

# Galactic Mail



Learning Computer Programming by  
Creating Computer Games

# Learning Programming With Game Maker

You will use:

- **events** and **actions** to **change sprites and objects**.
- **Depth property** of **objects** to **control** the **order** that **instances appear** on screen.
- This chapter also introduced the use of *variables*, *even though we didn't call them variables*.
- For example, the **word direction** is a **variable** indicating the **current direction** of an **instance**.
- the **variables x and y** that **indicate** the **position** of an **instance**.
- There are many **variables** in Game Maker, and they are very useful. Other **variables** will be used in the chapters to follow.

# Learning Programming With Game Maker

## Galactic Mail Game Design

- *You play an intergalactic mail carrier who must deliver mail to a number of inhabited moons.*
- *The pilot must safely steer a course from moon to moon while avoiding dangerous asteroids.*
- *The mail carrier is paid for each delivery , but pay is deducted for time spent on moons.*
- *This adds pressure to the difficult task of piloting the rickety, old rocket, which doesn't steer very well in space.*
- *When the rocket is on a moon, the arrow keys rotate it to set the launch direction*
- *The spacebar launches the rocket, and the vacated moon is removed from the screen to show that its mail has been delivered.*
- *In flight, the rocket will keep moving in the direction it is pointing, with only a limited control over steering using the arrow keys.*

# Learning Programming With Game Maker

- *When objects move outside the playing area, they reappear on the other side to give the impression of a continuous world.*
- *A player gains points for delivering mail, but points are deducted while waiting on a moon.*
- *This encourages players to move quickly from moon to moon.*
- *There are different levels, with more asteroids to avoid.*
- *The game is over if the rocket hits an asteroid, and a high-score table will be displayed.*
- *The final game looks like the earlier slide*

# Learning Programming With Game Maker

- The various elements needed to create the game are
  - **moons, asteroids, explosions and rockets.**
- We'll actually use two different moon objects (for a normal moon and an occupied moon) and two different rocket objects (for a “landed rocket” and a “flying rocket”).
- All the resources for this game are in the Go to the shared folder to copy the material.

# Learning Programming With Game Maker

## Sprites and Sounds

- Sprites provide images for each element of the game.
- In this chapter, we'll use some extra abilities of sprites
- First set Game Maker into **Advanced Mode**

## Setting Game Maker into Advanced Mode:

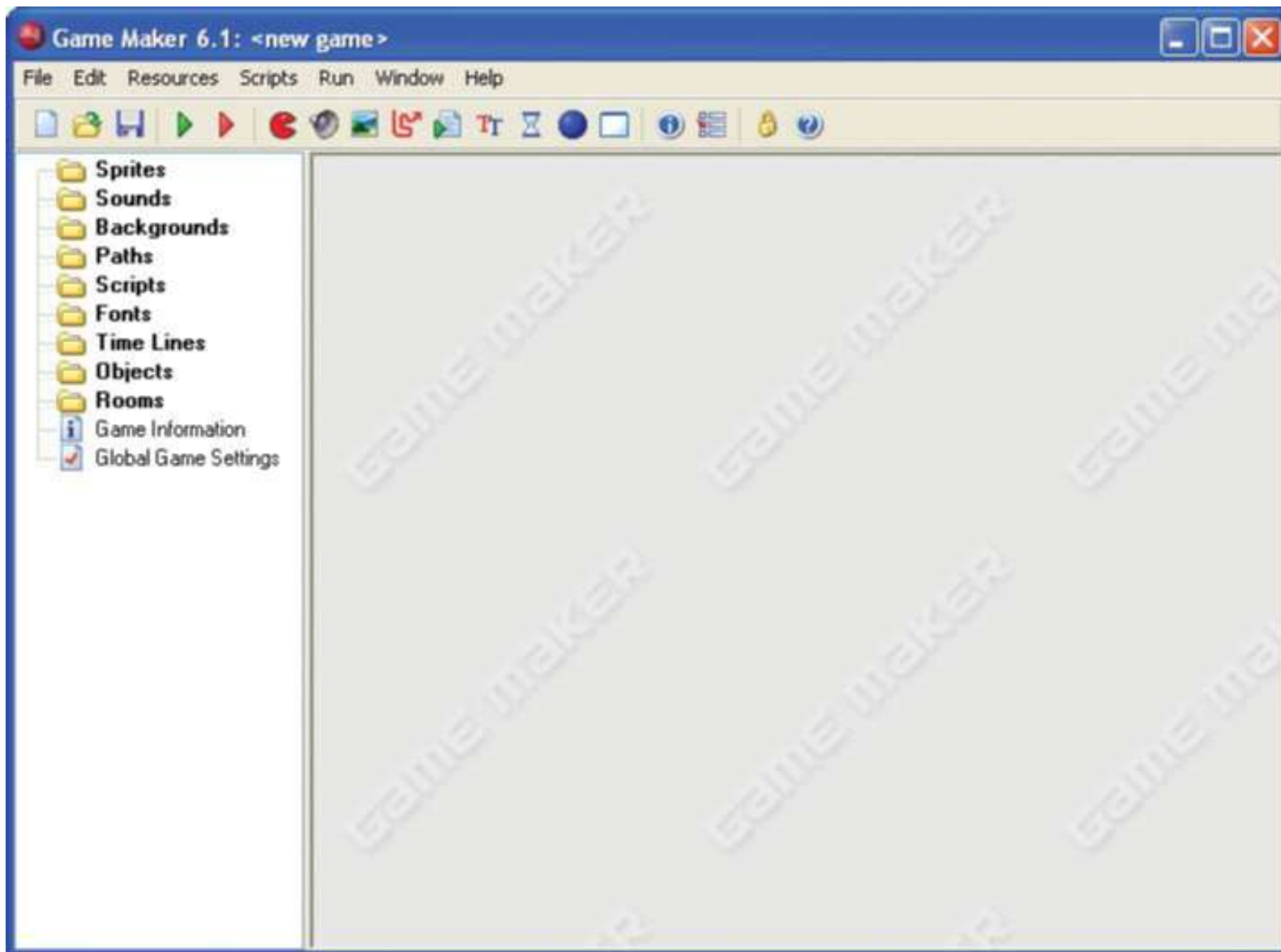
1. You must save any game you are working on before switching modes.
2. Click the **File** menu and look for **Advanced Mode**.

If there is a checkmark in front of it, then you are in Advanced mode.

Otherwise, click **Advanced Mode** to select it, and the main window should now look like Figure 3-2.

- We'll leave Game Maker in Advanced mode for the remainder of the book, even though some options are only used in later chapters.
- Now we're going to start a new, empty game.

# Learning Programming With Game Maker



**Figure 3-2.** *In Game Maker Advanced mode, there are additional resources on the left and a new menu.*

# Learning Programming With Game Maker

- First we'll create the sprites needed
- This time we'll add a couple of additional steps.
- Each sprite in Game Maker has its own *origin*, which controls the exact position that it appears on the screen.
- By default, the origin of a sprite is set at the top-left corner of the image.
- This means that when you move objects around in the game, it's as if you were holding them by their top-left corner.
- But, because the rockets in Galactic Mail need to sit in the center of the moons, it will be better if the origin of all sprites are centered.



# Learning Programming With Game Maker

Creating new sprite resources for the game:

1. From the **Resources** menu, choose **Create Sprite**.

The **Sprite Properties** form with additional Advanced mode options will appear, like the one shown in Figure 3-3.

2. Click in the **Name** field and name the sprite **sprite\_moon**.

3. Click the **Load Sprite** button.

Select **Moon.gif** from the CD's Resources/Chapter03 folder

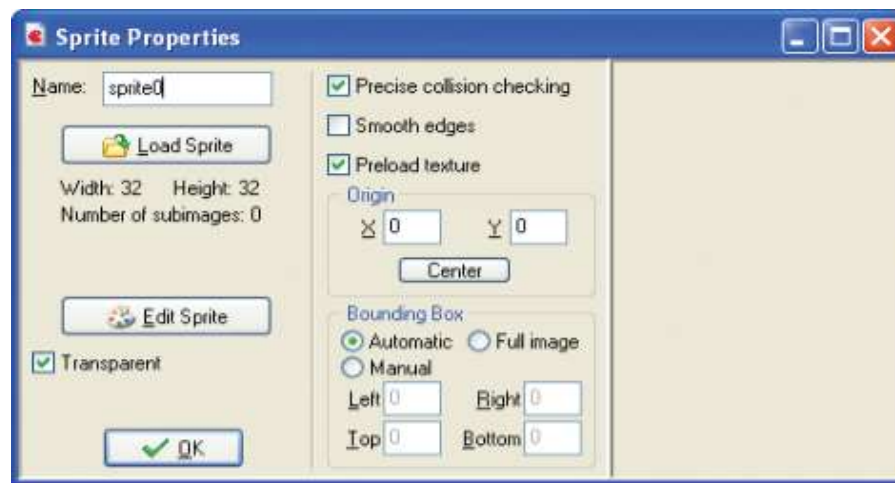


Figure 3-3.  
*The Sprite Properties form shows the advanced*

# Learning Programming With Game Maker

4. Controls to set the origin are in the middle of the form's second column

Click the **Center** button to move the origin to the sprite's center

Crosshairs will be in the middle of the sprite's image indicating the origin's position.

