



## MAKE A DODGE-EM STYLE GAME IN FLASH MX2004

In this tutorial, you will learn how to make a dodge-em style game where the player must move the mouse to avoid being hit by objects that are randomly moving across the screen.

An example of this type of game is the 'Dodgy Dan' game that is located in the Game Design section of the Flash Classroom gallery.

In the Dodgy Dan game, the user has to help move Dodgy Dan out of the way of the enemy fire that is streaming across the screen. Each time Dan is hit, he loses a life. The game ends when one of the following occurs:

- 1) Dodgy Dan loses all nine of his lives;
- 2) The player helps Dan avoid the fire for a period of 45 seconds.

This is a really good tutorial for beginners to game design in Flash because you will learn lots of useful tricks including how to make a simple timer for your game and how to make objects appear and move randomly.

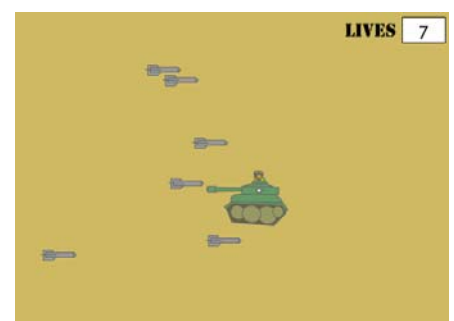
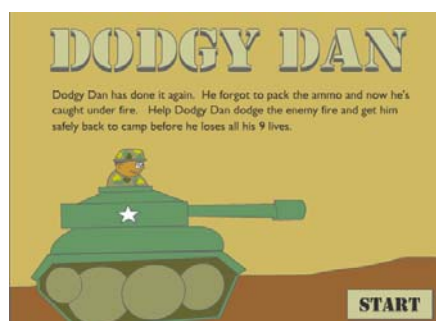
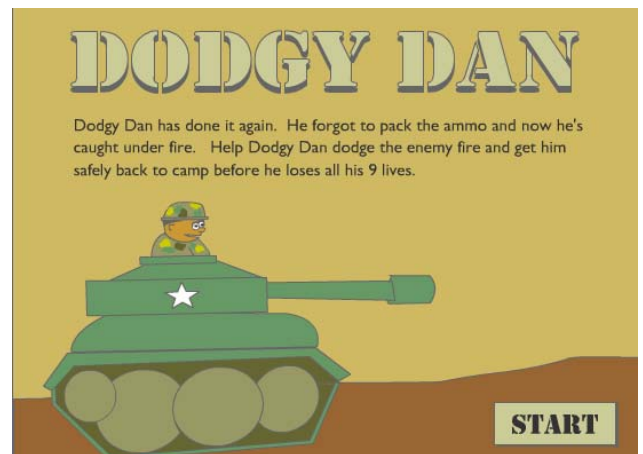
### Pre-Game Design Task

1. Take a few minutes now to play the Dodgy Dan Game in the Flash Classroom gallery.
2. Now that you understand the game play involved in Dodgy Dan, take a few moments to consider what other games could be created using this same approach. Brainstorm a scenario for atleast one other game that requires the user to avoid being 'hit' by moving objects.

### Features of the Dodgy Dan Game

The game you will learn to make in this tutorial will have the same features as the 'Dodgy Dan' game. This includes a title scene, a scene containing the actual game, the game over scene and the you made it scene.

These four scenes are shown to the right.





## PLANNING YOUR GAME

It's time for you to start planning the game that you will build by following this tutorial. As this tutorial only covers a certain amount of content, you will need to ensure that the game you are thinking of building initially only has the features listed below. Although, the sky's the limit with what's possible in game development in Flash, this tutorial will only introduce you to game design—so make sure you remember that your game will only have the features below—otherwise you'll be disappointed.

### FEATURES OF YOUR GAME

1. A **Title Scene** to introduce the game. The user will click a button on this scene to enter the Game Scene. This scene will feature the name of the game, instructions on how to play and the name of the author/s.
2. A **Game Play Scene** which contains a background, the objects that the player must dodge, the object that will represent the player (e.g. Dan) and a dynamic text box that shows the number of lives the player has left. In this scene, a timer movieclip will sit off stage. This clip will time the number of seconds the player has been playing the game.
3. A **Game Over Scene** which contains feedback that the user has been unsuccessful at completing the game. It also contains a replay button.
4. A **You Made It Scene** which contains feedback that the user has been successful at completing the game. It also contains a copy of the replay button.

