



MAKE AN ESCAPE THE ROOM GAME IN FLASH CS4

In this tutorial, you will learn how to create an 'Escape the Room' game in Flash CS4. An 'Escape the Room' game is one where the player finds themselves locked in a mysterious room. They must then explore the room to locate objects that will enable them to escape the room.

The 'Escape the Room' game genre was made popular by the Japanese game 'Crimson Room' that was released on the internet in 2004. This game is available on the internet and can be quite challenging to play.



This tutorial will show you how to create a simplified version of this type of game. You will create a collection of objects that can be moved throughout the room and will hide objects in different locations within the room. Once you have collected the three objects, you will be able to escape the room through the door. On completion of this tutorial, you will have learnt how to:

- create symbols in Flash and give them instance names
- add actionscript to the timeline
- change the properties of instances used in the game
- create and set up navigation between multiple scenes

Whilst the game we will build is quite simple, once you have developed these skills, you'll be able to extend the game or create other games with more complexity. You can also use these skills to create adventure games with multiple rooms and locations. You could also enhance your game by building in an interesting narrative, adding sound or using more intricate or detailed graphics. An example of the game you will create is located in the Begin the Adventure kit on the Flash Classroom website.

PLANNING YOUR GAME

To be able to work through this tutorial successfully, you will need to plan your game carefully. You can be creative in developing a context for your room, however please limit your game to having only the following features at this point to enable you to have a working game at the conclusion of this tutorial. Develop a rough storyboard for your game, ensuring that you include all of the following.

In the main room scene the player must escape from, you will have:

- a background containing the walls and objects that can't be moved
- a locked door
- three objects that the player must locate to be able to open the door
- a collection of objects that the player can drag and drop to other locations
- a space where inventory items (the items the player locates) can be displayed

In addition to this, you will have an introduction scene that contains a short narrative sequence containing four to five lines e.g. You have woken up in a strange room. The door is locked. It is dark. Can you escape the room?

You will also need to design the frame that the player reaches once they escape the room. We will keep this simple at this point by just having some text that says something like 'You've escaped the room' and a button that enables the player to start again.





FLASH CLASSROOM TUTORIAL—MAKE AN ESCAPE THE ROOM GAME IN FLASH CS4

Now that you have your plans in place, we'll start building the game in Flash.

PART 1—MAKING THE ROOM BACKGROUND

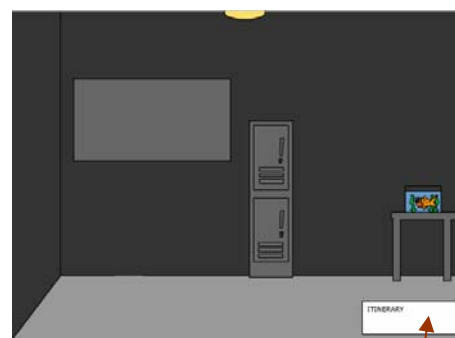
Let's begin by creating the main room for the game that the player must escape from. We will initially create a background layer where we will draw and place all of the features of the room that are not interactive.

1. Open a new document in Flash by selecting **File > New > Flash File (Actionscript 3.0)**.

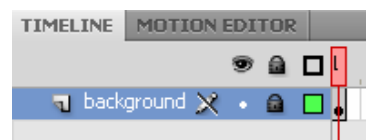
2. In the timeline, double click on the text **Layer 1** and rename the layer **background**.

3. Use the draw tools at the left of the stage to **design** the floor and walls for the room. On this layer, you can also include any objects that will not be interactive. In my example shown above, these objects include a notice board, a locker, a light, a table, a goldfish bowl and a white rectangle for the inventory. The inventory will hold the items the player finds in the room.

4. Once you are happy with your background, **lock the background layer** by clicking on the dot on the background layer that is underneath the lock icon.



The room containing the inventory.



PART 2—MAKING THE DOOR FOR THE ROOM

At this point, we are focusing on creating the visual elements or objects in the room. We will add the script that will make these interactive at a later stage. Let's begin by designing the door for our room.

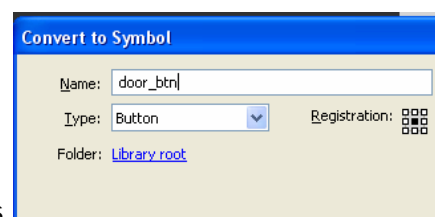
5. Create a new layer by selecting the **Insert Layer** button at the bottom left of the timeline.