You can also change the origin by clicking on the sprite image with the mouse or by typing in X and Y values directly.

5. Enable the **Smooth edges** option by clicking on the box next to it.

This makes sprite edges look less jagged during the game by making them slightly transparent. (reduces "jaggies")

6. Click **OK** to close the form.

# Learning Programming With Game Maker

7. Now create **asteroid** and **explosion** sprites in the same way using **Asteroid.gif** and **Explosion.gif** (remember to center their origins).

8. We'll need **two sprites** for the **rocket**: one for when it has landed on a moon and one for when it is flying through space.

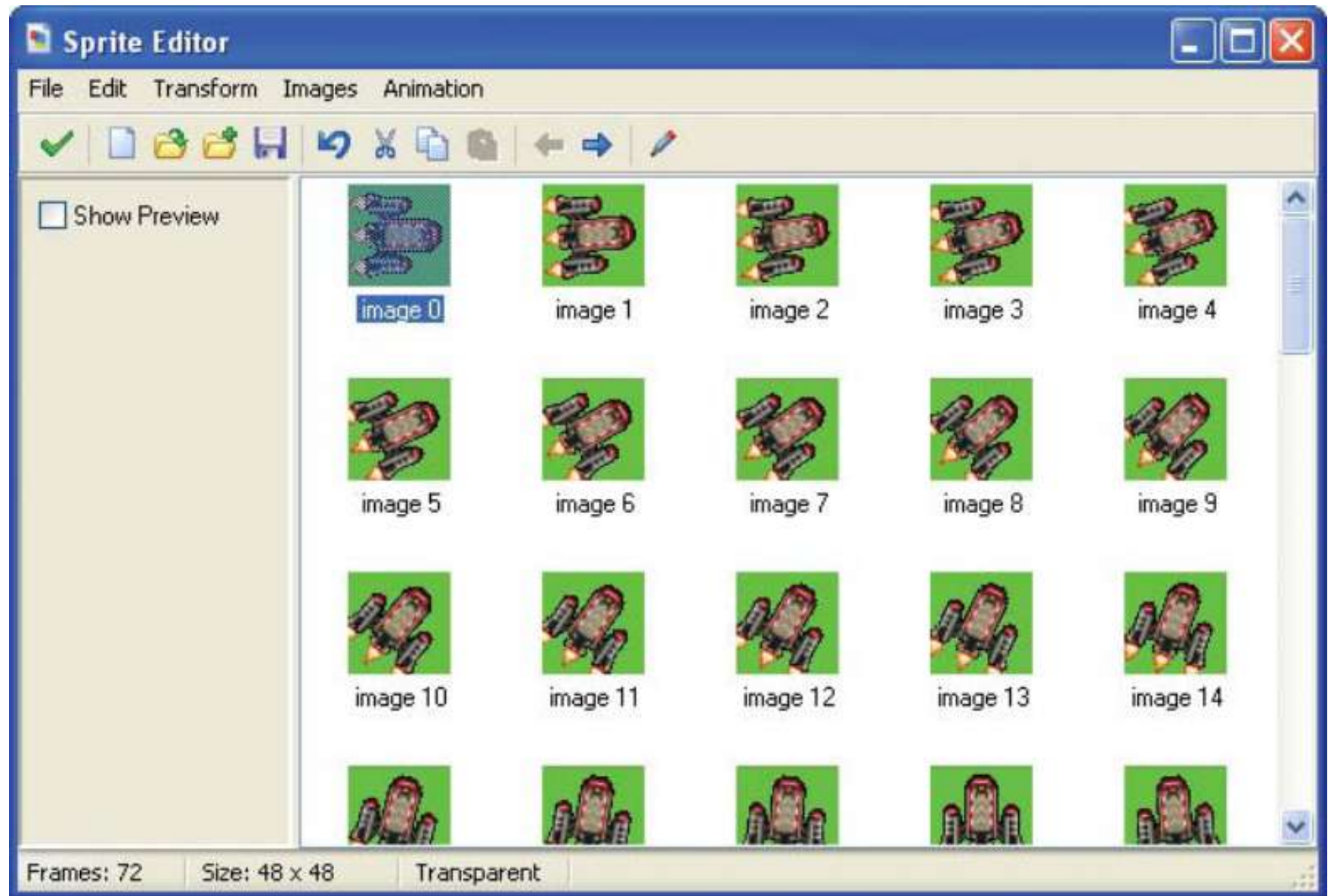
Create one sprite called **sprite\_landed** using **Landed.gif** and another called **sprite\_flying** using **Flying.gif**.

Center the origins of both sprites

- Before closing the **Sprite Properties** form for this last sprite, click the **Edit Sprite button**.

A form will appear like the one shown in Figure 3-4.

# Learning Programming With Game Maker



**Figure 3-4.** *The Sprite Editor shows all the images of the rocket.*

# Learning Programming With Game Maker

- By scrolling down the images contained in this sprite, you'll see that it is an animation of the rocket turning a full circle.
- There are **72 different images** at slightly different orientations, animating a complete turn of 360 degrees.
- We use the images to pick the correct rotation position for the rocket in the game.
- We can use the **Sprite Editor** to change the sprite in many ways, but for now simply close it by clicking the button with the **green checkmark** in the top left of the window.
- The game now has five different sprites.

**moon, asteroid, explosion and 2 rocket ships**

# Learning Programming With Game Maker

**Creating new sound resources for the game:**

1. Select **Create Sound** from the **Resources** menu.

Note that the **Sound Properties** form now has additional **Advanced mode** options, but we don't need them now (some are only in the registered version of Game Maker).

2. Call the sound **sound\_explosion** and click **Load Sound**.

Select the Explosion.wav file from Resources/Chapter03 on the CD.

3. Close the form by clicking OK.

4. Now create the sound\_bonus and music\_background sounds in the same way using the Bonus.wav and Music.mp3 files.

Adding all these resources at the start will make it easier to drop them into the game as we are going along—so let's get started on some action.

# Learning Programming With Game Maker

## Moons and Asteroid Objects

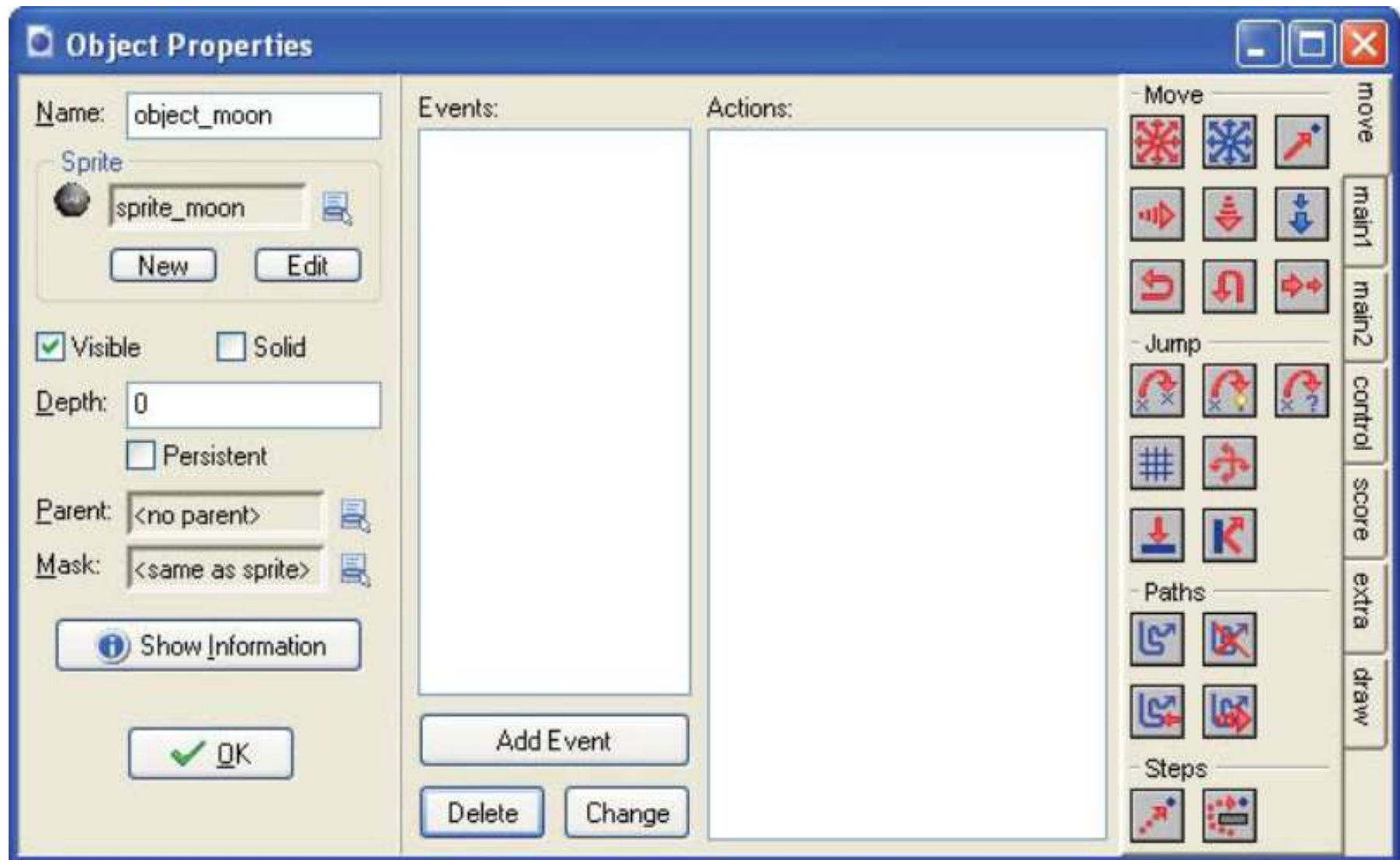
- Both **moons** and **asteroids** will fly around the screen in straight lines, jumping to the opposite side of the room when they go off the edge of the screen.