Your game can be enhanced dramatically through the use of sound. The Dodgy Dan game contains a sound loop to create ambience and a metal bang sound effect that is played when Dan is hit by enemy fire. You may want to take some time in the planning stage to locate sound loops / effects that will suit your game. A good place to start is the Flashkit site at [www.flashkit.com](http://www.flashkit.com). This site features a library of loops and sound effects that can be downloaded for use in your files. Most of these are freeware or shareware.

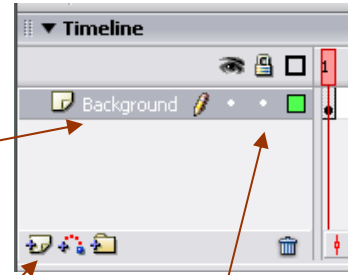
Now that you're aware of the features your game can have, start planning by drawing some storyboards / layouts of each scene on paper. Think of a catchy name for your game and write the instructions for your game. You may even want to do some rough sketches of the characters / objects in your game.

Once you've got all this worked out—you're ready to start creating your game. The following pages will show you step-by-step how to do this.



## SETTING UP YOUR TITLE SCENE

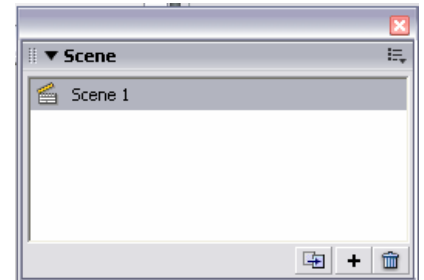
1. Open a new Flash file by selecting **File > New**.
2. Double-click on the text Layer 1 on the first layer of your timeline and type in **Background**.
3. On this layer, **draw** your background picture. Once you are happy with your background, lock this layer by clicking on the dot underneath the lock icon.
4. Make a new layer by clicking on the **Add Layer** button.
5. Rename this layer **Text** by double clicking on the words Layer 2. Add your title and any other text on this layer. **Lock** this layer once you are happy with it.
6. You may want to have a picture of one or more of the games characters / objects on the title page. In the case of my game, 'Dodgy Dan' I have a picture of Dan in his tank. If you do, make a new layer by clicking on the Add Layer button and rename this layer **Characters**. Draw or copy your character onto this layer.



That's nearly everything for the title scene—however, at the moment there isn't any way for the user to get to the next scene—or for Flash to know to stay on the title scene until the user clicks a button. We are going to fix this in the next sections by setting up multiple scenes for the game and adding buttons.

## SETTING UP MULTIPLE SCENES

7. To begin with, we're going to open up our Scene panel by selecting **Window > Design Panels > Scene**. This will open the panel shown here.
8. The first thing we are going to do is change the name of the scene from Scene 1 to **Title Scene**. To do this, double click on the words Scene 1 and type in **Title Scene**.
9. Now let's set up the other scenes for our game. You have a choice at this point based on the design of your game. In my game, 'Dodgy Dan', I used different backgrounds for all of my scenes. This meant that when I started setting up my scenes, it didn't make sense to duplicate my scenes as I did not want the same background in every scene. However, if I did want to use the same background in each scene, I could select the duplicate scene option. I could then remove any elements from the Title Scene that I didn't need.

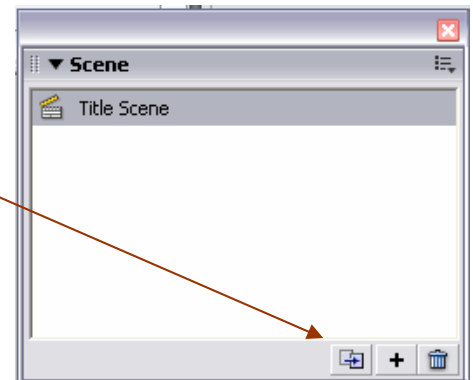


Take a moment to think about which of these options suit you.



### 9a. Option 1—Duplicate the Title Scene

If you've chosen to duplicate the title scene, you can do this by clicking on the title scene in the Scene panel and selecting the **Duplicate Scene** button shown here.



If all of your scenes are to have the same background, repeat this process 3 times.

If not, repeat it until you have the number of scenes you need with that background.

At this point, your Scene Panel should look something like this. Note that I have renamed all the duplicated scenes by double clicking on them and typing in the new names.

I encourage you to use the same names as I have so that you find it easier to follow the actionscript we write later.



If you have set up all your scenes—go to Step 10.

### 9b. Option 2— Add New Scenes

You may want each scene in your game to have a different look. That's fine too and if you choose to do this, you will now simply add blank scenes that you can design new backgrounds etc for later on.

To **add a new scene**, click on the + button at the bottom of the panel.



Rename the scenes you add by double clicking on their names and typing in your own names. I encourage you to use the same names as I have so that you find it easier to follow the actionscript we write later.

