walls – movie clips are used for the walls, Pac-Man cannot pass through the walls. (15 pts)
- Maze – the walls are used to create a complete, challenging, and interesting maze for Pac-Man to navigate. (10 pts)
- Dots – spaced evenly in the corridor of the maze, no dots placed outside of the corridors. When Pac-Man contacts a dot, the dot is removed and points are added to the score. (15 pts)
- Sound – 1) Add background music or sounds 2) Play a sound each time Pac-Man eats a dot. (10 pts)
- Extra features – to receive the full points, add a feature not mentioned in the above list. (20 pts) **Ideas:**
 - Timer that ends the game after a certain time has elapsed
 - Lives – remove a life when a condition occurs, such as hitting a ghost, hitting a special wall, or running out of time
 - Ghosts – that move randomly around the maze and chase Pac-Man
 - Levels – provide additional levels after the player completes the initial maze.

Extra Day 2 - Flash – Nested Loops

Question of the Day:

- What is a nested loop?
- What is **attachMovie()** used for in ActionScript?
- What is **_root** in ActionScript?
- What is **_parent** in ActionScript?

Objectives:

- Answer Question of the Day.
- **Demo:** use **attachMovie()** and nested loops to place multiple instances of a symbol on the Flash stage.
Ex. = [Pac-man-Final fla](#)

Assignment:

- Open **pac-man-yourName fla**
- Use nested loops and the **attachMovie()** method to add dots to the stage.
- **Hint:** use the code below to complete this assignment.

Answer to Question of the Day:

- **Nested Loops** are loops within loops. Nested loops can be used to place multiple instances of a symbol on the Flash stage, in a row by column configuration.
- **attachMovie()** is used to attach movie clip instances at run-time. The **attachMovie()** method attaches an instance of a movie clip symbol in the SWF file's library to the stage. Before **removeMovieClip()** can be used, the instances must be added to the stage with **attachMovie()**.

Example Code: (placed in frame 1 of Actions layer)

```
_root.onLoad = function() {  
    dotDepth=1 // depth of movie clips must be unique  
    for(i=0; i<=15; i++) { // rows, y-axis  
        for(j=0; j<=15; j++) { // columns, x-axis  
            name = "dot" + i + j  
            _root.attachMovie("mc_dot", name, dotDepth)  
            _root[name]._y = i*25  
            _root[name]._x = j*25  
            dotDepth +=1  
        } // end column loop  
    } // end row loop  
} // end _root.onLoad
```

- **_root:** The **_root property** specifies or returns a reference to the root movie Timeline. If the currently executing movie has multiple levels, *the current level is the _root reference*.
- **_parent:** The **_parent property** specifies or returns a reference to the movie clip or object that **contains** the current movie clip or object. The current object is the object containing the ActionScript code that references **_parent**. Use **_parent** to specify a relative path to movie clips or objects that **are above** the current movie clip or object.