In Game Maker this is called *wrapping*, and it is done using the **Wrap Screen action**.

### Creating the moon object:

1. From the **Resources** menu, choose **Create Object**. The Advanced mode Object Properties form has additional options and actions too (see Figure 3-5).
2. Call the object **object\_moon** and give it the **moon** sprite

# Learning Programming With Game Maker



**Figure 3-5. The Object Properties form for the moon object looks like this.**



# Learning Programming With Game Maker

- When a moon is created, we want it to start moving in a completely random direction.

## Adding a create event to the moon object:

1. Click the **Add Event** button and choose the **Create** event.

2. Include the **Move Free** action in the **Actions** list for the event.



3. This action form requires a direction and a speed.

Enter a **Speed of 4** and type **random(360)** in the **Direction** property.

This indicates a random direction between 0 and 360 degrees.

The form should now look like Figure 3-6.

# Learning Programming With Game Maker

- To make sure that the **moon reappears** at the other side goes off the edge of the room, include a **wrap action** for the moon object:

1. Click the **Add Event** button, choose the **Other** events, and select **Outside Room** from the pop-up menu.

2. Include the **Wrap Screen** action in the **Actions** list.



3. In the form that appears, indicate that wrapping occurs in **both directions** (**top to bottom** and **left to right**).

Now the form should look like Figure 3-7.

4. The **moon** object is now ready to go, so you can close the **Object Properties** form by clicking **OK**.



**Figure 3-6.**  
*Use the **random** command in a **Move Free** action.*



**Figure 3-7.** *The **Wrap Screen** action properties form*

# Learning Programming With Game Maker

- The **asteroid** object can be created in exactly the same way as the **moon**.
- However, we want to make the asteroids appear **behind** other objects when they cross paths with them on the screen.
- Object instances are usually drawn in the order their creation, so you can't be sure if one type of object will appear in front of another.
- But, you can **change** this with an object's ***depth value***.
- ***Instances*** with a smaller depth value are drawn on top of instances with a greater depth, and appear in front of them.
- **All objects** have a **default depth of 0**, so to make sure the **asteroids** appear behind other objects, simply give them a depth greater than 0.

# Learning Programming With Game Maker

## Creating the asteroid object:

1. Create a new object called **object\_asteroid** and give it the **asteroid** sprite.



2. On the left-hand side there is a text field labeled **Depth**.

Enter **10** in this field and change the **depth** of the **object** from **0** to **10**.

3. Add the **Create** event and include the **Move Free** action in the **Actions** list.

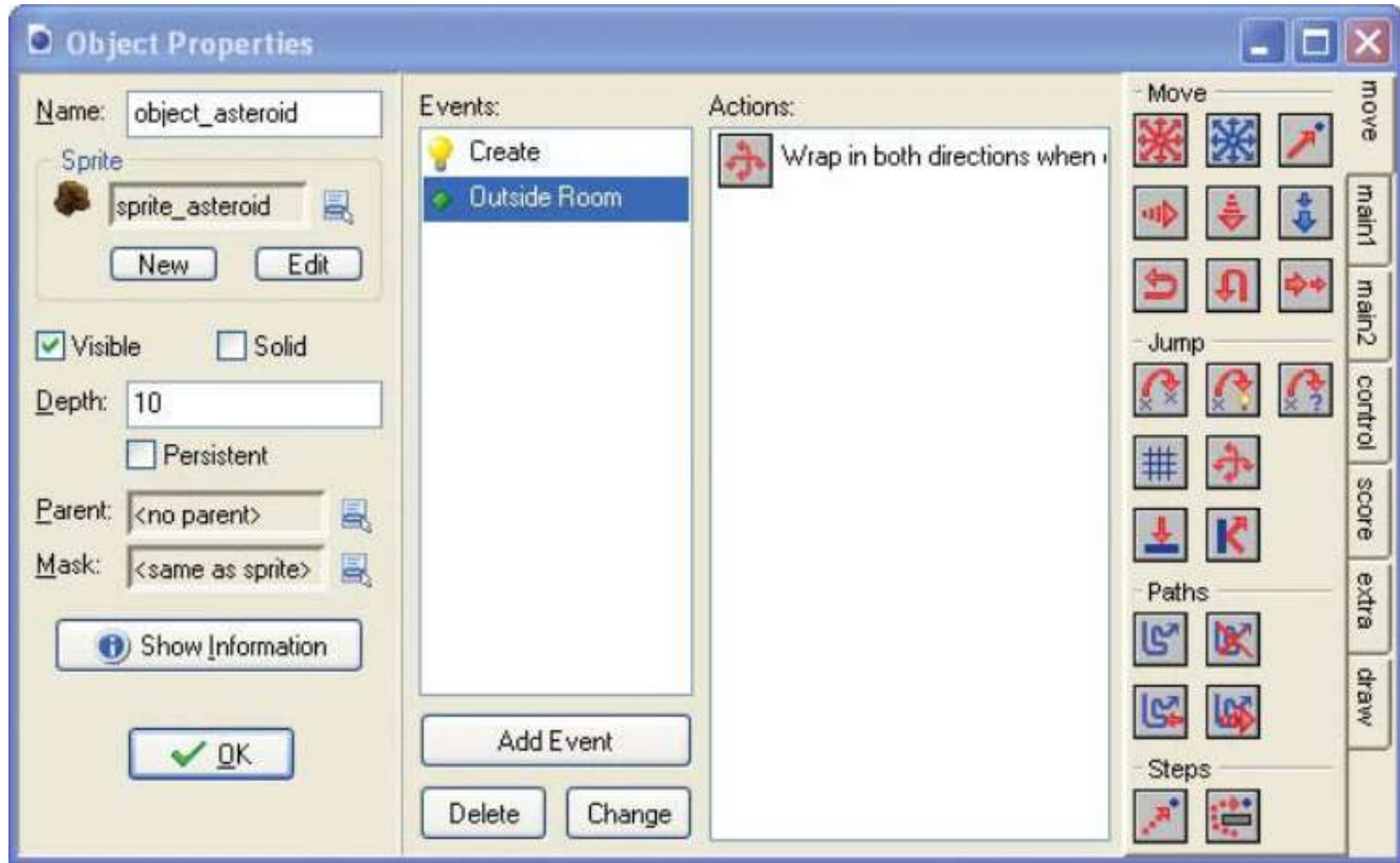


Type **random(360)** in the **Direction** property and enter a **Speed** of **4**.

4. Add the **Other, Outside Room** event and include the **Wrap Screen** action in the **Actions** list (indicate **wrapping in both directions**).

# Learning Programming With Game Maker

5. The **Object Properties** form now looks like Figure 3-8. Click **OK** to close the form.



**Figure 3-8. Set the depth for the asteroid object.**

# Learning Programming With Game Maker

- Now would be a **good time to check** that everything has gone according to plan.
- Before we can do that, we must **create a room** with some **instances of moons and asteroids**.

## Creating a room with moon and asteroid instances:

1. Select **Create Background** from the **Resources** menu.
2. Call the background **background\_main**, and click the **Load Background** button.

**Select** the **Background.bmp** image from the folder **Resources/Chapter03**

3. Click **OK** to close the **Background Properties** form.
4. Select **Create Room** from the **Resources** menu.

If the whole room isn't visible, enlarge the window.

# Learning Programming With Game Maker

5. Select the **settings** tab and name the room **room\_first**.

Provide an appropriate **caption** for the room (“**Galactic Mail**”).

6. Select the **backgrounds** tab.

Click the **menu** icon to the right of where it says **<no background>** and select the **background** from the pop-up menu.

7. Select the **objects** tab and **place a number of asteroids and moons in the room**.

(Remember that you can choose the object to place by clicking where it says “**Object to add with left mouse**”).

The **Room Properties** form should now look like **Figure 3-9**.