Your scene panel should look like this once you have added all four scenes.





## ADDING BUTTONS TO MOVE BETWEEN EACH SCENE

In the game we are creating, we need to add some buttons to enable the player to move between scenes.

In the 'Dodgy Dan' example, there are 2 buttons which make this possible. These are:

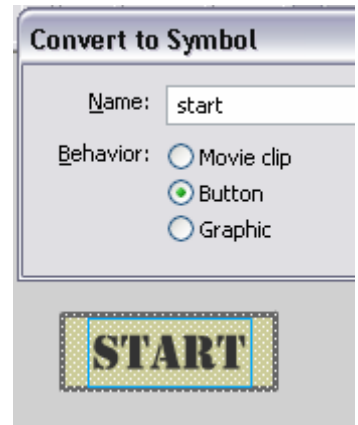
- 1 x **Start** button that is located on a layer named buttons in the **Title scene**.
- 1 x **Replay** button which is located on a layer named buttons in both the **Game Over** and **You Made It** scenes. These replay buttons will be programmed to take the player back to the title scene.

This page will walk you through how to create those buttons. The design of the buttons and type of buttons you use is up to you—you may even wish to use a button from the Common Library that Flash provides. To view the buttons in this collection, select **Window > Common Libraries > Buttons**.

The instructions below show you how to make your own buttons.

### Making Your Own Buttons

10. On the grey area to the side of the stage, draw what you want your button to look like when the player does not have the mouse over it. This is the **Up** state.
11. Select your button and select **Modify > Convert to Symbol**. Name your button and select the **Button** behaviour.
12. Click on your button and then right click to bring up the context menu. From this menu select **Edit in Place**.
13. You should notice that when editing your button, the timeline will only have a keyframe in the **Up** state. The button will still work when this is the case, however, you may want to change what the button looks like in the other states. For example, it is common for a button to move slightly or change colour when rolled over.

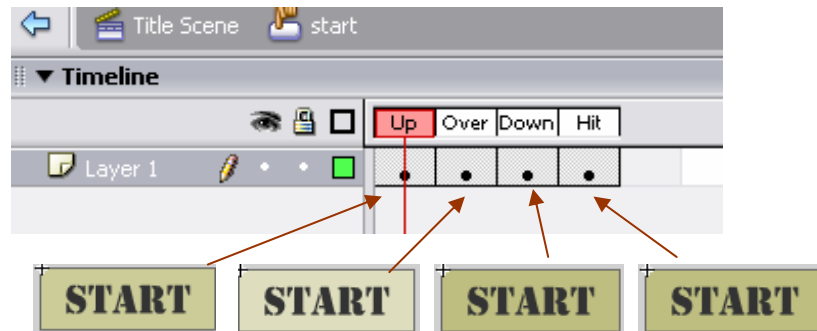


Add **Keyframes** to each of the other button states (**Over**, **Down** and **Hit**) by clicking on each frame and selecting **Insert > Timeline > Keyframe**.

In each keyframe, change what you want the button to look like. The hit state keyframe usually doesn't need to be edited. This state is usually only needed if you want to create an invisible button or say you have an image or text in the other states that is intricate. When this is the case, you may wish to draw a bigger shape over the text or image in this state so that the button is easy for the user to hit. For example, if you didn't do this for a button containing only text, the user would actually have to put their mouse over the lines of the letters to make the button work. Drawing a shape in the hit state makes the button easier to hit.



Below is a diagram that shows the four hit states of the Start button I created for the 'Dodgy Dan' game.



14. Once you are happy with your button states, click on the Scene name link at the top of the timeline to stop editing the button and return to the current scene.

### Programming Your Buttons to Work

It's now time to position the buttons on the button layers in the appropriate scenes and to add the actionscript that will make them go to another scene when clicked. Follow the steps below for each of the buttons to do this.

15. Position the button you want to program in the location you want it on the stage. To make multiple versions of the same button, simply copy and paste or open the Library using **Window > Library** and drag a copy onto the stage.
16. Click on the button you want to add the script to and select **Window > Actions** or press **F9**. This will open the **Actions Panel** where you will type in the script on the following page.
17. Type the following script into the Actions Panel. Note that the scene name has been put in **blue bold** to remind you that you will need to edit this name in the next step.

```
on (release) {  
    gotoAndPlay("Game Scene",1);  
}
```

This script tells Flash that when the player clicks on the button and then releases it, that Flash needs to go to and play another scene. In this case, the Game Scene. The number 1 indicates that it should go to frame 1 of that scene.

