CHAPTER 5

Collision Theory and Logic

Objectives

After completing this chapter, you will be able to:

- **Use** computer tools to create programming and artwork for a video game.
- **Create** a playable game using a game engine.
- **Create** animated objects.
- **Debug** a video game build.
- **Engage** in constructive criticism.

Name: _____

Date: _____

Class: _____

Bellwork

Day	Date	Activity	Response
1		Complete the anticipation guide for this chapter.	
2		A computer programming event is a cause-and-effect relationship. Other than action-reaction, what is used in programming to create a logic statement?	
3		Which logic operator is used when a condition is false?	
4		Write a logic statement that would state how the game should act if a chicken walked into a wall.	
5		Describe how to fix the error for when a coyote object stands on a hole and does not fall into it.	
6		Which toolbar in The Games Factory 2 holds the buttons to switch between the frame editor and the event editor?	
7		Which toolbar or window in The Games Factory 2 allows the programmer to select objects included with the program?	
8		Why would a designer want to use the **Zoom** tool?	
9		List three properties that can be changed in the **Properties** window in The Games Factory 2.	
10		In The Games Factory 2, what is the library?	

Name: _____

Day	Date	Activity	Response
11		List the steps needed in The Games Factory 2 to make a truck object move left, turn around, move right, and repeat along a path. Sketch buttons if needed to help explain.	
12		Draw a picture of the button used to create a new object in the **Event Editor** in The Games Factory 2.	
13		How do you insert a new object on a frame in The Games Factory 2?	
14		List the steps in The Game Factory 2 for saving a game as a stand-alone application.	

Note: there are more days of bellwork than days of activities. This is to allow extension of the days for completion of activities. Review all bellwork before the chapter test.

Name: _____

Date: _____

Class: _____

Anticipation Guide

Before Reading the Chapter

Read each statement in the table below. In the column titled Before Reading, write the letter *T* if you agree with the statement or *F* if you disagree with the statement.

After Reading the Chapter

Re-read each statement in the table below. In the column titled After Reading, write the letter *T* if you agree with the statement or *F* if you disagree with the statement. Be prepared to justify your answers in a class discussion.

Before Reading	Statement	After Reading
	Collision theory is the rules concerning how objects react in games when they touch each other.	
	A major component of logic is the "because" statement.	
	Most logic in games can be expressed as a cause-and-effect relationship.	
	Programmers refer to a path where a folder is located as a program tree.	
	It is not possible to combine logic statements. Each condition must be individually programmed!	
	The Games Factory 2 is an object-based programming product.	

Name: _____

Date: _____

Class: _____

Activity 5-1
Logic and Collision in Practice

Objective

Students will use a game engine to create a playable video game.

Situation

You are a new employee of the Really Cool Game Company. Your first task is to learn their game engine technology so you can build logic and collision statements. Complete the task provided to build design skills. You will create a game where the player must "eat" fruit falling from a plane while avoiding balls.

How to Begin

Launch The Games Factory 2 (TGF2). Your instructor will provide specific instructions for doing this, if needed.

Setting the Background Color


New

1. Click the **New** button on the **Standard** toolbar to begin working on a new game.
2. Double-click the thumbnail for Frame 1 to begin making a title page. See **Figure 1.**

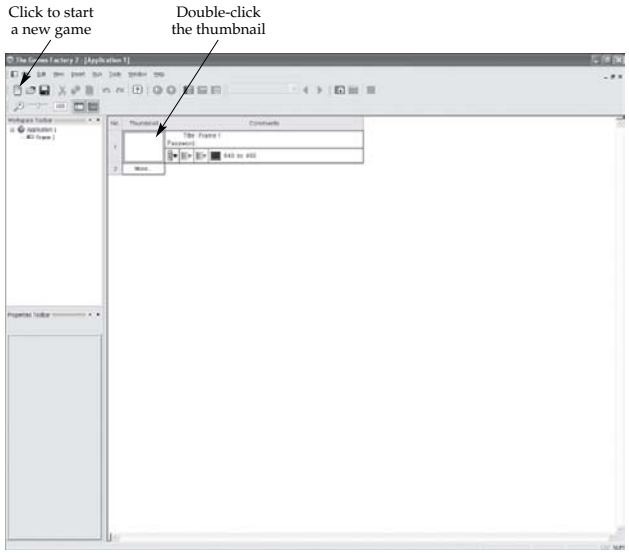

Figure 1

3. In the **View** pull-down menu, select **Toolbars**. Then, check all of the toolbars *except* the **Layers** toolbar. The toolbars that are checked will be visible. Unchecked toolbars are not visible.
4. In the **Workspace** window, right-click on the **Frame 1** branch and select **Rename**. The name is replaced with a textbox with the existing name highlighted. Type **Title Frame** and press the [Enter] key to change the name.

TGF2 is an object-oriented program. This means you will be referring to the names of various objects as you design a game. For this reason, it is very important to name the objects in your game build using logical names. The names **Button1**, **Button2**, and **Button3** are not very meaningful. The names **Close Button**, **Fire Button**, and **Move Button** are easily identified.

5. In the **Properties** window, pick the **Settings** tab, if it is not already displayed.
6. Locate the **Background color** property, which sets the background color of the frame. Then, click the small, white color swatch in the right-hand column. This displays the color palette. See **Figure 2.**
7. Select the bright blue color swatch. The RGB value of this color swatch is 0,0,255. You may need to resize the **Properties** window to see the entire color value.

Adding a Title

You have now set the background color for the first frame of the game. The entire frame should be bright blue. The next thing you need to do is add a title. First, you need to create a place to write the title.

8. In the **Insert** pull-down menu, select **New Object**. The **Create New Object** dialog box is displayed.
9. In the **Create New Object** dialog box, select **All Objects** in the list on the left-hand side of the dialog box. Then, select **Static Text** on the right-hand side of the dialog box. See **Figure 3.**
10. Click the **OK** button to close the dialog box. Then, click near the top and center of the frame to place a text box on your title page.

Figure 2

Name: _____

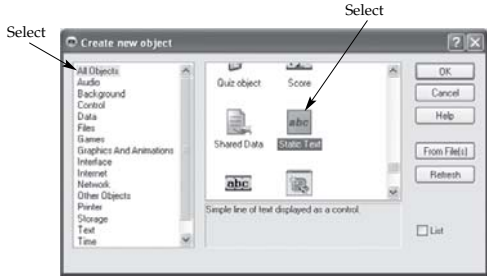

Figure 3

The exact location of the text box is not important at this time. It can be moved later. After the text box is placed, it is automatically selected. Notice that it is numbered (1). Also, notice the tree in the **Workspace** window has grown with the addition of a **Static Text** branch. This branch is the text box you just added.

11. Double-click on the text box. The **Enter New Text** dialog box is displayed. Type **Eating on the Run** and pick the **OK** button to add the text.
12. Click anywhere off of the frame background to deselect the text box. See how the text appears.
13. Select the text box by clicking on it.
14. In the **Settings** tab of the **Properties** window, uncheck the **Border** check box. Also, change the **Background color** property to the same bright blue used earlier.
15. In the **Text Options** tab of the **Properties** window, click on the right-hand column for the **Font** property. This displays the **Font** dialog box. Select **Arial** in the **Font:** list, **Bold** in the **Font Style:** list, and **22** in the **Size:** list. Then pick the **OK** button to change the font.
16. Also in the **Text Options** tab of the **Properties** window, change the **Color** property to white.
17. In the **Movement** tab of the **Properties** window, make sure the **Type** property is set to **Static**. If not, click on the right-hand column for the property to display a drop-down list and select **Static**.
18. In the **Runtime Options** tab of the **Properties** window, check the **Create at start** check box.
19. In the editor window, click on the text box to show the sizing handles.
20. Stretch the text box to the left or right so it is big enough to show the entire title, **Figure 4.**

Saving Your Work

Now is a good time to save your work. It is a good practice to save after completing a major step in your design. It is also good practice to save at a regular time interval, such as every 15 minutes.

Save

21. Click the **Save** button on the **Standard** toolbar. Since this file has not yet been saved, the **Save As** dialog box is displayed.
22. In the **Save As** dialog box, enter **Eat and Run** in the **File name:** text box. Also, navigate to the folder designated by your instructor. Finally, pick the **Save** button to save the game.