6. **Rename** this layer door by double clicking on the text Layer 2 and typing in the word door.

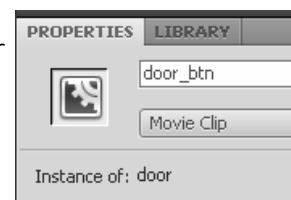
7. Use the **draw** tools to design the door. Once you are happy with the look of your door, select the door and press **F8** to open the **Convert to Symbol** box.

9. Type in the name **door_btn** and select **Button** for the type. Click on the **centre square** in the middle row of the registration option.



Note that the `_btn` part of the name is what we call a naming convention. These are used by developers in Flash to easily keep track of what type of symbol the object is. In this case, `_btn` indicates it is a button. If it was a movieclip, we would have called it `door_mc`. This isn't essential, but is good practice.

10. Our final step involves allocating an instance name for the door. When we add script to our game to make it interactive, we refer to the objects using their instance name. So even though we have named our objects when we converted them to symbols, we need to give them an instance name. To give the door an instance name, click on the door and in the **instance name cell** of the **properties panel**, type in the name **door_btn**.



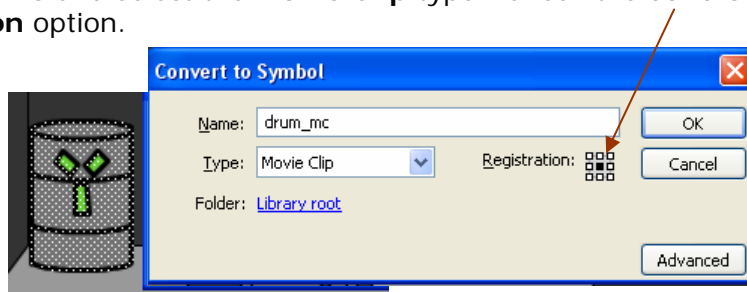


PART 3—MAKING THE DRAGGABLE OBJECTS FOR YOUR ROOM

In our simple version of this genre of game, we will be simply hiding different items behind other objects that the player will be able to drag out of the way. In my game, shown on the first page, these objects include drums, crates, a bin and a safety guide that is pinned on the notice board. Note that even though there are three drums, I have only created one drum symbol and then copied it twice. I have then given each drum a different instance name e.g. drum1_mc, drum2_mc and drum3_mc.

To create your draggable objects, follow these steps.

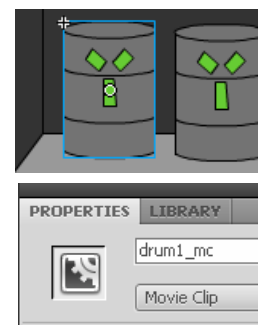
11. Create a new layer and **rename** the layer **objects**.
12. **Draw** a picture of each of the objects you want to have in the room for the player to drag. If you are going to have four crates that are identical, just draw one at this stage, we will copy it after we have converted it to a symbol.
13. Select each object and press **F8** to convert your object to a symbol. Give your object a **name** and select the **Movie Clip** type. Check the **centre square** in the **registration** option.



My draggable objects are called drum_mc, crate_mc, bin_mc and guide_mc. Ensure you have converted each of your objects into a movie clip symbol prior to moving on to the next step.

14. It is now time to make copies of any of the objects you want multiple copies of. To do this, either select the object and **Copy** and **Paste** it or simply position your mouse over the symbol, hold down the **Alt** key on the keyboard and drag your mouse to the space next to your object. This creates a second copy of your object and is a very quick and efficient way of copying objects and symbols in Flash.
15. You should now have all of your objects converted to symbols. **Place** them all in the locations you want them in the game.
16. Give each object an instance name by clicking on it and then entering the name in the **instance cell** of the **properties panel**. If you have multiple copies of the one symbol, each symbol will need its own instance name.

For example, in my game I had three drums so these have been given the instance names drum1_mc, drum2_mc and drum3_mc.



Remember that when we write the script for our game, we refer to the instance names that we have given to each instance of a symbol or object on the stage.

Ensure all of the objects you want to be able to dragged have been given an instance name.

If you are up to this point, you are doing well. We are now going to create the objects that we will hide in the room.