18. Edit the script by changing the name of the scene to the name of the scene your button is going to take the player to. Ensure that your replay buttons take the user back to the title scene. This is essential because we will add script later that will mean that our number of lives will be reset when the user enters this scene.



## Adding Stop Actions to Scenes

If you were to test your movie now you would find that your movie automatically jumps from one scene to another. This is because Flash naturally plays through scene 1 and then moves to scene 2 and so on, unless it is instructed to stop.

We are going to add **stop actions** to the final keyframe in our **Title Scene**, **Game Over Scene** and **You Made It Scene**. This will mean that the Flash player will stop at these scenes until the user clicks on the buttons you have added in the previous step. Note that we don't want to add a stop action to the **Game** scene as we will add other scripting to this scene in a future step.

19. To add a stop action to the final keyframe of each scene, click on the keyframe and press **F9** to open the **Actions Panel**.

20. Type in the stop action shown below:  
stop ();

This will trap the user in the current scene until they click on the button to go to another scene. Repeat this process for each scene the user will stop at.

## Building Your Game Scene

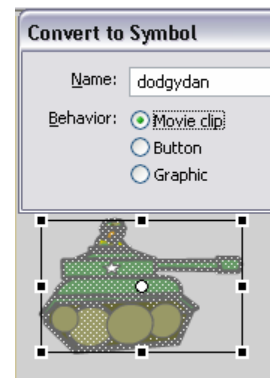
You have now completed most of the steps involved in setting up the different scenes and the navigation between them. In all of the scenes except for the **Game** scene, you really just need to change the content on the layers to include the text and objects you want.

This section of the tutorial will show you how to make the actual game play part of the game. To begin with we'll create the object that will represent the player and the objects that the player will dodge.

### Part A— Creating the Object that Represents the Player

21. On the grey area beside the stage in the **Game Scene**, draw the character/object that will represent the player. In my game, this object is Dodgy Dan (shown right).

22. Once you are happy with your design, select **Modify > Convert to Symbol**. In the panel, give your object a name and select the **Movie clip** behaviour.



23. Make a new layer for this object and give this layer the same name as the object. Ensure that the object is on this layer.

24. Click on the object and select **F9** to open the **Actions Panel**. Type in the following script. This script will hide the standard mouse cursor and will make the mouse drag this object throughout this scene.

```
onClipEvent(load){  
    Mouse.hide();  
    this.startDrag(true,0,0,Stage.width,Stage.height);  
}
```



## FLASH CLASSROOM TUTORIAL—MAKE A DODGE-EM STYLE GAME IN FLASH MX2004

In the previous steps, you have drawn the object that will represent the player and you have also added some script that will enable the mouse to drag this object. You could leave this as is, however your game could be enhanced by providing the player with instant feedback when a collision or hit has occurred.

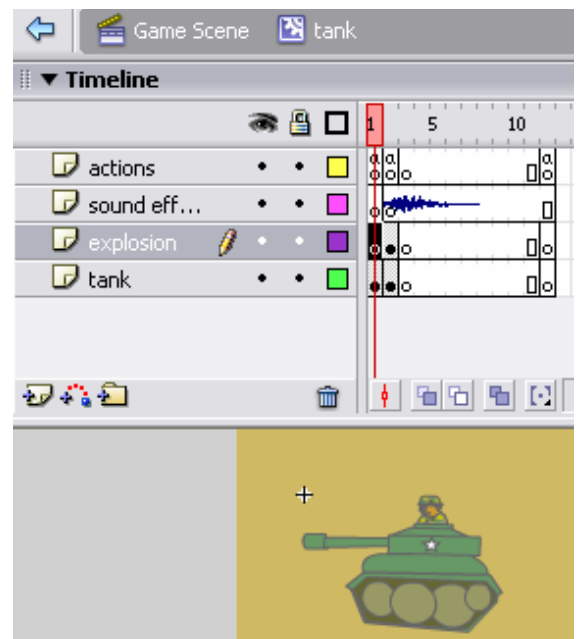
In Dodgy Dan, I have set up Dodgy Dan's tank to explode when it is hit by enemy fire. To do this, I edited the tank movieclip and added the explosion image and sound effect in Frame 2 of this movieclip. I have added a stop action to Frame 1 so that the movieclip will stay on Frame 1 until a hit occurs. We will add actionscript to our game later that will tell Flash to go to and play frame 2 of this movieclip if a hit is detected.

The steps below will take you through the process of how I did it.

25. In the Game Scene, **right click** on the Movieclip object that you just created to represent the player. Select **Edit in Place** from the menu that appears.

26. At present, your movieclip symbol will only contain a single Layer named Layer 1. Give this layer the same name as the object and then add three additional layers. These three layers should be renamed **actions**, **sound effects** and **explosion**.