8. Close the **Room Properties** form by clicking the **green checkmark** in the top-left corner.



# Learning Programming With Game Maker

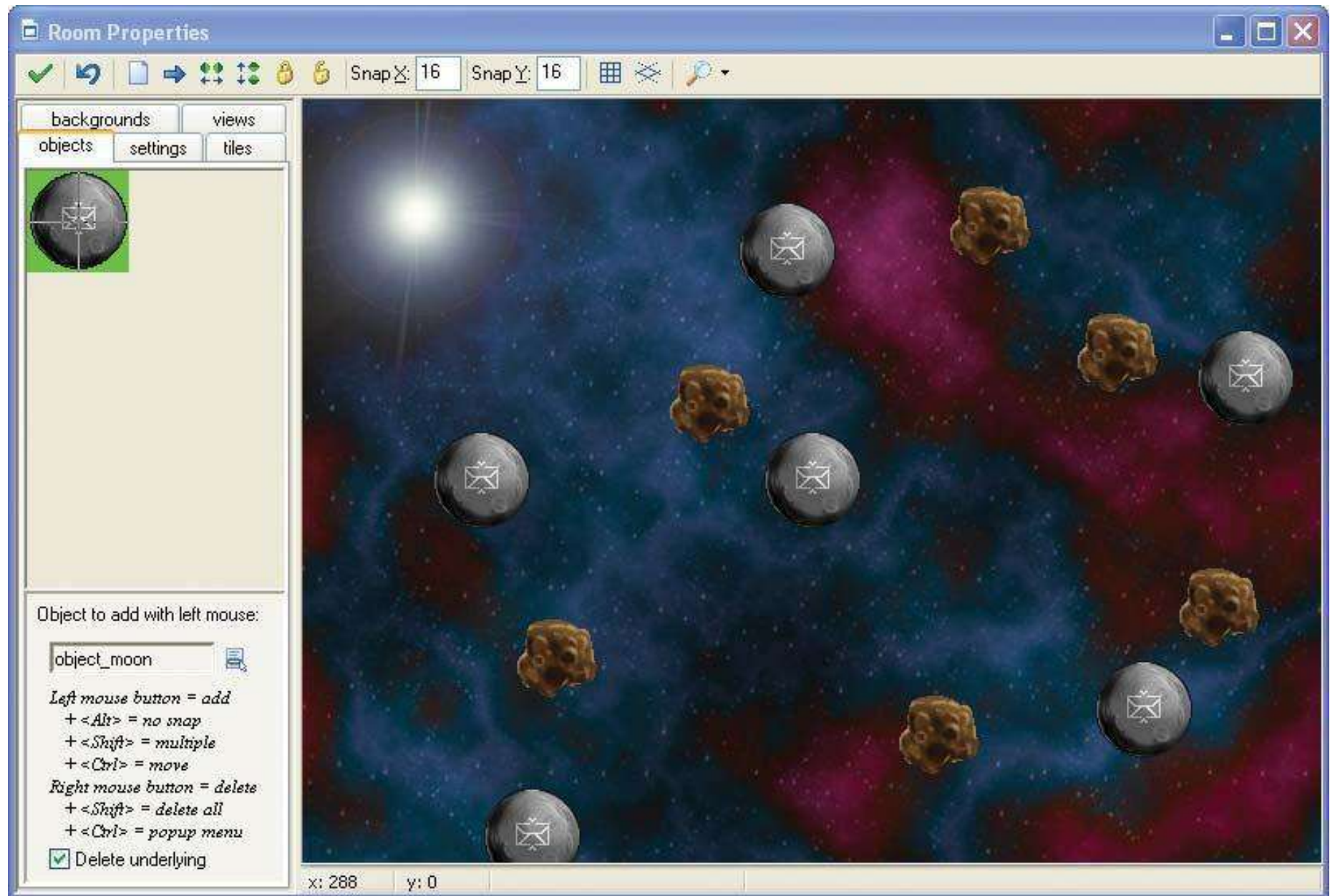


Figure 3-9. Here's our first room



# Learning Programming With Game Maker

## Saving and running the game:

1. Choose **Save** from the **File** menu (or click the disk icon).

Save the game where you can easily find it again (the desktop, thumbdrive, or your personal GWC storage area, for example).

2. Select **Run normally** from the **Run** menu. If all goes well, the game should then appear in a new window.

- Before continuing, **double-check** that everything is working the way it's **supposed to**.

a. Are the **moons** and **asteroids** moving in **different random directions**?

b. Do they **reappear** on the other side of the screen when they leave the room?

c. Do the **asteroids** always pass **behind** the **moons**?

If any of these are not working properly, check that you've followed the instructions correctly.

# Learning Programming With Game Maker

## Flying Around

- So far, so good, so we'll introduce some **gameplay** by bringing the **rocket** into the game.
- We'll make two rocket objects, but let's stop to consider why
- Our rocket has two different ways of behaving:

**Sitting** on top of a moving moon with **full** control over the ship's direction, and

**Flying** through space with only **limited** control

- If we used **two** ways of controlling one object it would require a complicated set of events and actions, but if we separate these behaviors into two different objects, then it becomes easier.
- As long as both objects look the same, the player won't notice that the ship changed from being a "**flying rocket**"

# Learning Programming With Game Maker

- We also need 2 **moon** objects, since we want the **landed** rocket object to follow the path of one particular moon around (the one it's on).
- Making it into a **separate object** will allow us to **single it out** from the others.
- Since the **2<sup>nd</sup> moon** object will be almost the same as the **normal** moon, we can take a shortcut and make a copy of the existing moon object.

# Learning Programming With Game Maker

## Creating the special moon object:

1. **Right-click** the **moon** object in the **resource** list, and select **Duplicate** from the pop-up menu.

A copy of the moon object will be added to the resource list and its properties form is displayed.


2. Change the **name** to **object\_specialmoon**.

**It's important to use this exact name** (including the underscore) since we'll use it to identify this object later.

3. **Set** the **Depth** of this object to **-5**.

This guarantees that instances of this moon are always in front of the other moons since it is less than 0.

4. We'll also make this **moon start** the **background music** at the **beginning** of the game.

Add an **Other, Game start** event and include a **Play Sound** action  (main1 tab).

Select the **background music** sound and set **Loop to true** so that the music plays continuously.

5. Click **OK** to close the **properties** form.

# Learning Programming With Game Maker

- Now open the **first room** and add a **single instance of this new special moon** to the level.
- **Run the game** and the **music** should play.
- (You won't notice any other difference because the special moon should look and behave exactly like the other moons.)

Now we can make our **2 rocket objects**.

- We'll begin with the **landed rocket**, which needs to sit on the **special moon object** until the player decides to **blast off**.
- We'll use a **Jump Position** action to make it follow the **special moon's** position as it moves around the screen.

# Learning Programming With Game Maker

## Creating the landed rocket object:

1. Create a **new object** called **object\_landed** and give it the **landed rocket** sprite.

Set the **Depth** to **-10** so that it appears in front of the moons and will look like it's sitting on the special moon's surface

2. Add a **Step, End Step** event to the new **object**.

An **End Step** allows actions to be performed immediately before instances are drawn at their new position on the screen.

**Therefore, we can use this event to find out where the special moon has been moved to and place the rocket at the same location – just before both of them are drawn.**

■**Note** – A **Step** is a short period of time in which everything on the screen moves a very small distance.

Game Maker normally takes **30 steps every second**, but you can change this by altering the **Speed** in the **settings tab** for each **room**.

# Learning Programming With Game Maker

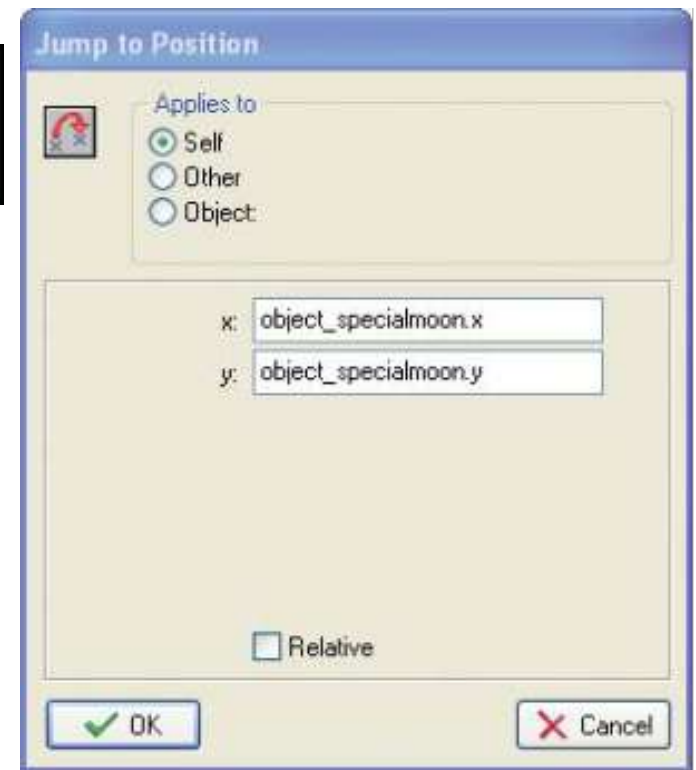
3. Include the **Jump Position** action in the **Actions** list for this event.



This action allows us to move an object to the coordinates of any position on the screen.

Type **object\_specialmoon.x** into the **X value** and **object\_specialmoon.y** into the **Y value**.