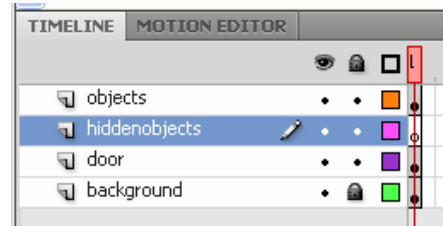




PART 4 —MAKING THE HIDDEN OBJECTS FOR YOUR ROOM AND INVENTORY

17. Create a new layer and **rename** the layer **hidden objects**.

18. Click on the layer and drag it so it is under the **objects** layer. This will ensure that your objects are positioned behind the draggable objects in the game. You may want to create additional layers at a later stage to enable the hidden objects to be between draggable objects, however for now, we will stick to just these layers.



19. Ensure the hidden objects layer is selected and **draw** a picture of each of the objects that the player will have to find in the room. Don't hide these at this stage as we have a few more things to do.

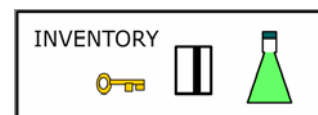
20. Select each object to be hidden and press **F8** to convert your object to a symbol. Give your object a **name** and select the **Movie Clip** type and **centre registration** option.



My hidden objects are called key_mc, swipecard_mc and potion_mc.

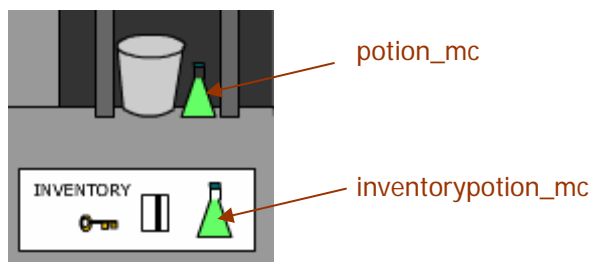
Ensure you have converted each of your objects into a movie clip symbol prior to moving on to the next step.

21. Even though you will probably have only one copy of each hidden object for the player to find, you will need to create a second copy of each object that will be hidden. This copy needs to be placed on top of the inventory area you have included in your background design.



22. We are now going to give our hidden objects and the copies we have placed in the inventory **instance names**. It is **essential to name them in the following way** as we will be setting up our script to refer to them in this way.

The copy of the symbol that will be hidden in the room needs to have the instance name that contains the 1. The copy of the symbol that is in the inventory, needs to be given the instance name that contains the 2.



For example, in my game the potion is hidden behind the bin. It has the instance name **potion_mc**.

The other copy of it in the inventory is called **inventorypotion_mc**.

To add the instance names, click on each symbol and enter the **instance name** in the **instance name** cell of the **properties panel**.

Now when the player starts the game, we want the inventory items to be empty as the player will not have found any items. We will set the items in the inventory to be invisible when we add the script for our game in the next step.

Before we move on, make sure you save your work. Continue to save your work regularly as you work through this tutorial.





PART 5 - ADDING THE ACTIONSCRIPT

We are now ready to add the script that will make our game interactive. In this tutorial, we will be adding all of the script for the game scene to the first keyframe on a layer we call Actionscript in the timeline.

We are going to add the actionscript to the timeline for a couple of reasons.

1. It's more efficient as we can go to the one place to change any of the script.
2. Flash CS4 doesn't allow you to add script to symbols. The version of actionscript we are using is Actionscript 3.0 and this only permits script to be added to the timeline.

Let's get started with scripting

The following steps will take you step by step through the script used in this tutorial. It will guide you through the process of adding all the script to your game and explain what each part of the script does. The base .fla file for this game is also available in the 'Begin the Adventure Resource Kit' on the Flash Classroom site. You can deconstruct this file to see how it works. I have also added comments to this file to explain how the script works.

23. Add a **new layer** to your timeline and rename this **Actionscript**.
24. On this layer, select the first keyframe and press **F9** to open the **Actions Panel**. This is where we will add the script to our game.
25. To begin with, we are going to add a line of script that tells the Flash Player to stop at this point in the timeline so that the player can play the game. To do this type in the following script. Take care to ensure you use the correct brackets. The ones used in this line are the ones above the 9 and 0 keys on the keyboard.

```
stop();
```

26. We are now going to add some script where we make the items in the inventory invisible to start with. We will do this by setting the 'alpha' property to 0. The alpha property relates to how transparent or see through an object is. An object given an alpha property will be completely invisible, whereas an object with an alpha property of 100 will be completely visible. Alpha properties between 0 and 100 will be somewhere in between, with the bigger the number, the more visible the object.