The timeline shown at the right shows the order of the layers. If your order is different, drag the layers into the correct order.



27. The only visible image in frame one of this movieclip is the object you have drawn. However, this first frame will also contain some actionscript to tell the movieclip to stay on this frame. To enter this script, **select the first keyframe** on the **Actions Layer**, press **F9** to open your Actions Panel and enter the script below into this panel.

```
stop();
```

28. Our next step is to add the image and sound to the second frame so that the player will know when they have been hit. We will set up the keyframes in Frame 2 to begin with. To do this, select Frame 2 in each layer and press **F6** to add a keyframe. In the object layer (mine is called tank) make no changes. In the keyframe in frame 2 of the explosion layer, draw the explosion or other image that will be shown when a hit occurs.

29. If you have a sound effect ready, select **File > Import to Library** and locate the sound file you want to import. Open up your Library in Flash by pressing **Ctrl + L**, select the keyframe in Frame 2 of the Sound Effects layer and then drag the sound effect from the library onto the stage.



## FLASH CLASSROOM TUTORIAL—MAKE A DODGE-EM STYLE GAME IN FLASH MX2004

30. When the object is hit we also want the player to lose a life. To set this up we will add a line of code to the second keyframe on the Actions layer. **Select Keyframe 2** on the **Actions** layer and press **F9** to open the Actions Panel. In this panel type:

```
_root.lives -= 1;
```

This script will subtract 1 from the value of a variable named 'lives' that you will set up in Part C of this section.

31. We also want the object to disappear for a small amount of time once it has exploded. To do this we will set up some blank keyframes at different positions on our timeline.

On the object layer (e.g. tank), click on the third frame and select **Insert > Timeline > Blank Keyframe**. This will make the tank or object only appear in the first two frames.

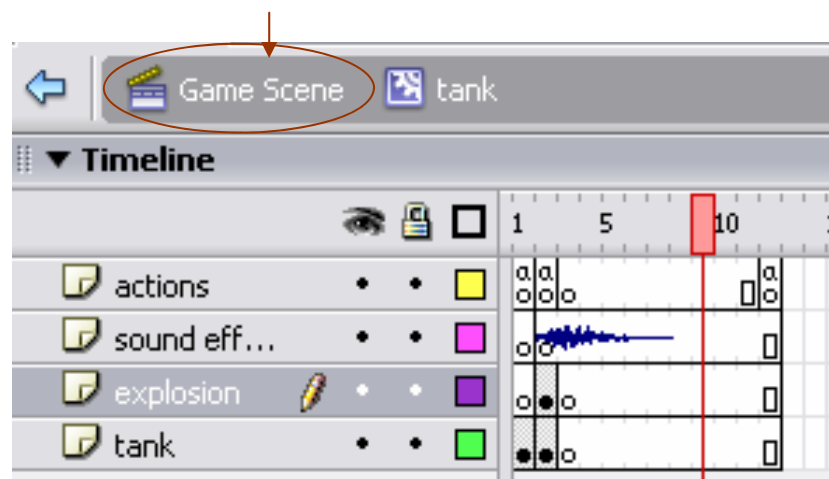
Repeat this process on the explosion layer. Click on frame 3 of this layer and select **Insert > Timeline > Blank Keyframe**. The explosion will only now be visible in the second frame of this movieclip.

32. Finally, we want to make the object or tank return to it's original look in Frame 1 so that it is ready to start dodging the objects again. To set this up we will add a new keyframe at frame 12 in the Actions layer. We will add script to this keyframe to make the movieclip go back and stop at Frame 1.

To do this, click on **Frame 12** of the **Actions** layer. Select **F6** to insert a new keyframe. Click on this keyframe and select **F9** to open up our Actions panel. In this panel, type the following text:

```
this.gotoAndStop(1);
```

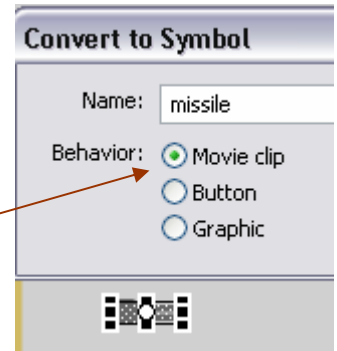
33. The timeline of your movieclip symbol should now look like the one shown below. If this is the case, you are ready to move on to the next step. **Return to the Game Scene** by selecting the Game Scene link.