- These indicate the **x and y** positions of the special moon.
- Make sure that you type the names carefully, including underscores and dots in the correct positions.
- The action should now look like **Figure 3-10**.



**Figure 3-10.** We set the rocket to jump to the **x and y positions** of the **special moon**, so that it will follow this moon around.

# Learning Programming With Game Maker

## 4. You might want to test the game now.

Place **one instance of the rocket** at a random position in the room and **run the game**.

The rocket **should jump to the position** of the **special moon** and stay on top of it as it moves around.

- When you run the game, you will also notice that the **rocket continually spins around without any user input**.
- This is because the **rocket sprite contains an animation showing the rocket rotating through 360 degrees**.
- By default, Game Maker automatically cycles through a sprite's subimages to create an animation.
- But, we don't want this for this game – we need Game Maker to select the appropriate **subimage** based on the **direction the rocket will be moving in**.



# Learning Programming With Game Maker

- This requires a small amount of arithmetic.
- There are 72 images representing a turn of 360 degrees, so each image shows the rocket rotated by 5 degrees more than the previous (because  $360/72 = 5$ ).
- Game Maker stores the direction of all objects in degrees, so it can determine which rocket subimage to use by dividing the **rocket object's** current **direction** by **5**.
- Then we can make the rocket face in the correct direction by using this rule (**direction/5**) to set the current subimage in a **Change Sprite** action

# Learning Programming With Game Maker

## Adding a change sprite action in the landed object:

1. With the **landed rocket Object Properties** form open, add a **Change Sprite** action (**main1** tab) in the **End Step** event.



- Choose the **landed rocket** sprite from the menu and type **direction/5** into the **Subimage property**.

- **direction** is a special term that Game Maker uses as the direction that this instance is currently facing

- Finally, set **Speed** to **0** to stop the sprite from animating on its own and changing the subimage.
- Figure 3-11 shows how this action should now look.

# Learning Programming With Game Maker



**Figure 3-11. Set the correct subimage in the sprite**

■ **Note** This way of dealing with rotated images might seem rather clumsy, but many old arcade games were made in a similar way so that each rotated image could include realistic lighting effects.

Nonetheless, the registered version of Game Maker contains an additional action to rotate a sprite automatically without the need for sub-images.

# Learning Programming With Game Maker

- We will also make use of the direction term for the object to add actions that allow the player to control the direction of the rocket using the arrow keys.

**Adding a keyboard event to the landed rocket object:**

1. Add a **Keyboard, <Left>** event to the **landed rocket** object.
2. Add the **Move Free** action and type **direction+10** in the **Direction** property.

This indicates that the current direction should be increased by **10 degrees**. Set **Speed to 0** because we don't want the rocket to move independently of the special moon.

- This action should now look like Figure 3-12.

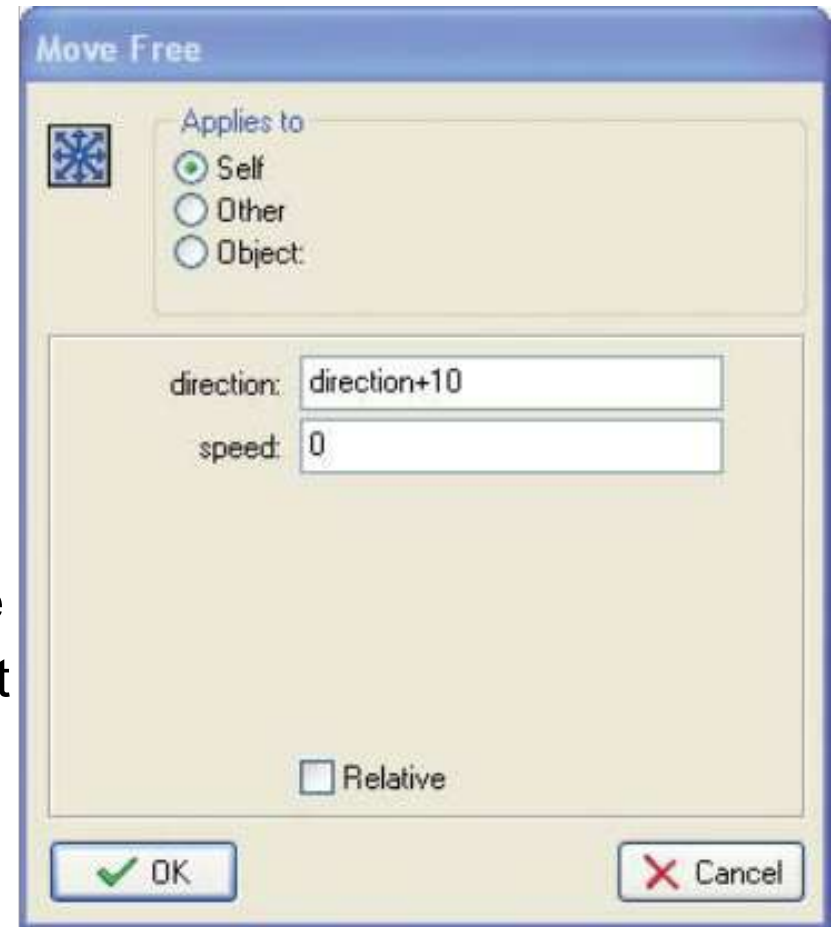


# Learning Programming With Game Maker

3. Add a similar Keyboard event for the <Right> key.

Add a **Move Free** action and type **direction-10** in the **Direction** property.

- **The last control** that we will need for the landed rocket will allow the player to **launch the rocket** using the **spacebar**.
- This control will **change** the landed rocket object into a **flying rocket** object, but since we haven't created the **flying rocket** object yet, we can't make an action yet either!



**Figure 3-12. Set the direction to equal itself plus 10**

# Learning Programming With Game Maker

- So let's make the **flying rocket** now and finish up later

## Creating the flying rocket object:

1. Create a new object called **object\_flying** and select the **flying rocket** sprite.

Set **Depth** to **-10** to make sure that this object appears in front of moons.



- 2. Add an Other, Outside Room** event and include a **Wrap Screen** action to wrap around the screen in **both directions**.

- 1. Add an End Step** event.



Add a **Change Sprite** action, choose the **flying rocket** sprite, type **direction/5** in the **Subimage property**, and set the **Speed** to **0**.

# Learning Programming With Game Maker



4. Add a **Keyboard, <Left>** event and include a **Move Free** action.

We don't want the player to have too much control over the flying rocket, so type **direction+2** in **Direction** and set **Speed** to **6**.

5. Add a **Keyboard, <Right>** event with a **Move Free** action. Type **direction-2** in **Direction** and set **Speed** to **6**.

- The **basic gameplay** is nearly complete now – just a few more events to add
- **First**, the **game should end** when the **rocket hits an asteroid**.
- **Next**, when the **flying rocket reaches a moon**, it must change to a **landed rocket**, and the **moon** must change to a **special moon** (so that the landed rocket can follow ).



# Learning Programming With Game Maker

- We change the objects by using the **Change Instance action**, which replaces an instance of one type of object with another.
- Its as if we melted the jello from one instance and poured it into a new object mold.
- Although the instance becomes a completely different kind of object, it keeps some of its original properties, such as its position on the screen and its direction.
- It is critical that these values remain the same – else the launch direction of the landed rocket be reset as soon as it turned into a flying rocket!





# Learning Programming With Game Maker

- **Adding collision events to the flying rocket object:**

**1. Add a Collision** event with the **asteroid** object and include the **Restart Game** action (**main2** tab) in the **Actions** list.



Later, we'll add an explosion to make this better.

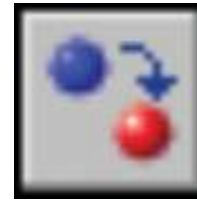
**1. Add a Collision** event with the **moon** object and add the **Change Instance** action (**main1** tab).

Set the **object** to change into **object\_landed** using the **menu button**, and leave the other options unchanged.



# Learning Programming With Game Maker

3. Add a second **Change Instance** action for changing the **moon** into a **special moon** object.



- So that this action changes the moon object (and not the rocket), click the **Applies to** option to **Other** from **Self**.
- Now the action will apply to the other object in the collision, which is the **moon**.
- Set the object to change into **object\_specialmoon**.
- Figure 3-13 shows the settings.



Figure 3-13. Change the **other** instance involved in the collision to a **special moon**.

# Learning Programming With Game Maker

- Now, we can **go back** to the **landed rocket** object.
- We need an **event** that changes it into a **flying rocket** and **deletes** the **special moon** when the **spacebar** is **pressed**.

## Adding a key press event to the landed rocket object:

1. Reopen the **Object Properties** form for the **landed rocket** by double-clicking on it in the resource list.

2. **Add a Key Press, <Space>** event and include a **Move Fre** action to set the rocket in motion.



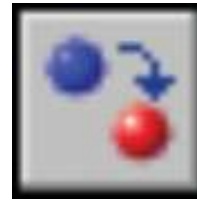
Type **direction** in the **Direction** property (this keeps the direction the same) and set **Speed** to **6**.

3. Now add a **Change Instance** action and **change** the object into an **object\_flying**.



# Learning Programming With Game Maker

4. Finally, we want to **delete** the **special moon** because it no longer needs to be visited.



- Include a **Destroy** action and change the **Applies to** option to **Object**.