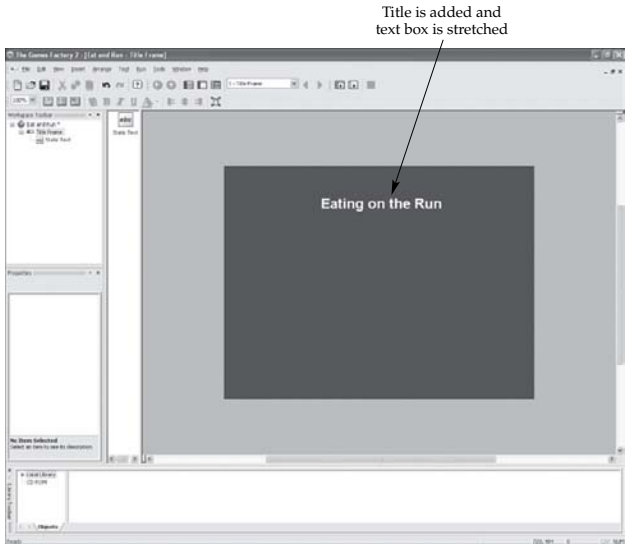

Figure 4

Now that the game has been saved once, clicking the **Save** button on the **Standard** toolbar simply saves the file. The **Save As** dialog box will not be displayed unless you select **File**>**Save As**.

Creating a Start Button

Next, you will insert a **Start** button into the game interface. This is the button the player will click to begin the game.

23. In the **Insert** pull-down menu, select **New Object**. This opens the **Create New Object** dialog box you used earlier.
24. On the left-hand side of the **Create New Object** dialog box, select **Interface**. Then, on the right-hand side of the dialog box, select **Button**. Finally, pick the **OK** button to close the dialog box.
25. Click anywhere on the frame to place the button. The location is not important since you can move it later.
26. Double-click on the button you just inserted to open the **Enter New Button Text** dialog box. Change the text to Start and pick the **OK** button to close the dialog box and update the button.
27. In the **Workspace** window, right-click on the Button branch and select **Rename** from the shortcut menu. Rename the branch Start Button and press the [Enter] key, **Figure 5.**

Is the button ready to go? Recall the discussion in the textbook reading. Just because this object *looks* like a **Start** button does not mean it will *act* like a **Start** button. You must program the button to function as a **Start** button. Using the event editor in TGF2, you will

Name: _____

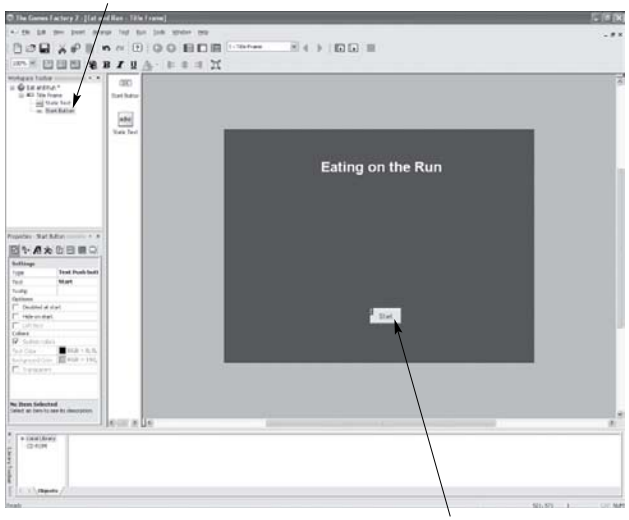

Figure 5

program the button to move the player to the next game frame when the button is clicked. The logic statement is: **IF** the button is clicked, **THEN** move the player to the next frame. If you get used to using this programming logic, then you can organize your thoughts and become a better programmer. In TGF2, the **IF** side of the logic statement is usually the new condition. The **THEN** side of the logic statement is one of the check mark conditions in TGF2.


Event Editor

28. Pick the **Event Editor** button on the **Navigate** toolbar. This changes the editor window to the event editor. There is currently one row with the entry New condition in the first column.
29. Click on New condition. This opens the **New Condition** dialog box, **Figure 6.**
30. Right-click on the button to display a shortcut menu. Select **Button Clicked?** in the shortcut menu.

This adds a new row to the event editor. The condition is when the button is clicked. This is the **IF** side of the logic statement. Now, you need to add the events, which are the **THEN** side of the logic statement.


Storyboard Controls

31. In the **Storyboard Controls** column, right-click in the cell for the Button clicked row.
32. Select **Next frame** from the shortcut menu, **Figure 7.**

A check mark is placed in the cell to indicate a storyboard control event is set for the condition. Hurray! You have programmed your first event. Save your work.

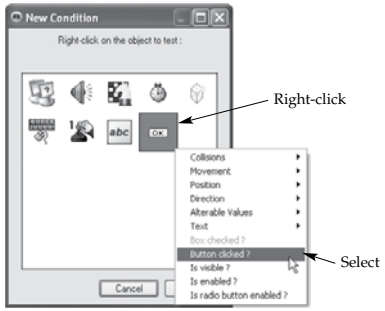

Figure 6

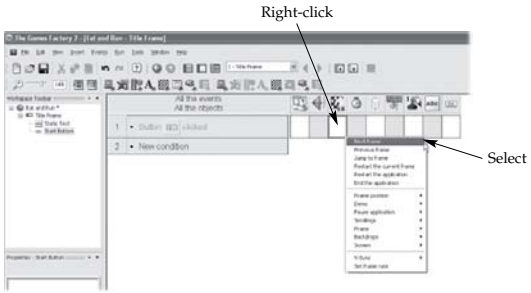

Figure 7

Inserting Objects

Now, you need to place more objects on your title frame. This is done in the frame editor. You used the frame editor earlier to add the title text and the **Start** button. First, switch to the frame editor by picking the **Frame Editor** button on the **Standard** toolbar. Then, continue as follows.

Frame Editor

33. In the **Library** window at the bottom of the screen, expand the **Local Library** branch by clicking on the plus sign (+).
34. Expand the Games branch and select the Miscellaneous branch. The "leaves" in the Miscellaneous branch are displayed on the right-hand side of the **Library** window. These are the library files in the selected category.
35. Double-click on the Fruits library file in the right-hand side of the **Library** window, **Figure 8.** The available objects in the Fruits library file are displayed.
36. Drag-and-drop one each of these objects onto the title frame: SApple2, SBanana1, SCherry2, and SGrape.
37. Place a text box (static text) below each fruit object. Remember, to insert static text, select **Insert>New Object** and then select **Static Text** in the **Create New Object** dialog box.
38. Change the text displayed in the text boxes to: Cherry: 5 points, Apple: 10 points, Banana: 15 points, and Grapes: 20 points. Remember, to change the text, double-click on the text box and enter the new text in the dialog box that appears.

Name: _____

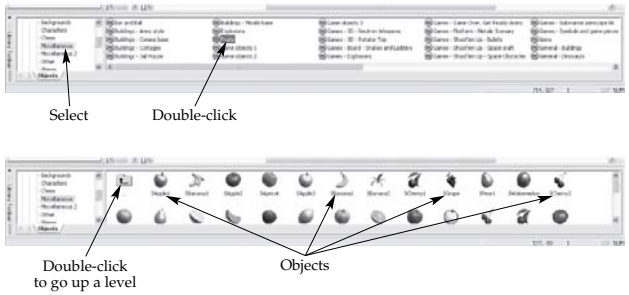

Figure 8

39. Using the **Properties** window, change the properties of each text box so it looks good on screen. For example, you may want to change the font size or color or the background color. Also, resize the text boxes as needed using the handles.
40. Align the fruit and static text objects in a row near the top of the frame. Select each object by clicking on it, then drag it into position. Multiple objects can be selected by holding down the [Shift] key. You can also use the arrow keys to nudge the objects as needed.
41. In the **Library** window, move up one level to see all of the library files in the Miscellaneous branch. Then, double-click on the Game Objects 2 file.
42. Drag-and-drop one each of these objects onto the game frame: Ball 1, Ball 4, and Ball 8.
43. Add a static text box to label these objects as Avoid These Objects, **Figure 9.**
44. Save your work.

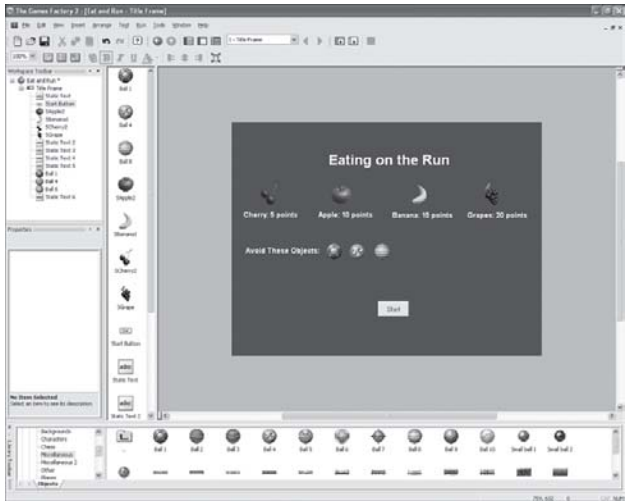

Figure 9

Selecting a Player Avatar

Time to choose an avatar for the player. The avatar is what the player will control. The player will move the avatar around the game frame to gobble up the fruit while avoiding the balls.