Make the items in your inventory invisible by adding the following line of script **for every item** you have in your **inventory**. With each line you add, replace the name of the item with the instance name you have given your inventory object. (These items were all given instance names beginning with inventory).

```
inventorykey_mc.alpha = 0;
```

Don't forget to change this part in each line to the instance name you have given your inventory object.





FLASH CLASSROOM TUTORIAL—MAKE AN ESCAPE THE ROOM GAME IN FLASH CS4

Once you have created a line for each of your inventory items, your script should look something like similar to this.

```
inventorykey_mc.alpha = 0;  
inventoryswipecard_mc.alpha = 0;  
inventorypotion_mc.alpha = 0;
```

27. Press enter a few times to space out your script to make it easier to view and edit. We will now add another line of script for each hidden item.

These lines will contain some more Event Listeners. These will check if the player has found (placed their mouse button down) over any of the hidden objects. When the listener 'hears' the player put their mouse down on the object, it will run the objectFound function. Basically this function will check if we have found all the items, show the found item in the inventory and then enable the door button if all the hidden items have been found.

We will add this function soon, but for now just **add the following line for all of your hidden objects.**

Remember to **replace the instance name** in each line with the instance name of your own hidden object.

```
key_mc.addEventListener(MouseEvent.CLICK, objectFound);
```

Your script should look similar to this:

```
key_mc.addEventListener(MouseEvent.CLICK, objectFound);  
potion_mc.addEventListener(MouseEvent.CLICK, objectFound);  
swipecard_mc.addEventListener(MouseEvent.CLICK, objectFound);
```

28. In one of the functions we write soon, we will be getting Flash to check if we have found all of the hidden items. For this to work, we need to set up a variable that will track how many items have found. This sounds tricky if you haven't dealt with the concept of variables before, but is quite easy. In our case, our variable will be called **founditems**. We will set it up to have a value of 0 to begin with - meaning that zero items have been found. In the function we will add in the next step, we will have Flash add 1 to this value each time it is run.

Add the following script for now to declare the variable and to set the initial value to zero.

```
var founditems:Number = 0;
```

Check Up Time - What have we achieved so far?

Let's do a quick recap. So far we have added script that:

- keeps the player trapped in the room (the stop(); action);
- made the items in the inventory invisible (by setting their alpha to 0);
- lets Flash know if the players has found the hidden items (by using an Event Listener);
- set up a variable to keep a count of the number of hidden items found.





ADDING THE FUNCTIONS

Let's move on and create the functions that we have referred to in the script we have already entered.

What are functions?

Functions are like mini-programs that we write within our main program (script). They can be 'called' or 'used' over and over again. Programmers use functions to make their programs more efficient. Rather than writing code over and over to do the same thing, you can set up a function and then call it when different events occur.

Let's have a look how this works within the game we are creating.

29. To begin with, let's add the two functions that will make the draggable objects able to be dragged and dropped.

To do this, press **F9** to open the **Actions Panel** and **add the following lines** of script.

```
function pickupObject(event:MouseEvent):void {
    event.target.startDrag(true);
}

function dropObject(event:MouseEvent):void {
    event.target.stopDrag();
}
```

You have just added our first two functions. The first function called `pickupObject` tells Flash to start dragging the object when a mouse event occurs. The second function called `dropObject` tells Flash to stop dragging the object when a mouse event occurs. Even though you have entered this script, the objects aren't able to be dragged yet. That's because we haven't set up the event listeners that tell Flash what mouse events to listen for and what functions to run. At present, you can click all you want on the objects but you won't be able to drag and drop them because basically the Flash player isn't listening.

To make Flash listen, we need to add the following lines of script for each draggable object. In an earlier step, you have created and given each draggable object an instance name. In my game, these objects are named `crate1_mc`, `crate2_mc`, `drum1_mc`, `drum2_mc` etc. You created and named your draggable objects in steps 11 - 16.