• Click the **menu** button next to this and select the **object\_specialmoon**

See Figure 3-14.

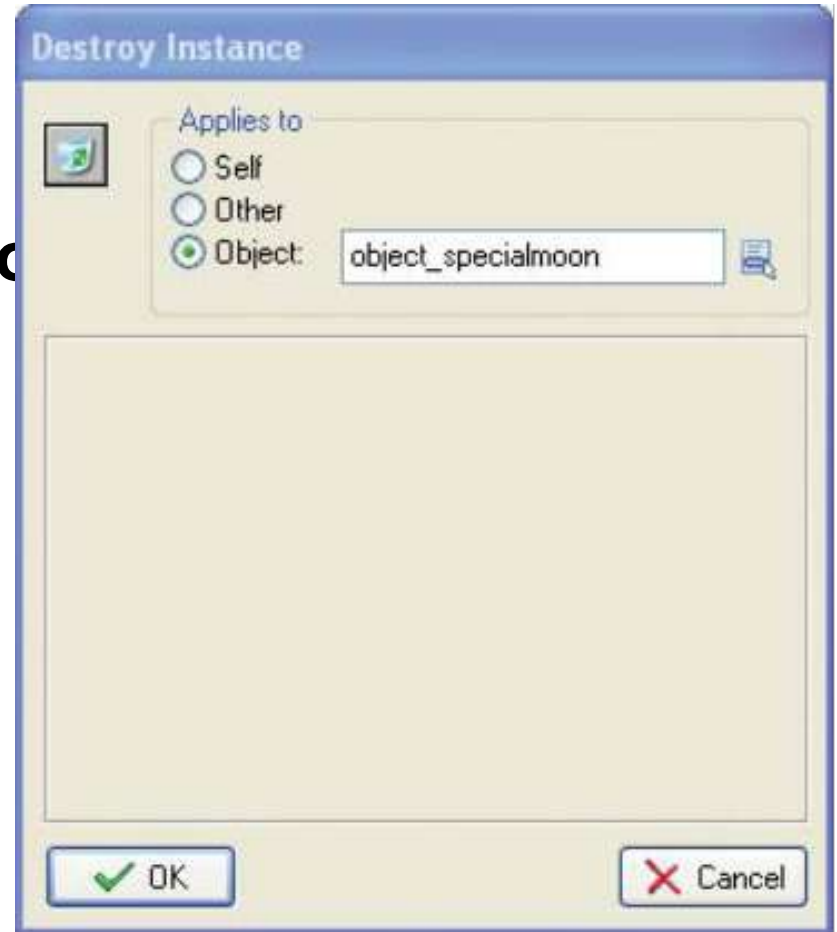
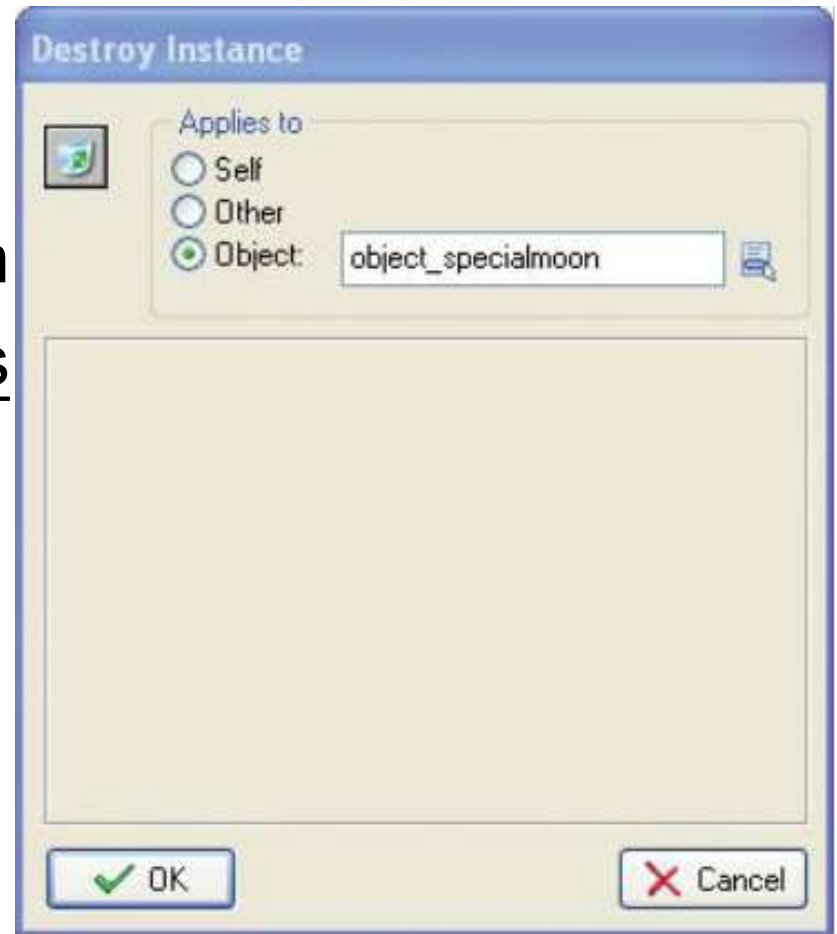


Figure 3-14. Include a Destroy action for the **special moon**..

# Learning Programming With Game Maker

■ **Caution - Using the Object setting for Applies to performs an action on ALL instances of that type of object in the room.**

- Deleting all of the special moon instances is OK in this case (as there is only one), but you will need to think carefully about the effects **this setting will have before using it in your own games.**



**Figure 3-14.** *Include a Destroy action for the **special moon**..*

# Learning Programming With Game Maker

- That completes the **second version** of the game.
- Make sure you save it and check that it all works as it should so far. Don't keep going unless this part works as planned
- You should be able to rotate the rocket on a moon, launch it with the spacebar, and steer through the asteroids to land on another moon.
- Moons should disappear as you visit them, and the game should restart if you hit an asteroid.
- If something isn't working, then check the instructions again.
- Or compare your version with the version on the CD (Games/Chapter03/galactic2.gm6).

# Learning Programming With Game Maker

## • SCORING, LEVELS, AND FINISHING TOUCHES

### Winning and Losing

- In this section we'll put a bit more effort into what happens when the player wins or loses the game.
- We'll start by making asteroids explode on contact

### An Explosion

- To make this work, we add a new **explosion object** and **create an instance** of it when the **rocket hits an asteroid**.
- This will play the **explosion sound** when it is created and **end the game** with a **high-score table** after the explosion animation has finished.

# Learning Programming With Game Maker

Adding an explosion object to the game:

**1. Create** a new **object** named **object\_explosion**, and select the **explosion sprite**.

Give it a **Depth** of **-10** to make it appear in front of other instances.

**2. Add** a **Create** event and add a **Play Sound** event (main1 tab) for the **explosion** sound.



**1. Add** an **Other, Animation End** event.

This **event occurs** when a sprite reaches the **final subimage** in its animation.



# Learning Programming With Game Maker

4. Add the **Show Highscore** action (**score** tab) in the **Actions list** for this event.

To make the high-score list look more interesting, set **Background** to the **same** as the **background** for the game, set **Other Color** to **yellow**, and choose a **different font** (e.g., **Arial, bold**).

The Show Highscore action window will now look like



Figure 3-15.