45. In the **Library** window, select the Characters branch in the Games branch. Then, double-click on the Various Characters library file.
46. Drag the Soldier object and drop it at the bottom-right corner of the game frame.
47. Resize the character by clicking on it to display the handles. Then, drag the handles to make the object bigger. Its height should be roughly 1/4 of the frame height. Be sure to use the corner handles to resize the object so it remains proportional.
48. In the **Properties** window for the soldier object, select the **Movement** tab. Then, click on the **Type** property, currently set to **Static**, and select **Path** from the drop-down list that is displayed.
49. Click the **Edit** button next to the **Edit Movement** property. The **Path Movement Setup** dialog box is displayed, **Figure 10.**


New Line

50. Click the **New Line** button. A line appears attached to the cursor and the object. This will allow you to draw the path the object will travel.
51. Move the cursor near the left-edge of the frame. Keep the line very straight across the screen. Click to create the path. The object will follow this line when the game is played.
52. In the Path Movement Setup dialog box, change the **Speed:** setting to 15, **Figure 11.**

The lower the speed number, the slower the object will move. Make sure the path is selected (flashing white) when setting the speed setting. If necessary, click the first endpoint (on the object) to select the path.


Reverse at End

53. Click the **Reverse at End** button to have character turn around when it reaches the end of the path.


Loop the Movement

54. Click the **Loop the Movement** button to have the character walk back and forth along the path.
55. Click the **OK** button to close the **Path Movement Setup** dialog box.


Run Frame

56. How does it all look? Click the **Run Frame** button on the **Standard** toolbar. A new window opens with your title frame running. Cool!
57. Click the Windows close button (red X) to close the frame preview.

Remember, you should rename each object to logical names. Doing so will help you keep all of the elements of your game organized. This is a good time to rename all objects you have added to the game. Also, if needed, adjust the location of the objects. For example, the avatar may be walking over the start button. Be sure to save your work before continuing.

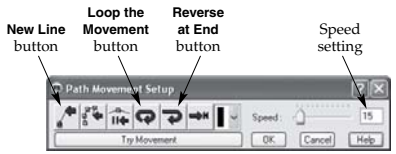

Figure 10

Name: _____

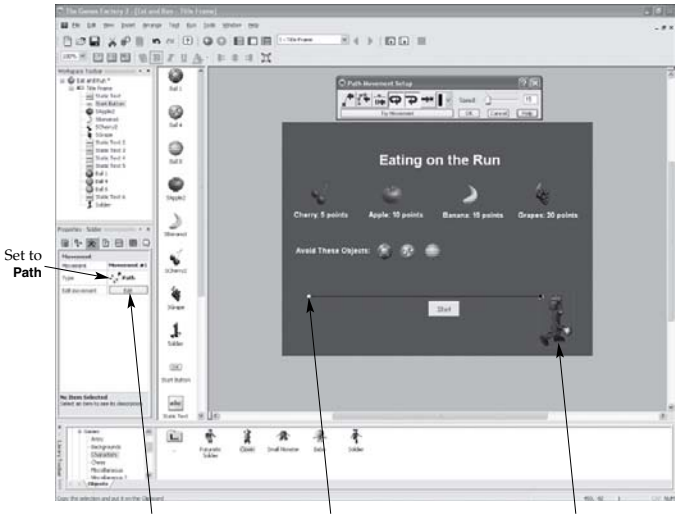

Figure 11

Game Frame

What you have created so far? You have a frame that looks pretty and contains some animation. Is this a video game? No, it needs to be interactive to be a game.


Storyboard Editor

58. Click the **Storyboard Editor** button to display the storyboard. This is where you will add a new frame.
59. Click on the label for the second row to add a new frame (row), **Figure 12**.
60. Double-click on the thumbnail for frame 2 to open the frame in the frame editor.
61. Change the name of the frame to Game Frame.
62. Change the background color of the frame to the same blue used on the title slide.

You will use a shortcut to save time in putting your objects in the game frame. Look at the **Workspace** window. All of the objects you have added are listed in the Title Frame branch. You can drag any of these objects onto the current frame. Here is an example of why it is good practice to rename objects. If you did not rename the static text objects, you would have six objects all named Static Text.

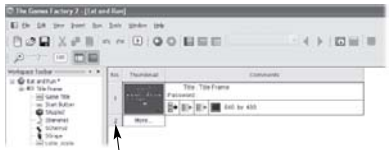

Figure 12

63. Drag the avatar object from the **Workspace** window into the editor window and drop it near the bottom-center of the frame.
64. Drag the apple object into the editor window, but do not put it on the actual frame. By placing it in the editor, but not on the actual frame, it can be set to drop from the top of the screen.
65. Drag the banana, cherry, grapes, and three ball objects into the editor window, but do not drop them onto the actual game frame, **Figure 13.**
66. Select the avatar object in the editor window.
67. Click the **Movement** tab in the **Properties** window.
68. Click the **Type** property that is currently set to **Path** and select **Mouse Controlled** from the drop-down list.
69. Click the **Edit** button for the Edit movement property.

The **Mouse Movement Setup** dialog box is displayed. Also, a bounding box with handles is displayed around the object (avatar). This box defines the space within which the object can move.

70. Using the sizing handles, stretch the box to the left and right edges of the frame. Leave 1/4" space at the edge. Move the top and bottom edges of the bounding box to the top and bottom of the avatar's head, **Figure 14.**
71. Click the **OK** button on the **Mouse Movement Setup** dialog box to close it.
72. Click the **Run Frame** button to test your movement.


Run Frame

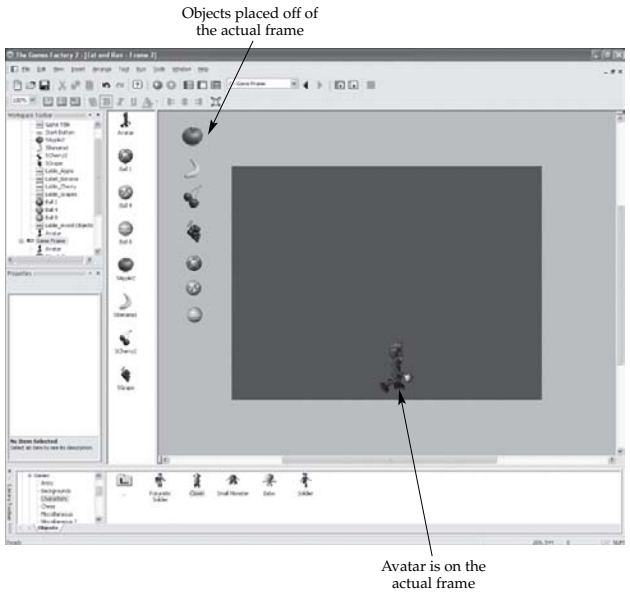

Figure 13

Name: _____

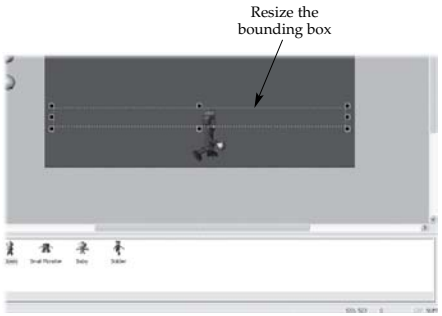

Figure 14

73. Move the mouse to see if the avatar stays on screen.
74. Press the [Alt][F4] key combination to close the preview window.
75. Edit the movement if needed to keep the avatar on screen.
76. Save your work.

Collision Theory in Practice

The first logic statement will deal with the avatar object. **IF** the avatar collides with a banana, **THEN** the banana is destroyed **AND** the player 1 score is increased by 10 points. To make this happen, it must be programmed into the game using the event editor. Pick the **Event Editor** button to display the event editor.


Event Editor

Notice there are no events, even though you already added one for the **Start** button. This is because frame 2 (Game Frame) is the current frame. If you right-click on frame 1 (Title Frame) in the **Workspace** window tree and select **Event Editor** from the shortcut menu, the event for the **Start** button is displayed. Make sure the event editor for frame 2 is current and continue as follows.

77. Right-click on New condition.
78. In the **New Condition** dialog box, right-click on the avatar and select **Collisions**>**Another object** from the shortcut menu. The **Test a Condition** dialog box is displayed.
79. In the **Test a Condition** dialog box, select the banana and click the **OK** button.

This sets the **IF** side of the logic statement. Notice how this appears in the event editor. Now you need to set the **THEN** side of the logic statement.

80. Right-click in the cell under the banana column and select **Destroy** from the shortcut menu. This means that when the avatar collides with the banana, the banana will be destroyed (not the avatar).
81. Right-click in the cell under the **Player 1** column and select **Score**>**Add to Score** from the shortcut menu. In the **Add to Score** dialog box that appears, type 15 and click the **OK** button, **Figure 15.**
82. Using the same techniques, set the events for the cherry, apple, and grapes. Add the number of points indicated in the labels on the title slide. If a different number of points is added, it would be considered a bug or glitch in the game.
83. Save your work.