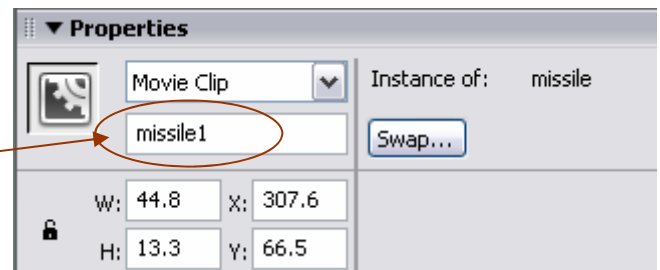
**Part B - Creating the Objects that the player will dodge**

- 34. On the grey area beside the stage in the **Game Scene**, draw the object that the player will need to dodge.
- 35. Once you are happy with your drawing, select **Modify > Convert to Symbol**. Give your symbol a name and ensure that you select the **Movie clip** behaviour.
- 36. Make a **new layer** in this **Game Scene** and give it the same name as the Movie clip you just created. (e.g. the object I created was a missile, so my layer was called missile).



- 37. Place the object on this new layer and then **copy and paste** copies of the object until you have six copies of the object on this new layer.

- 38. Select each copy one at a time and in the **Properties Panel**, give each copy a different **instance name**. These names should be exactly the same except for the number at the end. For example, my objects were missiles and I called them missile1, missile2, missile3, missile4, missile5 and missile6.

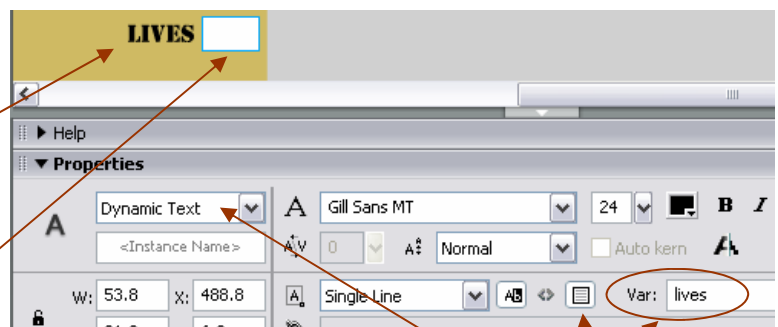


**Part C - Setting up the Player's Lives**

In this step, we are going to set up two text boxes that will enable the player to see how many lives they have left.

- 39. To begin with, select the Text tool and add a static (normal) textbox that contains the word **LIVES**.

- 40. Next to this text box, draw a second text box. This text box will be a dynamic text box which will display the number of lives that remain. Select the Dynamic Text option in the properties panel.



- 41. In the Var cell, give this textbox the variable name **lives**.

- 42. Select the **Show Border Around Text** button in this panel.





## Part D — Creating a Timer

As outlined earlier in this tutorial, there are two ways that the game can end. The first way is the player loses all their lives. This results in them being taken to the Game Over Scene. We have just laid the ground work for this option. The second way will be that the player will avoid losing all of the lives for a set time. If they manage to do this, they will be taken to the You Made It Scene.

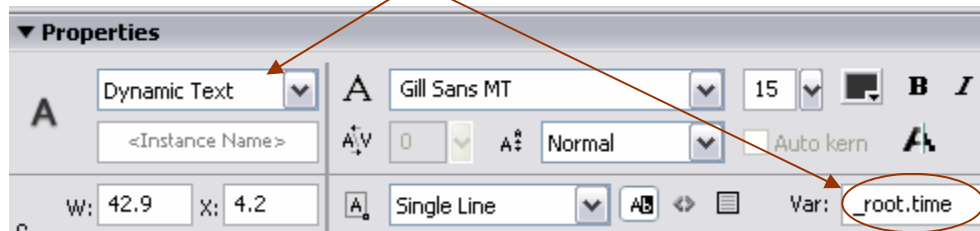
In this step, we will create a timer that will track the number of seconds that the player has been playing the game.

**43.** In the **Game Scene**, create a **new layer** and rename it **Timer**.

**44.** In the top left hand corner of the stage, use the text tool to type in the word **Timer**.

Don't worry too much about the font or colour, as the player won't be seeing this timer. You will simply use it to check that your game is working. Once you have established that this is the case, you will drag the timer off stage so that it won't be visible in the final game.

**45.** Underneath this text box, draw a **dynamic text box** and give it the variable name **\_root.time**



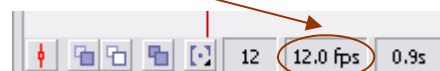
Your timer should look similar to this.



**46.** Select both of these text boxes and select **Modify > Convert to Symbol**. Enter the name **Timer** and select the **Movie clip** behaviour.