Enter the following script in your **Actions Panel (F9)** and **copy and paste it** until you have a copy of these three lines for each draggable object. **Change the instance names** at the start of each line to match those you have given to your draggable objects.

```
crate1_mc.addEventListener(MouseEvent.MOUSE_DOWN, pickupObject);
crate1_mc.addEventListener(MouseEvent.MOUSE_UP, dropObject);
crate1_mc.buttonMode = true;
```





FLASH CLASSROOM TUTORIAL—MAKE AN ESCAPE THE ROOM GAME IN FLASH CS4

Once you have copied, pasted and edited the three lines of script for each of your draggable objects, your script should look very similar to that shown here. You will ofcourse have three lines of script for every draggable object.

```
crate4_mc.addEventListener(MouseEvent.CLICK, pickupObject);
crate4_mc.addEventListener(MouseEvent.CLICK, dropObject);
crate4_mc.buttonMode = true;

drum1_mc.addEventListener(MouseEvent.CLICK, pickupObject);
drum1_mc.addEventListener(MouseEvent.CLICK, dropObject);
drum1_mc.buttonMode = true;

drum2_mc.addEventListener(MouseEvent.CLICK, pickupObject);
drum2_mc.addEventListener(MouseEvent.CLICK, dropObject);
drum2_mc.buttonMode = true;

drum3_mc.addEventListener(MouseEvent.CLICK, pickupObject);
drum3_mc.addEventListener(MouseEvent.CLICK, dropObject);
drum3_mc.buttonMode = true;

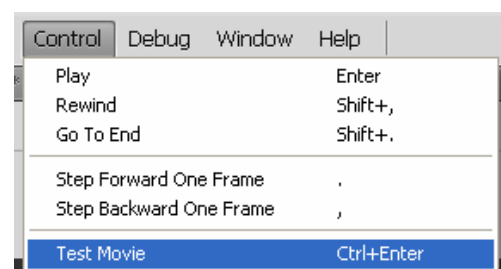
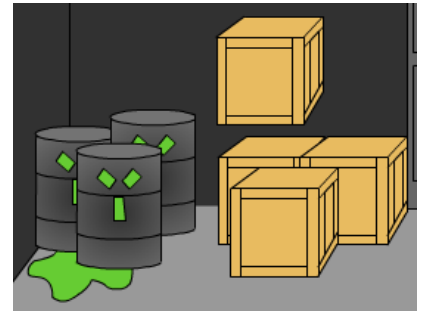
bin1_mc.addEventListener(MouseEvent.CLICK, pickupObject);
bin1_mc.addEventListener(MouseEvent.CLICK, dropObject);
bin1_mc.buttonMode = true;
```

31. You can now test your game to see if your objects can be dragged and dropped.

To do so, select **Control > Test Movie** and then try to move your objects around the room.

If you can't go back and work through the steps and check that you have given your objects instance names and that your script matches these.

Also double check that your script matches mine. Flash is unforgiving. If you use the wrong brackets or have the smallest typo - it will prevent the script from doing what you want it to do.



32. Let's add another function to our script. This function will be called when the player clicks on the door after they have found all three hidden objects. This function tells the Flash player to go to and stop at frame 2 in the game scene. We'll add frame 2 later. This is the frame that will tell the player they have escaped the room.

```
function escapeRoom(event:MouseEvent):void {
    gotoAndStop(2);
}
```





Let's keep going....

If you've got this far, the good news is that we have nearly finished adding the script for the room scene of our game. If you are new to scripting, don't feel intimidated by writing script. After you work with it for a while, you begin to understand what you are writing and how it works. Initially, you will probably be just copying and typing it in and making small changes. This is all part of the learning process so don't feel disheartened.

- 33.** We have a third function to add at this point. This function is the `objectFound` function that is called when any of our hidden objects are clicked on by the player. We wrote the following three lines of script to make the `objectFound` function run when the mouse is clicked on any of the hidden objects.

```
key_mc.addEventListener(MouseEvent.CLICK, objectFound);  
potion_mc.addEventListener(MouseEvent.CLICK, objectFound);  
swipecard_mc.addEventListener(MouseEvent.CLICK, objectFound);
```

Now **lets add the `objectFound` function**. Note that we only have to copy the script below in once. This is the great thing about using functions. Flash will work through the script below anytime the user clicks on any of the hidden objects. **Type in the following script** in your **Actions Panel (F9)**.

```
function objectFound(event:MouseEvent):void {  
    var inventoryName:String = "inventory" + event.target.name;  
    var inventoryItem:DisplayObject = getChildByName(inventoryName);  
    event.target.alpha = inventoryItem.alpha = 100;  
    founditems ++;  
    event.target.alpha = 0;  
    if(founditems == 3){  
        door_btn.addEventListener(MouseEvent.CLICK, escapeRoom);  
    }  
}
```

Make sure you double check the script to make sure it is correct.

