

KEY IN-CLASS STUDENT ACTIVITY 4.3: WORK WITH COLLISIONS

Lesson Objective 4.3:

Work with collisions. *Topics:* rectangle collision, collision response, and fundamentals of physics simulation.

Resources, software, and additional files needed for this lesson:

1. A workstation with Windows 7®, Windows Vista®, or Windows XP®
2. Microsoft Visual Studio® or Microsoft Visual C# Express® (2008 or 2010)
3. XNA® Game Studio 2.0, 3.0, 3.1, or 4.0
4. OrangeBall.png, PurpleBall.png, and TealBall.png, located in 98-374-ENU-4.3-IC-resources.zip

Guiding questions:

1. **What should happen when two objects in the animation collide?** This can be answered in several different ways, depending on the environment. Maybe one object should blow up or disappear; maybe they should bounce off in different directions; or maybe one gets bigger.
2. **Why is testing for a collision using the bounding rectangle deceiving?** Testing to see if the bounding rectangles overlap can be deceiving because there might be extra space around the object that is not part of the sprite but is included in the rectangle. The result is that the bounding rectangles collide, but not the actual image.
3. **Why is an understanding of physics important to animation and game design?** Understanding physics helps make the animation more realistic.

Student Activity:

Directions to the student:

Read the following scenario and complete the hands-on activities. Note that the screenshots in the activity may appear differently on your system. Answer the questions as you work through the steps. Verify your answers with the instructor. Request assistance from the instructor as needed.

Scenario:

Working as a student intern at Tailspin Toys gets more and more exciting every day. The supervisor thinks you are ready for a new challenge that is the very essence of what you love about video games—collisions!

And just as in the previous task, you know that a quick refresher of your skills will pay off in the long run.

Content:

This animation tutorial provides directions for adding images to a 2-D environment, with initial positions, textures, and speeds. When the images hit the edges of the screen, they reverse their direction. When they bump into each other, they change direction.

1. Open Visual Studio with the XNA software development kit (SDK).
2. Create a project named BouncingBalls.
3. Add the three images (OrangeBall.png, PurpleBall.png, and TealBall.png) to the project.

4. Add global variables for the following items:
 - a. Texture2D objects for each ball.
 - b. A bounding rectangle for each image
 - c. The speed for each ball's x and y starting velocity
5. In the load content section:
 - a. Add the texture image for each ball to the texture variable.
 - b. Define the rectangle for each object (you can choose random starting points).
6. In the *Update* method:
 - a. Test to make sure each object is within the bounds of the screen, if not, reverse the direction.
7. Call a "check for collision" method that checks for any collision of objects and then reverses their directions.
8. Add a draw method to draw each ball on the screen.
9. Test your program. Watch for balls that go beyond the edges of the screen. Verify that collided objects change direction.
10. Make adjustments as necessary.
11. Do the balls always follow the same pattern? How can you change that?
12. Experiment with the speed of collided objects. Increase it when they first collide and reduce it after the second collision.

Answer:

The code solution is located in 98-374-ENU-4.3-IC_Key.zip.