Figure 15

The basic collision theory here will make the fruit go away when it touches the character. When this happens, points are added to the player's score. The balls are a little different. The logic statement for these is **IF** the avatar collides with a ball, **THEN** the player loses one life **AND** the ball is destroyed.

84. Right-click on **New condition**.
85. In the **New Condition** dialog box, right-click on the avatar and select **Collisions**>**Another object**.
86. In the **Test a Collision** dialog box, select the first ball and click the **OK** button.

The **IF** side of the logic statement is set. Notice that it is similar to the logic statements for the fruit objects. The **THEN** side of the logic statement is different from the fruit objects.

87. Right-click in the cell under the **Player 1** column for the collision statement between the avatar and the ball. Select **Number of lives**>**Subtract from number of lives** from the shortcut menu, **Figure 16.**
88. In the **Set Number of Lives** dialog box that is displayed, type 1 and click the **OK** button.
89. Right-click in the cell under the **Ball 1** column (the name of the ball object selected earlier) and select **Destroy** from the shortcut menu.
90. Follow similar steps to have the other two ball objects subtract one life on collision with the avatar.
91. Save your work.

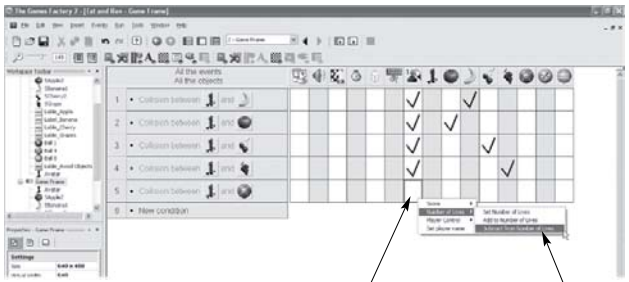

Figure 16

Name: _____

The Drop

To have the objects drop from the sky, you need to add an object to generate the drop. You now will add a plane to fly across the screen and drop the objects.


Frame Editor

92. Select **Game Frame** in the **Workspace** window. Then, pick the **Frame Editor** button on the **Standard** toolbar.
93. In the **Library** window, select **Games**>**Planes**. Then, double-click on the **3D Jet Airline** library file. Finally, select the object **Fighter 4** and drag it to the top-left corner of the game frame, **Figure 17.**
94. Select the plane object and set its movement property to **Path**.
95. Draw a straight-line path for the object across the top of the game frame.
96. Set the speed to 20.
97. Click the **Reverse at End** and the **Loop the Movement** buttons. Then, close the **Path Movement Setup** dialog box.
98. Save your work.

Create a New Object

All of the objects needed for the game are in place. Now, you only need to set the events to make it all work. The event editor is where this is done. The first logic statement is **IF** 3.4 seconds elapse, **THEN** create a banana that will fall from the fighter.


Event Editor

99. Click the **Event Editor** button. Make sure the event editor is displayed for the game frame. If not, right-click on the **Game Frame** branch in the **Workspace** window and select **Event Editor** from the shortcut menu.

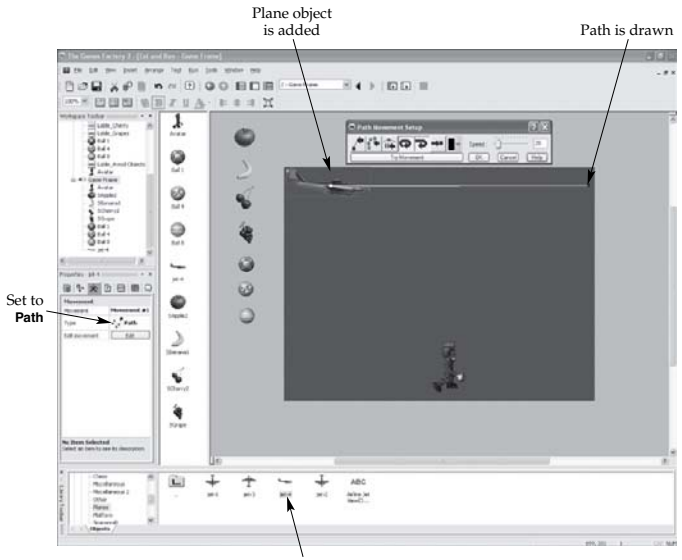

Figure 17

100. Right-click on New condition.
101. In the **New Condition** dialog box, right-click on the timer and select **Every** from the shortcut menu, **Figure 18.**
102. In the **Every** dialog box that is displayed, enter 3 in the **Seconds** text box, 40 in the **1/100** text box, and pick the **OK** button.


Create New Objects

103. Right-click in the cell in the **Create New Objects** column for the condition you just added and select **Create Object** from the shortcut menu.
104. In the **Create Object** dialog box that is displayed, select the banana and click the **OK** button.

The frame editor is displayed and a different **Create Object** dialog box is opened, **Figure 19.** Now, you need to tell the computer where the banana will start to fall. It should fall from the exact spot where the fighter is when the timer counts off 3.4 seconds.

105. In the **Create Object** dialog box, select the **Relative to…** radio button. The **Create Object** dialog box is hidden and the **Choose an Object** dialog box is displayed.
106. In the **Choose an Object** dialog box, select the plane and click the **OK** button. The **Create Object** dialog box is redisplayed.
107. Set the X and Y coordinates to 0 and 0 to make the object appear in the center of the plane. Click the **OK** button to close the **Create Object** dialog box.

You are returned to the event editor. Using similar steps, create an event for each of the fruit and ball objects. Drop a cherry every 1.10 seconds. Drop an apple every 5.00 seconds. Drop a grape every 6.80 seconds. Notice how the higher-point-value objects appear less frequently? Drop one type of ball every 7.20 seconds. Drop a second type of ball every 8.10 seconds. Drop the third type of ball every 10.30 seconds. Save your work.

Setting the Object Speed

Now, you need to set the speed for the fruit and ball objects. You will also set the motion for the objects. Switch to the frame editor for the game frame. Then, continue as follows.

108. Select the banana.
109. In the **Properties** window, select the **Movement** tab.
110. Change the **Type** property to **Bouncing Ball**. A new set of properties is displayed in the **Properties** window.

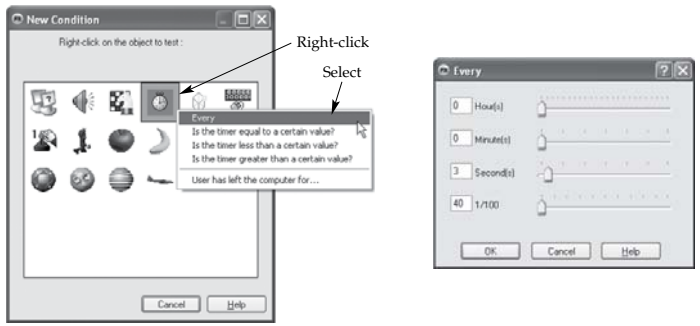

Figure 18

Name: _____

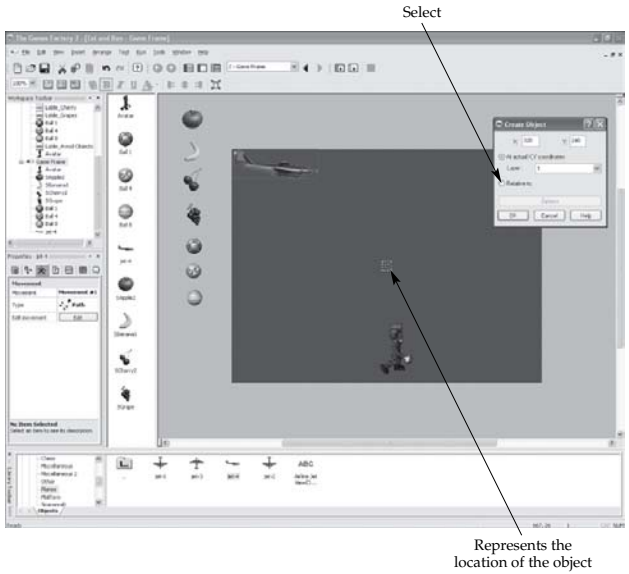

Figure 19

111. Change the **Speed** property to 12.
112. Click the **Initial Direction** property.

A compass window is displayed, **Figure 20.** The wheel selects the direction the object will start. Multiple directions can be set to randomize the movement. To clear all the arrows, click the **Reset** button at the lower-left corner of the window.


Reset

113. Click the block two to the left and right of the six o'clock position. You should have two arrows.

Using similar steps, change the remaining objects. All should be set to bouncing-ball motion with the same initial directions. The cherry speed should be 30, the apple 60, and the grape 75. The three balls should have speeds of 65, 75, and 85. Save your work.