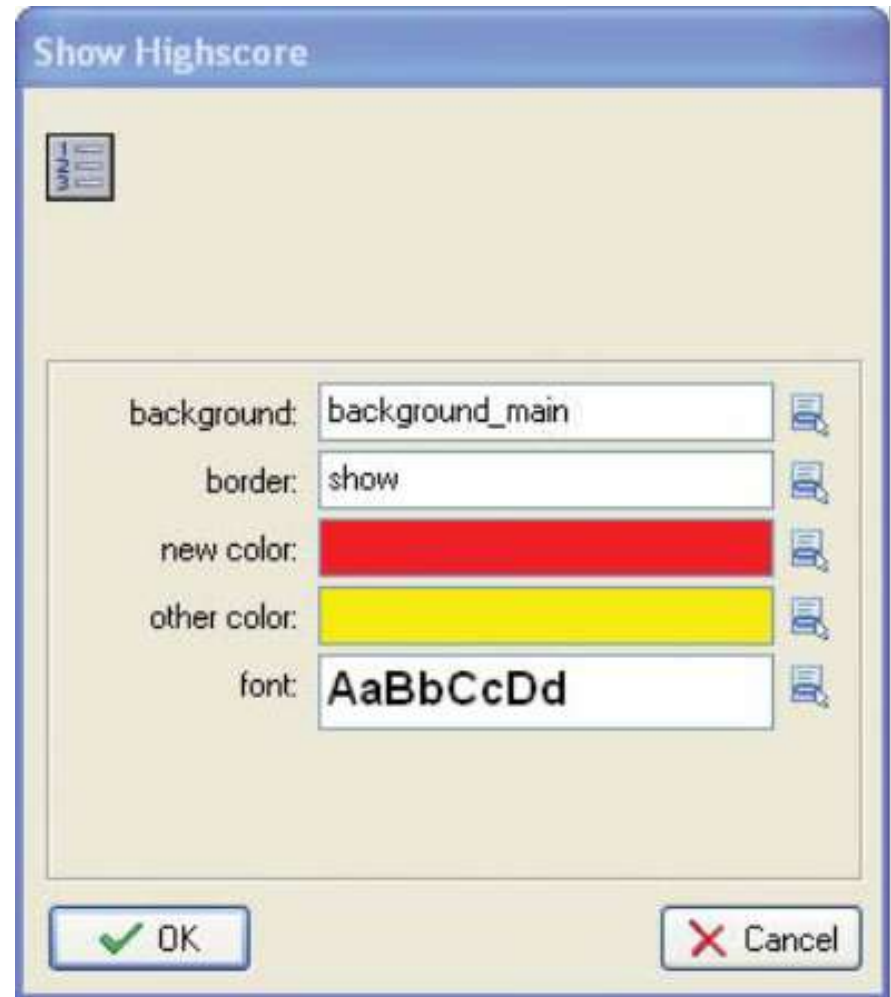


Figure 3-15. *You can spice up the high-score table*

# Learning Programming With Game Maker

5. Also add a **Restart Game** action to start the game again after the high-score table is closed (**main2** tab).

6. Click **OK** to close the object.



- Next we **change the behavior of the flying rocket** when it **hits an asteroid**.

# Learning Programming With Game Maker

## Editing the flying rocket object:

1. Reopen the **properties** form for the **flying rocket** object by double-clicking on it in the **resource** list.

2. Select the **Collision** event with the **asteroid** by clicking on it **once**.

Click **once** on the **Restart Game** action and press the **Delete key** to remove it from the action list.

3. Add a **Create Instance** action (**main1** tab) in its place, and set it to **create** the **explosion** object.

Make sure the **Relative** property is **enabled** so that the explosion is created at the current position of the rocket.

4. Add a **Destroy Instance** action (**main1** tab) and leave it set to **Self** so that the rocket gets deleted.

Click **OK** on the **properties** form to finish.



# Learning Programming With Game Maker

- You might want to run the game now to see how it looks.
- Try colliding with an asteroid and you should get an explosion followed by the high-score table.
- You can't score any points yet, so let's add this now.

## Scores

- When playing the game, you may have noticed a way of “cheating.”
- You can avoid the risk of hitting asteroids by waiting for another moon to fly right next to your own and then quickly hop between moons.
- The game is less challenging once you see this, so the scoring system that we'll use will discourage playing this way.

# Learning Programming With Game Maker

- Although players gain points by delivering mail, they will also lose points by waiting on moons.
- Then a player that takes risks by launching the rocket as soon as possible will have more excitement trying to avoid asteroids but can also score more points.

## Editing game objects to include scoring:

1. Reopen the **properties** form for the **special moon** object and select the **Game Start** event.

Include a **Set Score** action with a **New Score** of **1000**.

This will give players points at the start.

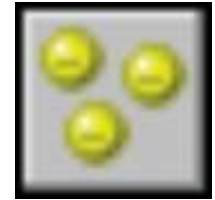
**Close** the **properties** form.



# Learning Programming With Game Maker

**2.** Reopen the **properties** form for the **landed rocket** and select the **End Step** event.

Add a **Set Score** action with **New Score** as **-1** and the **Relative** option **enabled**.



This reduces the score 1 point after every **Step** for as long as the player remains on a moon.

With 30 steps every second, a player will lose 30 points every second by staying on a moon.

Close the properties form.

**3.** Reopen the **properties** form for the **flying rocket** and **select** the **Collision** event with the **moon object**.

Include a **Set Score** action with a **New Score** of **500** and the **Relative** option **enabled**. (cont'd on next slide)



# Learning Programming With Game Maker

**4. Add a Play Sound** action after **setting** the **score** and **select** the **bonus** sound.

## Levels

- At this point, there is no reward for delivering all the mail.
- In fact, once all the moons are removed, the rocket just flies through space until it collides with an asteroid!
- This is a bit unfair, and so let's advance the player to a new level.

**Making multiple levels in Game Maker is easy.**

- We'll use actions to move between the levels (rooms), and add more asteroids in the levels to make play more difficult

# Learning Programming With Game Maker

- Let's start by creating the new levels.
- You'll repeat these steps to make one more level so that there are 3 levels in total.
- You can always add more of your own later on.

## ■ Note

**The order of the rooms in the resource list determines the order of your levels in the game, with the top level being first and the bottom level last.**

If you need to change the order, just **drag and drop** them into new positions into the list.



# Learning Programming With Game Maker

**Creating more level resources for the game:**

**1. Right-click** on a **room** in the **resource** list and **choose Duplicate** from the **pop-up** menu.

This will create a copy of the level.

**2. Go** to the **settings** tab and give the **room** an appropriate **name** (**room\_first**, **room\_second**, etc.).

**3. Switch** to the **objects** tab, and **add** or remove **instances** using the **left** and right **mouse** buttons.

**4. Make sure that each level contains exactly one special moon and one instance of the landed rocket. (VERY IMPORTANT!)**

# Learning Programming With Game Maker

- To tell Game Maker when to move on to the next room level, we must be able to determine if all the moons are destroyed in the current level (remember a moon instance is destroyed when the rocket is launched from it.

- To do this, we will use a ***conditional action*** that asks:

**“Is the total number of remaining moons (moon count) equal to zero?”**

- **If the answer is YES (or in computer terms, true), then a block of actions will be performed;**

**- otherwise (or ELSE) if the answer is NO (or false), then the block of actions is skipped.**

- We'll put this **check** in the **collision event** between the **flying rocket** and the **moon**, so that players will **finish** the **level** as soon as they **land on the final moon**.

# Learning Programming With Game Maker

## ■ Note

All conditional actions ask questions, and their icons are octagon-shaped with a blue background to easily recognize them.

Editing the flying rocket object to test for the number of remaining moons:

1. Reopen the **properties** form for the **flying rocket** and **select** the **Collision** event with the **moon** object.



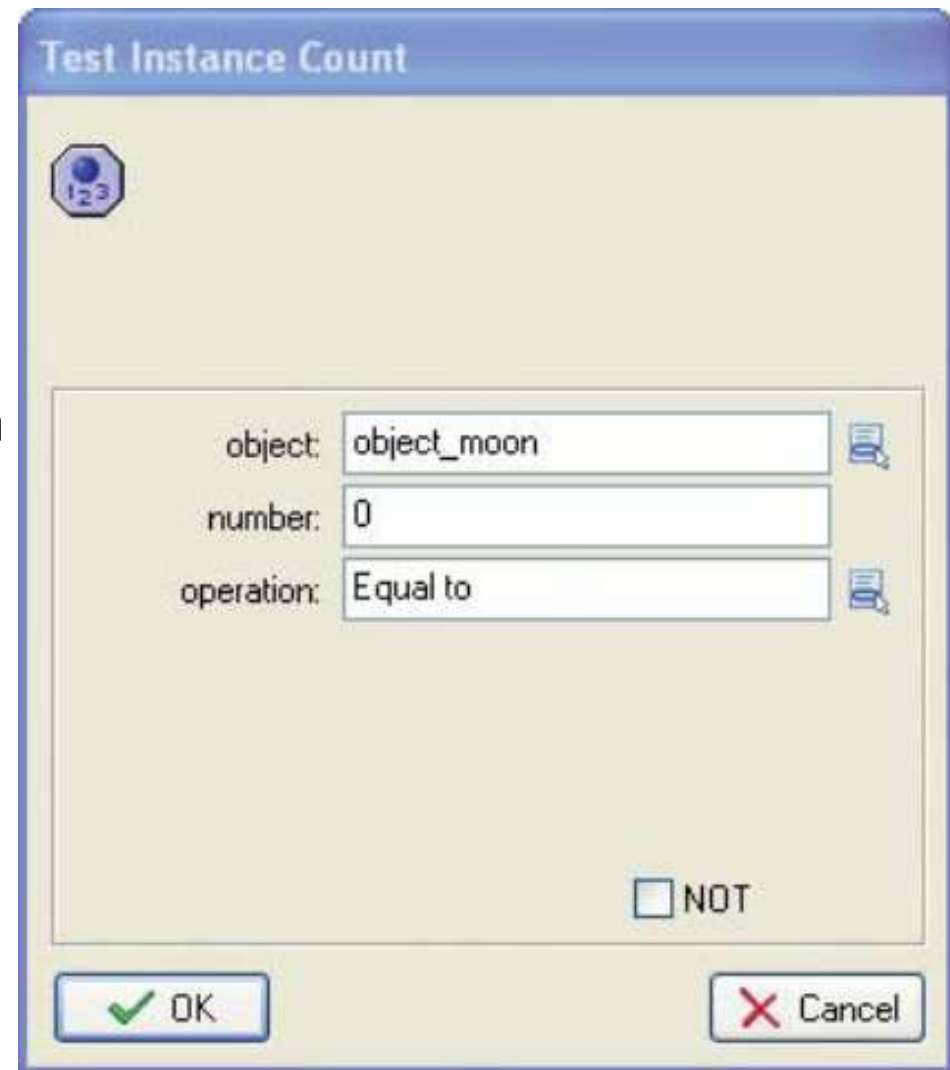
# Learning Programming With Game Maker

**2. At the end of the current list of actions, add a Test Instance Count action (control tab).**

**Set the Object field to `object_moon`; other**

**settings will default to how we need them (Number, 0 and Operation, Equal to).**

- This is now **equivalent** to the question “**Is the total number of remaining moons equal to zero?**”
- The form should look like this.