I'm guessing you're not quite sure what the script does. Let's break it down line by line. The first line tells Flash that this is a function called `objectFound` and that the function will be called by a mouse event (e.g. the user clicking on the object they have found).

The next four lines work together by telling Flash the instance name of the object that was found and then the instance name of the matching item in the inventory. Once Flash has worked this out, it makes the alpha of the inventory item 100 (visible) and the alpha of the object that was clicked on 0 (invisible). This means that when the player finds an object it will look like the user has picked it up and placed it in the inventory.

The sixth line is pretty simple. It tells Flash to add 1 to the `founditems` variable we set up. The remaining lines work by checking if all three of the hidden objects have been found and if so tell Flash to listen for the mouse to be pressed down on the door and to run the `escapeRoom` function when it is.





FLASH CLASSROOM TUTORIAL—MAKE AN ESCAPE THE ROOM GAME IN FLASH CS4

PART 6 - ADDING THE 'YOU'VE ESCAPED' FRAME

We've just completed all the script for the room frame. As part of this script, we wrote a function called `escapeRoom` (shown below).

```
function escapeRoom(event:MouseEvent):void {  
    gotoAndStop(2);  
}
```

In this function, we told Flash to go to and stop at frame 2. This will happen when the player has found all of the objects and can escape the room.

34. Click on frame 2 of the **Background** layer and press **F6** to **add a new keyframe**.
35. Use the text tool to add 'You've escaped the room.' or another similar message.



You've escaped the room.

36. Select **Control > Test Movie** to test your game. You should now be able to drag your items, find and click on the hidden objects and then click on the door to escape the room.

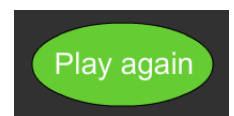
Now at this stage you are probably thinking that the game is a bit easy to play. Once you complete this full tutorial, you can go back and modify your game to make it a bit harder to escape the room. You could do this by creating more objects that the user can locate and interact with or by having multiple rooms that the player has to escape from.

PART 7 - ADDING A PLAY AGAIN BUTTON

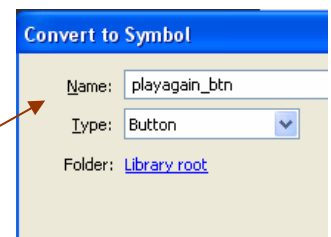
At present, there is no way that that the player can replay the game. To solve this problem, we are now going to design a button and add some script to enable the user to go back to the start of the game.

To save time, we won't set up the button to change on rollover. If you know how to edit a button and change the button states, feel free to do so. However, in this tutorial, we'll just move on to adding the script to make the button work.

37. Click on **frame 2** of the **background layer**.
38. In this frame, draw the design for your button. My button is a simple green oval shape which contains the words 'Play again'. (shown here)
39. Select the button and press **F8** to convert your design into a button symbol. Give your button the name **playagain_btn**.
40. We also need to give your button an **instance name** so that we can add some actionscript to make it work.



Click on your button and then in the **Properties** panel, enter **playagain_btn**.





FLASH CLASSROOM TUTORIAL—MAKE AN ESCAPE THE ROOM GAME IN FLASH CS4

41. It's now time to add the script that will take the user to the first frame in the intro scene. We haven't created this scene yet but will do so in the next part of this tutorial.

For now, **click on frame 2** of the Actionscript layer, press **F6** to add a new keyframe and then press **F9** to open your actions panel. Enter the following script:

```
playagain_btn.addEventListener(MouseEvent.CLICK, playAgain);

function playAgain(event:MouseEvent):void {
    gotoAndPlay(1, "intro");
}
```

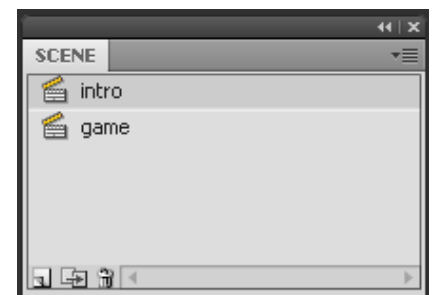
There's no point testing our button as we haven't created our intro scene yet. Move on to the next part of this tutorial to do this.