Setting Boundaries

Right now, the objects will fall off of the screen edges. However, they should bounce around the screen as if the edges are the bumpers on a pool table. A boundary needs to be set to keep the objects from falling off of the edge of the screen. In the event editor for the game frame, set a new condition for this logic statement: **IF** the banana leaves the play area to the left or right, **THEN** bounce.

114. Right-click on New condition.
115. In the **New Condition** dialog box, right-click on the banana and select **Position>Test Position of SBanana1**. The **Test Position of** *object* dialog box is displayed, **Figure 21.**
116. Click the left and right arrows for **Leaves play area on** *side***?**.
117. Click the **OK** button to close the **Test Position of** *object* dialog box.

118. Right-click in the cell under the banana and select **Movement**>**Bounce** from the shortcut menu.

Using similar steps, set a left and right boundary for the cherry, apple, grape, and three types of balls. To save time, you can drag and drop the check marks. This will copy the reaction programming to the other item. In this case, the properties of the fruit are changed to act like bouncing balls and have them bounce at the edge of the frame. This keeps them in play until they reach the bottom of the screen. Test play the frame to see if everything is working properly. Use the **[Alt][F4]** key combination to close preview window. Save your work.

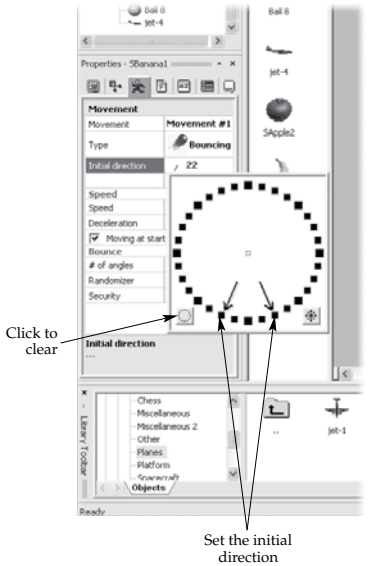

Figure 20

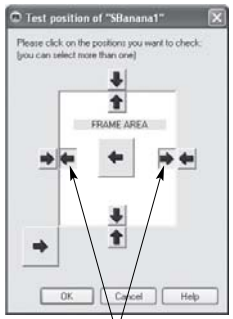

Figure 21

Name: _____

Showing the Score and Number of Lives

On the game frame, there needs to be some way to show the player how many lives are left. Similar to lives, you need to have a way of displaying the score to the player. You will insert new objects for these tasks. Switch to the frame editor for the game frame.

119. Select **New Object** from the **Insert** pull-down menu.
120. Select **Games**>**Score** and then click the **OK** button to close the Create New Object dialog box.
121. Click in the bottom-right corner of the frame to place the score object.
122. Select **New Object** from the **Insert** pull-down menu.
123. Select **Games**>**Lives** and then click the **OK** button to close the **Create New Object** dialog box.
124. Click in the bottom-left corner of the frame to place the lives object, **Figure 22.**

The default value for the lives object is three lives. This is indicated by the three hearts. The number of lives and the icon indicator can be changed. Test the score and lives features using the **Run Application** tool. Run into some scoring objects and life-removing obstacles. Of course, you still need to add programming to end the game when you are out of lives. Close the preview window and continue as follows.


Run Application


Event Editor


Storyboard Controls

125. Display the event editor for the game frame.
126. Add a new condition. In the **New Condition** dialog box, right-click on the **Player 1** icon and select **When number of lives equals 0** from the shortcut menu.
127. In the cell under the **Storyboard Controls** column, right-click and select **Next Frame** from the shortcut menu.

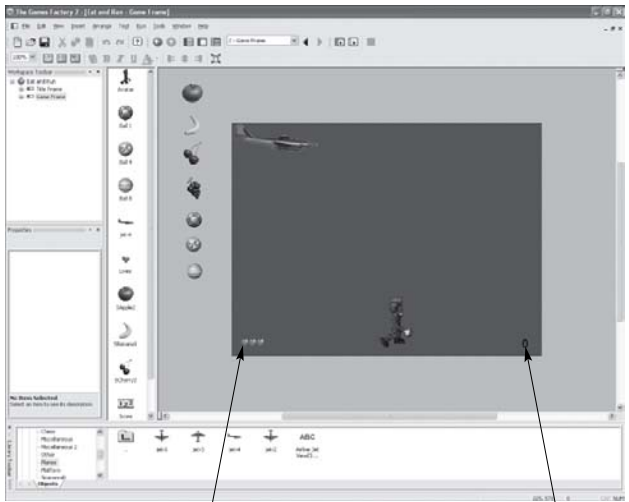

Figure 22

Hall of Fame/High Score Page


Storyboard Editor


Frame Editor

128. Switch to the storyboard editor and create a new frame. Review the previous steps if you do not recall how to do this. Rename the frame as **High Score**.
129. Open the frame editor for the **High Score** frame.
130. In the **Library** window, expand the **Backgrounds** branch, select the **Other** branch, and open the **Games—General Playareas** library file.
131. Drag the **Playtime Background** from the library into the editor window. Align the image so it completely fills the frame.
132. Select **New Object** in the **Insert** pull-down menu. In the **Create New Object** dialog box, select **Games>Hi-Score** and click the **OK** button.
133. Click in the center of the background to place the high score object.
134. Resize the high score object so it covers about 2/3 of the dark gray area of the background. Also, center the object left-to-right on the dark gray area. Leave some room at the bottom for two buttons.
135. Using the **Properties** window, change the font size, colors, etc., to suit your tastes, **Figure 23.**
136. Insert a button and change the name to **Restart**. Review the previous steps if you do not recall how to do this. Place the button below the high score object.
137. Insert another button and change the name to **End Game**.

It is good practice to make the buttons appear the same. They should have the same font, be the same size, and their positions should be balanced on the frame. Review various software that you use. Make note of how buttons appear similar and are consistently located. Users expect certain buttons, such as **OK** or **Cancel**, to be near the bottom of a dialog box. Also, you should rename the buttons in the **Workspace** window. Next, use the event editor to program these two new buttons.


Event Editor

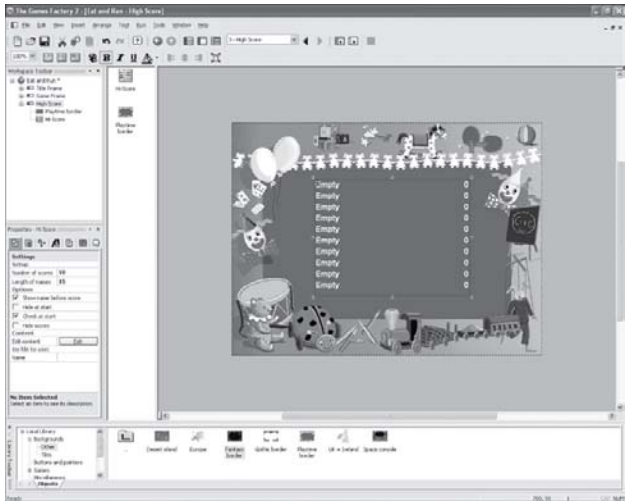

Figure 23

138. Add a new condition. In the **New Condition** dialog box, right-click on the **Reset** button icon and select **Button Clicked?** from the shortcut menu.


Hi-Score

139. Right-click in the cell under the **Hi-Score** column and select **Reset** from the shortcut menu.
140. Add another new condition. In the **New Condition** dialog box, right-click on the **End Game** button icon and select **Button Clicked?** from the shortcut menu.


Storyboard Controls

141. Right-click in the cell under the **Storyboard Controls** column and select **End the application** from the shortcut menu.


Run Application

142. Test the game using the **Run Application** tool, **Figure 24.**
143. Save your work.

Pause or End Game Application

One of the finishing touches is to add the ability to pause or end the game with the touch of a single button. Display the event editor for the game frame. Try programming the game to meet the following logic statements. Use what you have learned in this activity. Hint: you are testing the condition of a keyboard key.

- **IF** on pressing the [Esc] key, **THEN** end the application.
- **IF** on pressing the [P] key, **THEN** pause the application.