**Figure 3-16. Use the Test Instance Count action to count the moons.**

# Learning Programming With Game Maker

3. Below this action we need to start a *block*.

- A *block* indicates that a number of actions are grouped together as part of a conditional action.
- This means that all of the actions in the block will be performed **if the condition is true** and **none of them if it is not**.
- Add the **Start Block** action (control tab) directly below the **condition** to **test the instances**.

Start Block



# Learning Programming With Game Maker

**4. First**, we will pause for a moment to give the player a chance to notice they have reached the last moon at that level.



- Include the **Sleep** action (**main2** tab) and set **Milliseconds** to **1000**.

- There are 1,000 milliseconds in a second, so this will sleep for 1 second.

**5. We'll award the player a bonus of 1,000 points when they finish a level.**

- **Add a Set Score** action (**score** tab) with a **New Score** of **1000** and **check** that the **Relative** option is **enabled**.

**6. Add the Next Room** action (**main1** tab) to **move** to the **next room**.



- **No properties** are **set** here.



# Learning Programming With Game Maker

7. Finally, add the **End Block** action (control tab) to end the **conditional** action **block**.

- The complete set of actions should now look like Figure 3-17.



o indicate that they belong

**Figure 3-17.**

***Note that the actions in the block are indented.***

# Learning Programming With Game Maker

- **GAME TIME!**
- It is time to try out the game again.
- Save and play the game to check that you can go from one level to the next by visiting all the moons.
- You can also load this version of the game from the file `Games/Chapter03/galactic3.gm6` on the CD.
- However, if you end the game by visiting all the moons, you'll get an error message indicating that it has run out of levels.
- Don't worry – we'll fix it when we add the finishing touches to the game.



# Learning Programming With Game Maker

## Finishing Touches

- To **finish** the game, we'll add an opening **title screen**, a **help screen**, and a **congratulatory message** displayed when completing the game.
- We'll also include a **few visual touches** to add a **little bit of variety** in the **moons** and **asteroids**.

## A Title Screen

- To create the **title screen**, we need a **new object** to display the **name of the game** and **perform some initial tasks** (game initialization).
- We'll **have it start the music** and **set the initial score**, and then **wait for the player to press a key** before taking them to the **first level**.

# Learning Programming With Game Maker

Creating a new title object resource for the game:

1. **Create** a new **sprite** called **sprite\_title** using **Title.gif**.
2. **Create** a new **object** called **object\_title** and give it this **sprite**.

**Set** the **Depth** property to **1** so that the moons go in front of it and the asteroids behind.

- **3. Add a Create event.**

This will contain the **actions** to **start the music** and **set the score**, but we've already created these in the special moon object, so we can simply move them over.

4. **Open** the **special moon Object Properties** form from the **resource list** and **select** the **Game Start** event to view its **actions**.

# Learning Programming With Game Maker

5. Drag and drop the two actions (start the music and set the score) from the special moon Game Start event into the Create event of the title object.

The Game Start event in the special moon should now be empty, and so it will delete itself automatically when the Object Properties form is closed.

Do this now by clicking OK on the special moon's properties form.



1. Add a Key Press, <Any key> event to the title object and include the Next Room action in the action list for this event (main1 tab).

- Next we need to create a new room for the title screen.

# Learning Programming With Game Maker

Creating a new title room resource for the game:

**1. Create** a new room called **room\_title** and give it an appropriate **caption**.

Also **set** the **room's background** in the same way as before.

**2. Add a few moon and asteroid instances** to the **room** (just for effect).

**3. Place an instance** of the new **title screen object** in the **center** of the **room**.

**4. Close the room properties.**

**5. To be sure** that this is the **first room** in the **game**, **drag** the new **room** to the **top of the list** of **rooms** in the resource list.

- Now **test the game** to check that this all works correctly.

# Learning Programming With Game Maker

## Winning the Game

- We also need to stop the game from producing an error at the end and congratulate the player instead.
- Similar to how we created the title room, we will create a finish room with a finish object to display the message and restart the game.

## Creating a new finish object resource for the game:

**1. Create a new object called `object_finish`.**

It **doesn't** need a sprite.

**2. Add a `Create` event to the `object` and include the `Display Message` action in it (`main2` tab).**



**Set the `Message` to something like:**

**“Congratulations! You’ve delivered all the mail.”**

# Learning Programming With Game Maker

3. Include a **Set Score** action, with a **New Score** of 2000 and the **Relative** option enabled.



4. Include the **Show Highscore** action, with **Background**, **Other Color**, and **Font properties** set as before.



5. Finally, include the **Restart Game** action.

- Now that we have the object, we can create a room for it to go in.

**Creating a new finish room resource for the game:**

1. **Create a new room and place 1 instance** of the **new finish object** inside it.

As this **object** has **no sprite**, it will appear as a **blue ball** with a **red question mark** on it.

This will **not appear** in the game, but it **reminds us** that this (**invisible**) **object is there** when we are editing the room.



# Learning Programming With Game Maker

- Now test the game to check that you can complete it – and that you get the appropriate message when you do (in other words, not an error message!)

## Adding Some Visual Variety

- At this point, all moons look the same, and the asteroids rotate in unison as they move around the screen.
- But, with a different moon sprite and by using the random command, we can change this.

# Learning Programming With Game Maker

- **Editing the moon and asteroid objects:**

1. **Open the moon object properties** form and click the **Edit button** below the **name of the object's sprite**

(another way of opening the moon sprite's properties).

2. In the **moon sprite's** properties, click **Load Sprite** and **select Bases.gif** instead of the existing sprite.

This **sprite** contains **8 subimages** with different kinds of structures on each moon.

Click **OK** to **close** the **Sprite Properties** form.



# Learning Programming With Game Maker

- **Editing the moon and asteroid objects:**

**3. Back in the moon Object Properties form, select the Create event and add a new Change Sprite action.**

- **Select the moon sprite and type random(8) in the Subimage property.**

- **This will randomly choose one of the 8 different moon sprite images.**



- **Also set Speed to 0 to stop the sprite from animating on its own and changing the subimage.**

**4. Close the Action Properties and the moon Object Properties forms.**

# Learning Programming With Game Maker

5. Add an identical **Change Sprite** action to the **Create event** of the **special moon** object in the same way.

- There is **no need to edit the moon sprite** again, as both objects use the same one.



6. Open the **properties** form for the **asteroid** object and add a new **Change Sprite** action in its **Create event** as well.

- **This time choose the asteroid sprite**, and type **random(180)** in the **Subimage** property.
- There are **180 images** in the **rotating asteroid** animation, so this will **start each one at a different angle**.
- Also type **random(4)** in the **Speed property** so that **asteroids rotate at different speeds**.

# Learning Programming With Game Maker

- **Help Information**
- Once you finish making a game, it's easy to sit back and bask in your own creative genius, but there is one more important thing to do before moving onto your next game.
- It's obvious to you how to play this game, but always remember that it is rarely obvious to a newbie.
- If players get frustrated and stuck because they can't figure out the controls, then they usually assume it is just a bad game rather than giving it half a chance.
- You must always provide help in your game to explain the controls and basic game idea.
- Game Maker makes this easy through its **Game Information**.

# Learning Programming With Game Maker

- **Adding game information to the game:**
  1. **Double-click on Game Information** near the **bottom** of the **resource list**.
  2. **A text editor opens and you can type game information in different fonts and colors.**
  3. **Typically** this has the **game's name**, the **author(s) name(a)**, a short **description of goals**, and a list of **controls**.
  4. **Click the green checkmark to close the editor.**
    - That's it.
    - **Press the F1 key during game play**, and the **game pauses** until the help window is closed.
    - **Test the game one last time to check that this final version** works correctly. Final version Games/Chapter03/galactic4.gm6

# Learning Programming With Game Maker

## Congratulations

- You've now completed your second game with Game Maker.
- You might want to experiment with the game a bit further before continuing as there is much more you could do with it.
- To start with, you could make more levels with faster-moving asteroids or smaller moons to make it harder to land on them.
- There are larger planet sprites and smaller planetoids to experiment with, so see what you can come up with.

# Learning Programming With Game Maker

## SUMMARY

- This chapter showed you **more features** of Game Maker.
- You **used events and actions to change sprites and objects.**
- You also used the **Depth property of objects to control the order that instances appear** on screen.
- This chapter also introduced the use of *variables*, *even though we didn't call them variables.*
- For example, the **word direction** is a **variable indicating the current direction** of an instance.
- We also used the **variables x and y** that **indicate the position** of an instance.
- There are many variables in Game Maker, and they are very useful. Other **variables** will be used in the chapters to follow.