**47.** Right click on your **Timer Movie clip** and select **Edit in Place**. You will now be editing the movieclip.

What we will do now is set up this movieclip symbol so that it has the same number of frames as the number of frames per second our Flash movie is set up for. By default, this should be 12 frames per second (fps). You can check this at the bottom of the timeline (shown here).



**48.** If your number of frames per second is different to 12, double click on the number in the space shown and change it to 12.0 fps.

49. On layer 1 in your Timer Movie clip symbol, click on frame 12 and select **F5**. This will extend the frames to the 12th frame.

50. Add a **new layer** and name it **Actions**.

51. Click on frame 12 of this new layer and add a keyframe by pressing **F6**.

52. Select this keyframe and press **F9** to open the **Actions Panel**.

Type in the following script:

```
_root.time++;
```

Take a moment to look at the timeline you have setup within the Timer movieclip. What you have is a movieclip with the length of exactly 1 second. This movieclip is not set to stop and therefore once the player enters this scene, the movieclip will continue to run through these frames over and over until the player leaves this scene. The script you just added will add 1 to the `_root.time` variable each time the movieclip reaches frame 12. This variable will therefore in effect track the number of seconds that the player has been playing the game.

53. Return to the main **Game Scene** by selecting the Game Scene link above the timeline.



### Part E — Adding the Actionscript that will control the game

54. Create a new layer in the **Game Scene** and call it **Actions**. On the **first keyframe**, press **F9** and add the following script. Note that if you may need to change the words highlighted in blue if you have used different names to mine.

```
if (lives eq 0) {
    gotoAndStop ("Game Over Scene",1);
} else {
    gotoAndPlay("Game Scene",1);
}

for(i=1; i<7;i++)
{
    _root["missile"+i].onLoad = function(){
        this._x = -(Random(100)+1);
        this._y = Random(390) + 1;
    }
    _root["missile"+i].onEnterFrame = function(){
        if(this._x >= 550)
        {
            this._x = -(Random(100)+1);
            this._y = Random(390) + 10;
        }
        this._x += 20;

        if(this.hitTest(tank))
        {
            _root.tank.gotoAndPlay(2);
        }
    }
}
```

#### How the Script Works

This part of the script checks the number of lives. If there are 0 lives, the player will be taken to the Game Over Scene.

This is a loop that sets up the objects that the player must dodge. It places all 6 of them at random locations.

This is the hit test part which checks for collisions between the player object and the objects that the player must dodge. If a collision is detected, the object movieclip moves to frame 2 where an explosion occurs and 1 life is subtracted from the number of lives.



55. For our game to work, we need to add script to a second keyframe on the Actions layer.

Click on Frame 2 of the **Actions Layer** and press **F6** to add a keyframe.

56. Press **F9** to open up the Actions Panel and type in the following script.

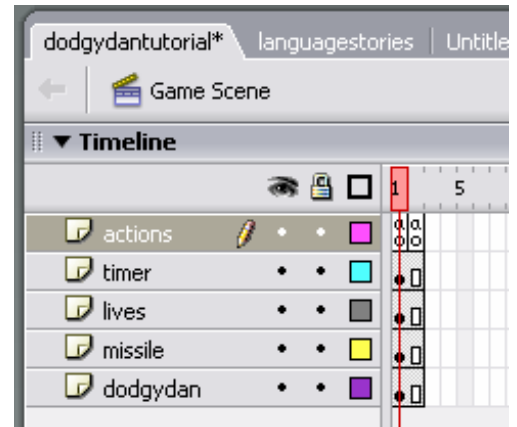
```
if (lives eq 0) {
    gotoAndStop ("Game Over Scene",1);
}
```

← This part of the script checks the number of lives. If there are 0 lives, the player will be taken to the Game Over Scene.

```
if (_root.timer._root.time eq 45){
    gotoAndStop ("You Made It Scene",1);
} else {
    gotoAndPlay("Game Scene",1);
}
```

← This part of the script checks the value of the `_root.time` variable you set up within the timer movieclip. If the value is 45, the player will be taken to the You Made It Scene. If it hasn't reached 45, the player will remain in the Game Scene.

57. The Game Scene will continually loop through these two frames until one of two conditions are met. These are the player losing all 9 lives or the player successfully dodging missiles for a period of 45 seconds. At present our second frame only contains actions on the Actions layer. We therefore need to extend the frames on our other layers so that the objects will be visible in the second frame. To do this, **click on Frame 2** in all other layers. **Select F5** to add the extra frame to each layer. Your timeline should now look like the one shown above.



### Adding the final script...

There are just two other places where we need to add script to complete our game.