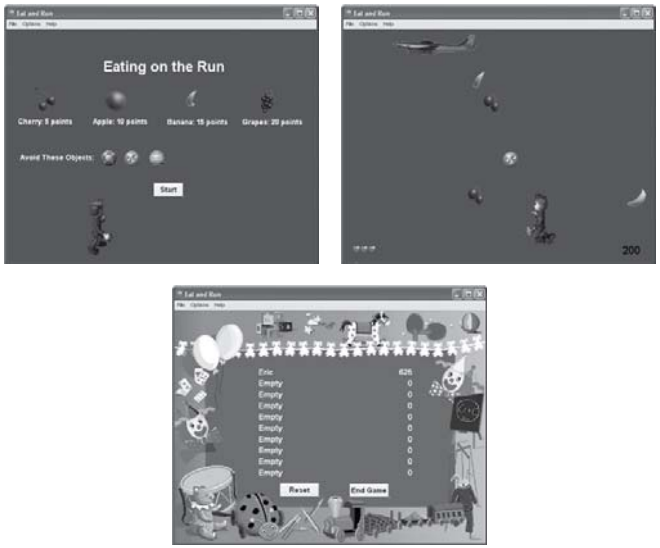

Figure 24

Create as Stand Alone

Once you finish designing your game, you want to make it so anyone can play. You will now "package" the game so it will play on any computer. Save your work before continuing.

144. In the **File** pull-down menu, select **Build>Application**. The **Save As Stand-Alone Application** dialog box is displayed. This is a standard Windows save as dialog box.
145. Navigate to the folder where you want the game file saved.
146. In the **File name:** text box, enter **Eat and Run**. Since this is the name of your TGF2 file, it is the default name. You can enter a different name if you wish.
147. In the **Save as type:** drop-down list, make sure **Stand-alone Applications (*.exe)** is selected. Then, pick the **Save** button.

The game is saved as a self-contained EXE file. See **Figure 25.** You can now burn the game to a CD or attach it to an email for friends to play.

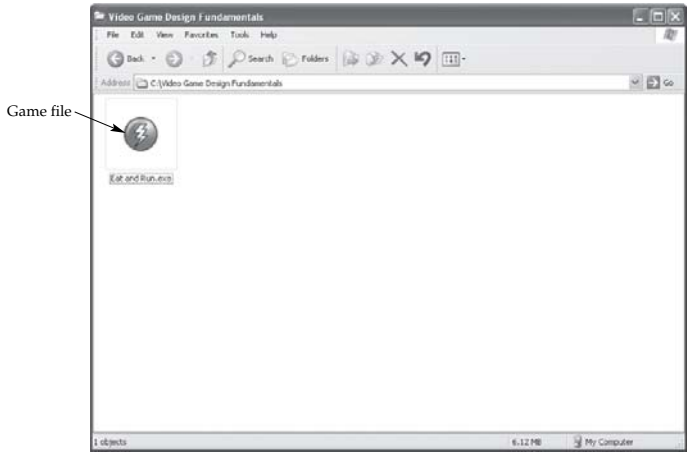

Figure 25

Name: _____

Date: _____

Class: _____

Activity Review

1. Write a logic statement to describe how to program a **Start** button to move to the next frame.

   IF _____

   THEN _____
2. Under which event button are the controls to add points?

   _____

   _____
3. What is the name of the frame that should include general directions so the player knows what to do?

   _____
4. Which button allows the designer to test play just the frame they are working on?

   _____
5. Where do you place an object if you do not want it to show on the frame until programmed to drop?

   _____
6. What type of movement allows the designer to draw a line for the object to follow?

   _____
7. What type of condition is used to make the banana disappear when it touches the player?

   _____
8. What was the user interface for this game?

   _____
9. How do you quickly copy an event from one place to another?

   _____
10. What type of object is needed to record the scores of the best players?

    _____

Name: _____

Date: _____

Class: _____

Activity 5-2
Game Review One

Objective

Students will solicit and accept constructive criticism on a video game design.

Situation

Every designer needs to receive feedback on their design and the functionality of their game. Complete a point rating rubric as a self evaluation of the video game you designed in Activity 5-1. This was a simple game and will not score high, but you need to understand what will make a game better. You will also evaluate another student's game as they evaluate yours. Talk about the good elements of the game, the elements that could be improved. Discuss how you could make the game better if you were required to build this style of game again.

Personal Build Point Rating Rubric

	0	1	2	3	4	5	Score
Concept Is the idea well developed?	No main idea.					Clear throughout.	
Aesthetics Do the look and colors fit the game?	Poor quality graphics and color.					Awesome graphics and theme-based colors.	
Sound Effects Do the sounds play well? Are the music and ambient sounds appropriate?	No sound; sounds too loud or not related to the game.					Good sound for each item at good levels.	
Entertainment What is the fun factor?	Did not work or very boring.					Cannot stop playing this game!	
Replay How likely are you to play this game again?	Game solved, too easy or uninteresting.					Cannot wait to play this again!	
						Total Score (higher is better)	

ESRB rating: _____ Why?

_____

_____

Name: _____

Possible improvements:

_____

_____

_____

_____

Peer Build Point Rating Rubric

Student review by: _____

Game build by: _____

	0	1	2	3	4	5	Score
Concept Is the idea well developed?	No main idea.					Clear throughout.	
Aesthetics Do the look and colors fit the game?	Poor quality graphics and color.					Awesome graphics and theme-based colors.	
Sound Effects Do the sounds play well? Are the music and ambient sounds appropriate?	No sound; sounds too loud or not related to the game.					Good sound for each item at good levels.	
Entertainment What is the fun factor?	Did not work or very boring.					Cannot stop playing this game!	
Replay How likely are you to play this game again?	Game solved, too easy or uninteresting.					Cannot wait to play this again!	
						Total Score (higher is better)	

ESRB rating: _____ Why?

_____

_____

Possible improvements:

_____

_____

_____

_____

Name: _____

Date: _____

Class: _____

Activity 5-3
Gamescape Design and Dimensions

Objective

Students will size a background scene to fit the frame of a video game. Students will use fill colors, fill effects, and line colors to adapt shapes to fit the scene.

Situation

Your company is designing a racing game. The game scrolls top-to-bottom to achieve the objective finish line. Your team is required to produce the background frame as part of the process. The background frame is 3500 pixels high by 600 pixels wide.

How to Begin

You will use Microsoft Paint for part of this activity. Launch Microsoft Paint by selecting the **Start** button on the Windows task bar, then select **Programs**>**Accessories**>**Paint**. If needed, your instructor will provide other instructions for launching Paint. Maximize the window and continue as follows.

Setting the Image Size

1. Select **Attributes...** from the **Image** pull-down menu.
2. In the **Attributes** dialog box, enter 600 in the **Width:** text box and 3500 in the **Height:** text box. Then, pick the **OK** button to set the image size, **Figure 1.**
3. Click the dark green color swatch in the palette at the bottom of the screen.
4. Click the **Fill With Color** button on the toolbar and then click in the drawing area.


Fill With Color

Adding a Road

The entire image is filled with green. You have added "grass" to the background image. Next, you need to build a winding road. This is where the car will be driven.

5. Click the dark gray color swatch in the color palette. You must use dark gray instead of black. Black will be interpreted as transparent areas.

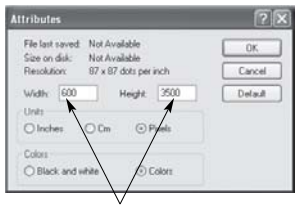

Figure 1

Name: _____


Curve

6. Click the **Curve** tool on the toolbar.
7. Starting at the top of the image and about 1" to the right of the center, draw a straight, vertical line about 4" long.
8. Move the crosshairs of the cursor to the midpoint of the line. Click and hold while dragging to the right to create a curved line. Release the mouse button to set the curve.
9. Click the **Curved** tool again. Draw another line about 2" to the left of the line you just drew. The two lines should look like the boarders of a curving road.


Line

10. Click the **Line** button.
11. Draw a line to close the bottom of the road (and top, if necessary).

Zoom as needed. You must have a complete line all the way across or it will not "seal" the bottom for using the "fill" tool. If you make a mistake, select **Edit**>**Undo** to reverse an action.