PART 8 - ADDING THE INTRO SCENE TO THE GAME

It does seem a little backward, but the final part of making our game involves adding a scene that will form the introduction to the game. The reason I have left this to last is because I found that it is more effective to get your main room done first and also, it's a bit painful running through the introduction each time you want to test the game.

At the beginning of this tutorial you were asked to do some planning. As part of this, you were asked to come up with a short narrative of about 4 - 5 lines in length that could be used as part of the introduction. We are going to add each of these lines to different keyframes in our intro scene so that they appear one at a time. The result will be a short sequence that will create a mood for the game. After viewing this sequence, the player will automatically end up trapped in the room scene.

42. Let's start by setting up a new scene. To do this select **Windows > Other Panels > Scene** (or **Shift + F2**).
43. Add a **new scene** by selecting the plus sign button at the bottom of the panel and **rename** the new scene by double clicking on the text. Name the new scene **intro**.
44. Click on this scene and drag the **intro scene** to the top of the panel. This will mean it will be the **first scene** that the player views.
45. **Double click** on the words Scene 1 in the Scene panel and **rename** the scene **game**.
45. Select this scene and click on the first layer and rename it **text**.
46. Select the **first keyframe** on the timeline and in the centre of the stage, enter the **first line** of your narrative.



My first line is shown in the screenshot below.

YOU HAVE WOKEN IN A STRANGE ROOM.





FLASH CLASSROOM TUTORIAL—MAKE AN ESCAPE THE ROOM GAME IN FLASH CS4

We will now add keyframes at points along the timeline and change the text until we have added all of the lines of our narrative.

47. Click on **Frame 50** and press **F6** to add a new keyframe. Replace your first line of text with your second line.
48. **Repeat this process** by adding another keyframe at say frame 100 and adding the third line, adding another keyframe at 150 and adding the fourth line and so on.
49. Remember to add a final keyframe about 50 frames after your final line. This will ensure this line stays visible long enough for the player to read the text.

You may need to alter the position of the keyframes to enable the text to be read. To do this, you can drag and drop the keyframes to different frames of the timeline to adjust how long each sentence or line is visible. It takes a bit of trial and error and you will need to test your movie to see if the timing is right.

PART 9 - TESTING YOUR GAME

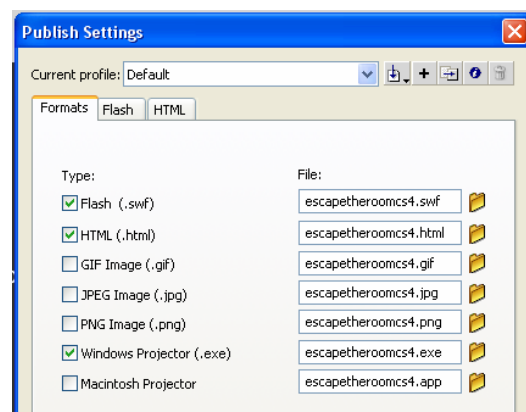
You've done it. You've completed your game. It's now time to test it. Select **Control > Test Movie** and watch your intro, escape the room and then restart the game.

Does your game work as it should? Are you stuck in the room? Can you drag and drop the objects and locate the hidden items? Are you able to escape once the objects are found?

If so, well done. You have created your first simple 'Escape the Room' game. If not, check through your game, paying particular attention to your instance names and the script. If you have done this and you still have problems, get your teacher to have a look or email me at the address below.

PART 10 - PUBLISH & SHARE YOUR WORK

50. Save your work by selecting **File > Save**.
51. Turn your flash file into a game that can be played on any computer by publishing it in different file formats. To do this select **File > Publish Settings**. The box to the right will appear.
52. Tick the file formats you want and click on the **Publish** button. These files will be saved in the same location you saved your original file. If you want your game to be a standalone file that can be played on Windows or Macintosh machines—ensure you check the Windows Projector (.exe) and Macintosh Projector (.exe) format options.



Congratulations! You have created and published your first game in Flash. Check out the Flash Classroom to see what other teachers and students have created by using this resource.