58. In the Title Scene of the game, select the first keyframe in the Actions layer where you have added the `stop();` action. Select **F9** to open your Actions panel and enter the following script underneath `stop();`. This will reset the number of lives and the timer for the start of each game.

```
lives = 9;
_root.time = 0;
```

59. Finally, in the Game Over and You Made It Scenes, add the following to the first keyframe underneath the `stop();` actions. This will make the mouse reappear. This is necessary because we made it disappear in the Game Scene.

```
Mouse.show();
```



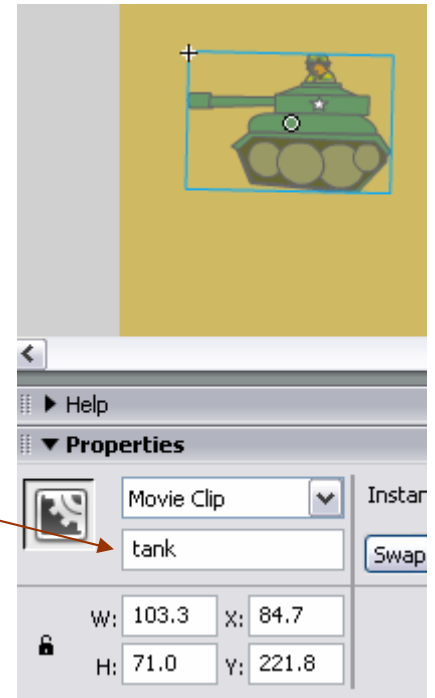
## A Final Step - Adding Instance Names

Earlier in this tutorial, we gave the objects you created to be dodged instance names. The final step we need to do to ensure our game works, is to give the following movieclip symbols in our game instance names.

- a) the movieclip symbol that represents the player (e.g. tank)
- b) the timer movieclip

**60.** In your game scene, select the object that Represents the player and in the Properties panel, give the object an instance name. Note that this name needs to match the name you used in the actionscript in Keyframe 1 of this scene. I used the name tank in the script, so my instance name was tank. The instance name cell is shown here.

**61.** In your game scene, single click on your timer movieclip and give the movieclip the instance name timer in the Properties panel.



Your game should now work. Move on to the next page to learn how to publish your game.



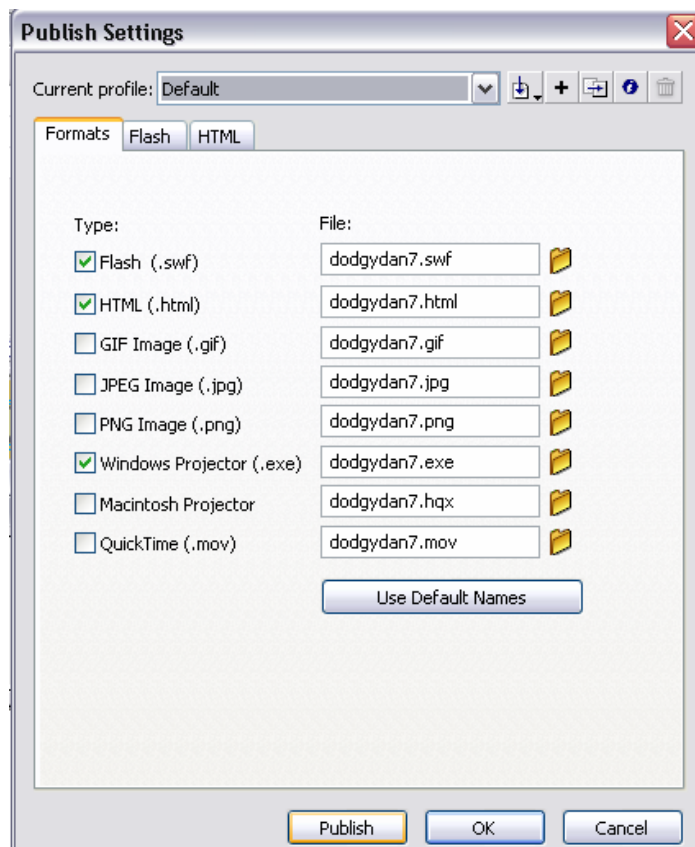
## TEST YOUR GAME

60. Your game should now be complete. To test your game, select **Control > Test Movie**. You will need to play it at least twice to ensure that it works correctly. If your timer is still visible, you may wish to drag this off the stage when you are ready to publish the final version of the game.

If everything works - congratulations! If not, trace back through your steps to try to work out where you went wrong. If you're still stuck, send your file to me at [kkope1@eq.edu.au](mailto:kkope1@eq.edu.au) and I'll give you a hand.

## PUBLISH & SHARE YOUR GAME

61. 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 shown below will appear.



62. 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 (.hpx) format options.