Fill With Color

12. Click the **Fill With Color** tool and then click inside of the road boundaries you drew. A segment of concrete road is now created, **Figure 2.**
13. Using similar steps, create a twisting road from the top of the image to the bottom. Remember, the area must be "sealed" to fill it with color.
14. Add one intersection somewhere in the middle. This is where a different road crosses the winding road.
15. At the bottom of the winding road, create a blue-filled section for the starting line.
16. At the top of the winding road, create a checkerboard section for the finish line, **Figure 3.**

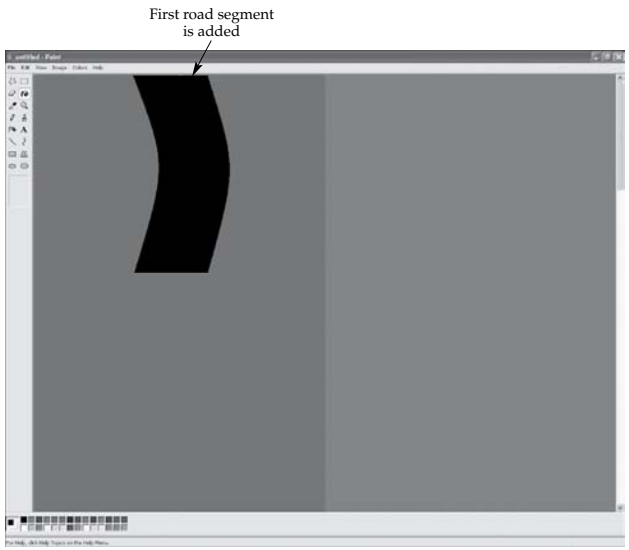

Figure 2

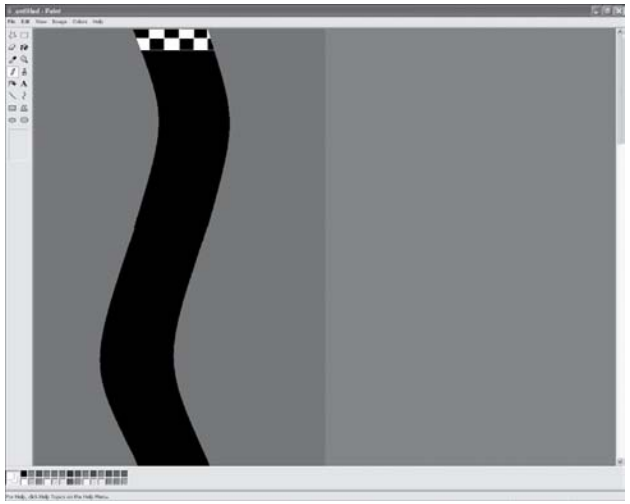

Figure 3

17. Select **Save As...** in the **File** pull-down menu.
18. Navigate to the folder where you wish to save the file. Name the file Driving Course A. In the **Save as type:** drop-down list, select **24-bit Bitmap**. Finally, pick the **Save** button to save the image file.
19. Close Microsoft Paint.

Adding the Background to a Game

You will now create a very simple video game based on the background image you created. The game will include an animated obstacle. Launch The Games Factory 2 (TGF2) and continue as follows. Review the skills learned in developing the game in Activity 5-1. You will use many of the same skills to build this driving game. Note: if you still have the game from the previous activity open, right-click on the game name in the **Workspace** window and select **Close** from the shortcut menu. You can have multiple games open, but having only one open at a time will make things easier.

New

20. Click the **New** button on the **Standard** toolbar to begin a new design.
21. In the **Workspace** window, rename the default frame to Game Frame.
22. Click the thumbnail for Game Frame. Using the **Settings** tab in the **Properties** window, change the **Size** property to 600 × 3500.

Frame Editor

23. Open the frame editor for Game Frame.
24. Select **Insert>New Object**. In the **Create New Object** dialog box, choose **Backdrop** on the left and then double-click the **Backdrop** object on the right. Click anywhere on the frame to place the default backdrop object.

Clear

25. Double-click on the default backdrop object to open the default backdrop image in the image editor.
26. Click the **Clear** button to erase the default image.

Name: _____

Import

27. Click the **Import** button and navigate to the Driving Course A image file.
28. In the **Import Options** dialog box, click the **OK** button to accept the default options.
29. Click the **OK** button at the bottom of the image editor to update the backdrop image.
30. Drag the image without resizing it so it fits the game frame.
31. Insert static text to label the starting line as START.
32. Insert static text to label the finish line as FINISH.
33. Change the properties of the text as needed. Remember to rename the objects to logical names.
34. Save your work as Driving Game.

Adding an Avatar

The "playing field" has been added to the game. Now, you need to add an avatar for the player. Since this is a driving game, you will add a car object.

35. In the **Library** window, expand the Games branch and select Miscellaneous. Then, open the Military library file.
36. Drag the Jeep 1 (passenger) object onto the start area of the game frame, **Figure 4.** Rename the object Jeep.
37. With the Jeep selected, click the **Movement** tab in the **Properties** window.
38. Change the **Type** property to **Racecar**.
39. Uncheck the **Moving at start** and **Enable reverse** properties.
40. Change the **Initial direction** property to be a single arrow pointing up.

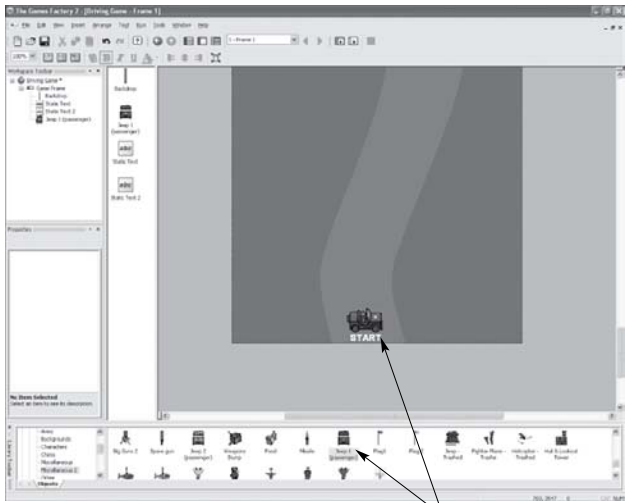

Figure 4

41. From the **Library** window, drag the **Tree** object into the frame. Place it by the edge of the road.
42. In the **Properties** window for the tree, click the **Runtime Options** tab.
43. Change the **Obstacle Type** property to **Obstacle**. This allows a collision event to be set with the tree acting as a solid object.
44. From the **Library** window, drag the **Jeep 2 (passenger)** object and place it on the intersecting road. Rename it **Jeep Obstacle**.
45. Create a path for **Jeep Obstacle** that crosses the intersection and goes off of the frame, **Figure 5.** Loop the movement so that it continues to drive through the intersection as an obstacle.
46. Set the speed of **Jeep Obstacle** to 25.
47. Save your work.

Making It Scroll

Before you can test the game, the scene needs to be set to scroll. Once this is done, you can also finish setting the obstacles and rewards. The first condition you will add in this section is called a *special condition.*


Event Editor

48. Display the event editor for **Game Frame**.
49. Right-click on **New Condition**.
50. In the **New Condition** dialog box, right-click on the **Special** icon. Choose **Always** in the shortcut menu, **Figure 6.**


Storyboard Controls

51. Right-click in the **Storyboard Controls** column and choose **Scrollings>Center vertical position of window in frame** from the shortcut menu.

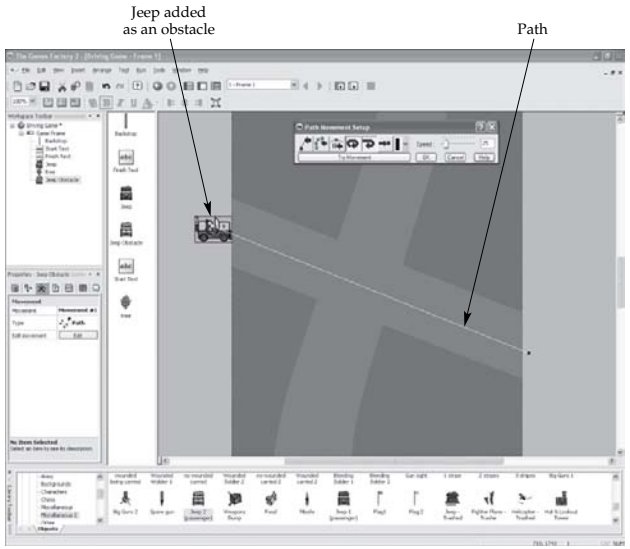

Figure 5

Name: _____

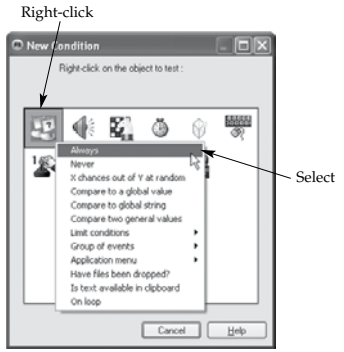

Figure 6

52. In the **Center vertical position of window in frame** dialog box that is displayed, click the **Retrieve data from an object** button, **Figure 7.** The **New Expression** dialog box appears, which looks similar to the **New Condition** dialog box.
53. In the **New Expression** dialog box, right-click on the icon for the **Jeep** object (not **Jeep Obstacle** object). Select **Position>Y Coordinate** from the shortcut menu.

Notice the expression that has been entered in the **Center vertical position of window in frame** dialog box. This indicates the vertical position will be centered on the Y value of the **Jeep** object. The Y coordinate is the up and down (vertical) measurement. The X coordinate is the left and right (horizontal) measurement. You will use this concept in later game designs.

54. Click the **OK** button to close the **Center vertical position of window in frame** dialog box.


Run Frame

55. Test play the game by clicking the **Run Frame** button.

The avatar is controlled with the arrow keys on the keyboard. The up arrow moves the Jeep forward. The down arrow applies the brakes. See if you can keep the Jeep on the road while avoiding the Jeep on the crossroad. When done, close the preview window and save your work.

