CSCI 1106
Lecture 15

Movement and Collision Detection

Announcements

• Today’s Topics
  – A brief reminder of the Movie Metaphor
  – Autonomous object movement
  – Movement beyond the stage
  – Collision detection
Reminders

- Drop deadline is March 8
- You need to pass the individual component (quizzes and final) to pass the course
- If your quiz average is less than 50%, please come and see me

The Movie Metaphor

- Key Idea: Stage is updated 60 times per second
  - ENTER_FRAME event
  - Redraw all objects on the stage
- Key Idea: On the ENTER_FRAME event
  - Update the positions and properties of all objects
  - Add/remove objects as needed
  - Update graphics as needed
- Idea: Change in an object’s position from frame to frame looks like object motion
Autonomous Motion

- Listen for the NEXT_FRAME event
- Set the object’s velocity \( (vx, vy) \)
  - \( vx \) is how many pixels per frame the object moves horizontally
  - \( vy \) is how many pixels per frame the object moves vertically
  - *Either can be positive or negative*
- On each frame increment the \((x,y)\) position of the object by its velocity
  - \( object.x = object.x + object.vx; \)
  - \( object.y = object.y + object.vy; \)
Issues with Motion

• Where should we set the object’s velocity?
• What does it mean if the velocity is negative?
• What happens if the velocity is too great?
• Must the velocity be constant?
• What happens if we hit the wall?

Hitting the Wall

• Fact: If the object keeps moving it will reach the edge of the stage
• Two options:
  – Fall off the edge
  – Bounce back
• How do we know when we have hit the wall?
• Does it matter which wall it is?
Falling of the Edge

- Idea: Once object is no longer visible, remove it
- How do we know when an object is no longer visible?
  - Bottom edge of object is at the top wall:
    \( \text{object.y} < -\text{object.height} \)
  - Top edge of object is at the bottom wall:
    \( \text{object.y} > \text{stage.stageHeight} \)
  - Right edge of object is at the left wall:
    \( \text{object.x} < -\text{object.width} \)
  - Left edge of object is at the right wall:
    \( \text{object.x} > \text{stage.stageHeight} \)
- Where do we perform the test?
- If the test is positive: remove the object

Bouncing of the Wall

- Idea: Once object touches a wall, reverse velocity
- How do we know which velocity to reverse (\( \text{vx} \) or \( \text{vy} \))?  
- Reverse \( \text{vy} \) if
  - Top edge of object is at the top wall
    \( \text{object.y} \leq 0 \)
  - Bottom edge of object is at the bottom wall
    \( \text{object.y} + \text{object.height} \geq \text{stage.stageHeight} \)
- Reverse \( \text{vx} \) if
  - Left edge of object is at the left wall:
    \( \text{object.x} \leq 0 \)
  - Right edge of object is at the right wall:
    \( \text{object.x} + \text{object.width} \geq \text{stage.stageWidth} \)
- How do we reverse velocity?
- Where do we perform the test?
Collision Detection

- Obs: Previous slides described a special form of collision detection
- In general, collision detection is needed to detect if two or more objects are intersecting or touching in some way
- Why is this useful?

Mechanisms for Collision Detection

- Three ways to detect collisions:
  - Cheap and fast: Check if bounding boxes overlap
    - Use hitTestObject()
  - Expensive and slow: Check if the points of one object intersect with the other
    - Use hitTestPoint()
  - More complicated and fast: Use invisible objects
- For most purposes, the first way suffices
Bounding Boxes

- **Defn:** A *bounding box* of an object is the smallest orthogonal rectangle that can contain the object.

- **Idea:** If the bounding boxes of two objects intersect, a collision has occurred.
- **Pros:** Fast, cheap, simple to use.
- **Cons:**
  - Cannot determine where the collision occurred.
  - Irregularly shaped objects have large bounding boxes.
  - False positives.
- **Use:**
  ```java
  if(objA.hitTestObject(objB))
  { ...
  }
  ```
- **Obs:** Need finer granularity mechanism.
hitTestPoint()

• Ideas:
  – Detect whether a specific point is within the shape of the object
  – Useful on vector objects (ones you draw with the rectangle tool)
  – Only the drawn part is checked for overlap with the point
  – The bounding box isn’t considered!
• Pros: Still pretty simple
• Cons:
  – Can only check one point
  – Objects comprise many points so object collisions require multiple checks
    • E.g. ball and paddle
    • Expensive and slow if many points need to be checked
  – Does not work for bitmapped graphics
• Use:
  if(obj.hitTestPoint(x,y))
  {
    ... collision
  }

Vector vs Bitmapped Graphics

• **Vector** based graphics are those that you draw using the rectangle, circle, or other tools
• **Bitmap** based graphics are pictures that you import
• Problem: Flash uses the bounding box for point hit detection on bitmapped graphics
  if(obj.hitTestPoint(x,y))
  {
    ... collision
  }
A Compound Approach

- **Problem:**
  - Need to use `hitTestObject()` on irregular shaped object
  - Bounding box of object differs from object shape

- **Solution:**
  - Create invisible objects within this object with smaller bounding boxes
  - Use the smaller bounding boxes to detect collisions