Setting the Course

Return to the frame editor and place objects along the side of the road so the Jeep will hit an obstacle if it does not stay on the road. Do not leave any spaces big enough so the Jeep can pass through and leave the road. Remember to change the **Obstacle**

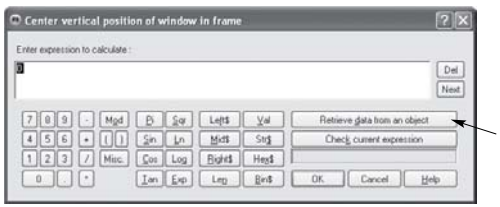

Figure 7

type properties for the objects along the roadside to **Obstacle**. Next, program this logic statement: **IF** the Jeep object collides with an obstacle, **THEN** return the object to the starting position. Display the event editor and continue as follows.

Event Editor

56. Add a new condition. In the **New Condition** dialog box, right-click on the icon for the **Jeep** object (not **Jeep Obstacle** object) and select **Collisions**>**Backdrop** from the shortcut menu.
57. In the row for the new condition, right-click in the cell in the **Jeep** column. Select **Position**>**Select Position...** from the shortcut menu. The game frame is displayed. A semi-transparent X appears on the frame, **Figure 8.**
58. Click and hold the X and drag it to the starting point of the Jeep.
59. Click the **OK** button to close the **Select Position...** dialog box and return to the event editor.

Run Application

Now, if the player hits an object the avatar is reset to the beginning of the road course. Test play the game using the **Run Application** tool. Move any obstacle that needs refinement so your Jeep can complete the course. Save your work.

Victory Condition

Frame Editor

Does your video game meet the definition of a game? No, because it does not have a victory condition. Create a simple condition so when the player completes the course they are the winner. Return to the frame editor.

60. Drag the green flag from the **Library** window and place it on the finish line.
61. Place another flag on the finish line, **Figure 9.**

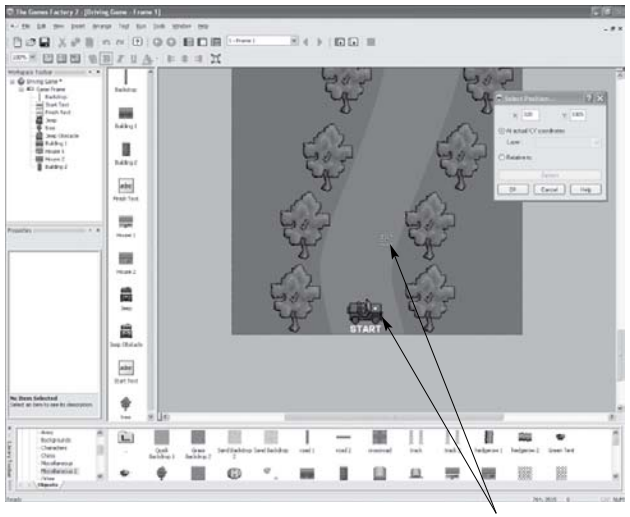

Figure 8

Name: _____

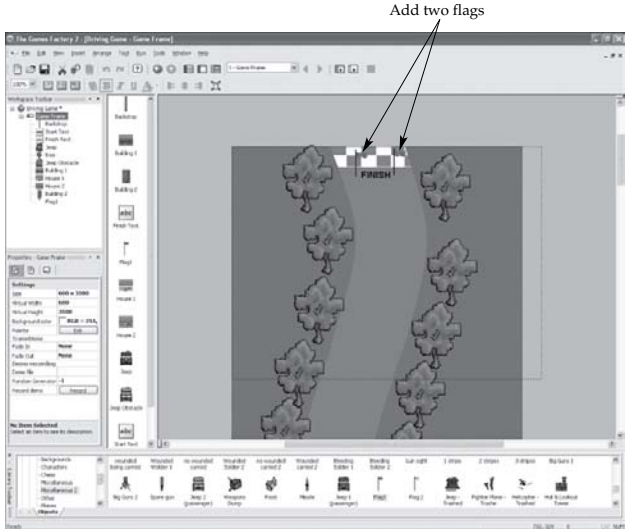

Figure 9

Event Editor

Storyboard Editor

Run Application

62. Create an event such that **IF** the **Jeep** object collides with either green flag, **THEN** the game jumps to the next frame.
63. Add a new frame to the game. Name it **Winner Frame**.
64. Add to the winner frame a decorative background and a large textbox stating **WINNER!**.
65. Test play the game using the **Run Application** tool.
66. Save your work.

Adding Improvements

Now, you have a video game. However, there is not much challenge. The player simply drives from one end of the road to the other end without hitting any obstacles. Add some other features to the game. In Activity 5-1, the video game had a score and lives. Can you program these into the driving game?

- Add a lives object and a condition to take away a life when the Jeep collides with an obstacle.
- Add a score object. Note: the lives and score objects should be located in the dashed rectangle near the top of the frame. This rectangle represents the screen.
- Add objects, such as food and ammo, that will increase the score.
- Add a high score object and a high score frame to the game.
- Add a title page that includes a **Start** button. Note: to rearrange the frames, drag a frame to a new location in the tree in the **Workspace** window.

Be sure to save your work as you go. After testing the final game build, save it as a standalone application that you can share with others.

Name: _____

Date: _____

Class: _____

Activity Review

1. What happens if you do not "seal" a drawn shape before you fill it with color?

   _____

2. What was the user interface for the driving game?

   _____

3. What event setting allows the visible playing area to show different areas of the game frame?

   _____

4. For a tree to act like a solid backdrop object, what runtime options needs to be set?

   _____

5. Which button allows the designer to test play the entire game?

   _____

	0	1	2	3	4	5	Score
Concept Is the idea well developed?	No main idea.					Clear throughout.	
Aesthetics Do the look and colors fit the game?	Poor quality graphics and color.					Awesome graphics and theme-based colors.	
Sound Effects Do the sounds play well? Are the music and ambient sounds appropriate?	No sound; sounds too loud or not related to the game.					Good sound for each item at good levels.	
Entertainment What is the fun factor?	Did not work or very boring.					Cannot stop playing this game!	
Replay How likely are you to play this game again?	Game solved, too easy or uninteresting.					Cannot wait to play this again!	
						Total Score (higher is better)	

Name: _____

	0	1	2	3	4	5	Score
Concept Is the idea well developed?	No main idea.					Clear throughout.	
Aesthetics Do the look and colors fit the game?	Poor quality graphics and color.					Awesome graphics and theme-based colors.	
Sound Effects Do the sounds play well? Are the music and ambient sounds appropriate?	No sound; sounds too loud or not related to the game.					Good sound for each item at good levels.	
Entertainment What is the fun factor?	Did not work or very boring.					Cannot stop playing this game!	
Replay How likely are you to play this game again?	Game solved, too easy or uninteresting.					Cannot wait to play this again!	
						Total Score (higher is better)	

Name: _____

Date: _____

Class: _____

Activity 5-4
Game Review Two

Objective

Students will solicit and accept constructive criticism on a video game design.

Situation

Every designer needs to receive feedback on their design and the functionality of their game. Complete a point rating rubric as a self evaluation of the video game you designed in Activity 5-3. This was a simple game and will not score high, but you need to understand what will make a game better. You will also evaluate another student's game as they evaluate yours. Talk about the good elements of the game, the elements that could be improved. Discuss how you could make the game better if you were required to build this style of game again.

Personal Build Point Rating Rubric

	0	1	2	3	4	5	Score
Concept Is the idea well developed?	No main idea.					Clear throughout.	
Aesthetics Do the look and colors fit the game?	Poor quality graphics and color.					Awesome graphics and theme-based colors.	
Sound Effects Do the sounds play well? Are the music and ambient sounds appropriate?	No sound; sounds too loud or not related to the game.					Good sound for each item at good levels.	
Entertainment What is the fun factor?	Did not work or very boring.					Cannot stop playing this game!	
Replay How likely are you to play this game again?	Game solved, too easy or uninteresting.					Cannot wait to play this again!	
						Total Score (higher is better)	

Name: _____

Peer Build Point Rating Rubric

Student review by: _____

Game build by: _____

	0	1	2	3	4	5	Score
Concept Is the idea well developed?	No main idea.					Clear throughout.	
Aesthetics Do the look and colors fit the game?	Poor quality graphics and color.					Awesome graphics and theme-based colors.	
Sound Effects Do the sounds play well? Are the music and ambient sounds appropriate?	No sound; sounds too loud or not related to the game.					Good sound for each item at good levels.	
Entertainment What is the fun factor?	Did not work or very boring.					Cannot stop playing this game!	
Replay How likely are you to play this game again?	Game solved, too easy or uninteresting.					Cannot wait to play this again!	
						Total Score (higher is better)	