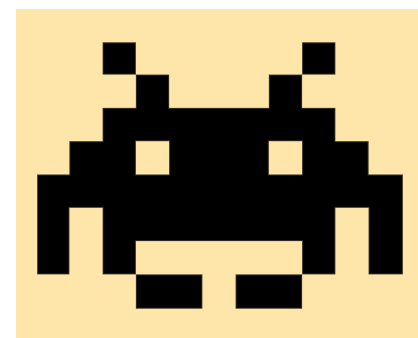


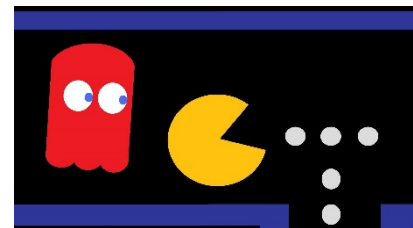


CSCI 1106

Lecture 4



Movement and Collision Detection



Today's Topics

- A brief reminder of the Movie Metaphor
- Autonomous sprite movement
- Movement beyond the stage
- Collision detection
- Variables

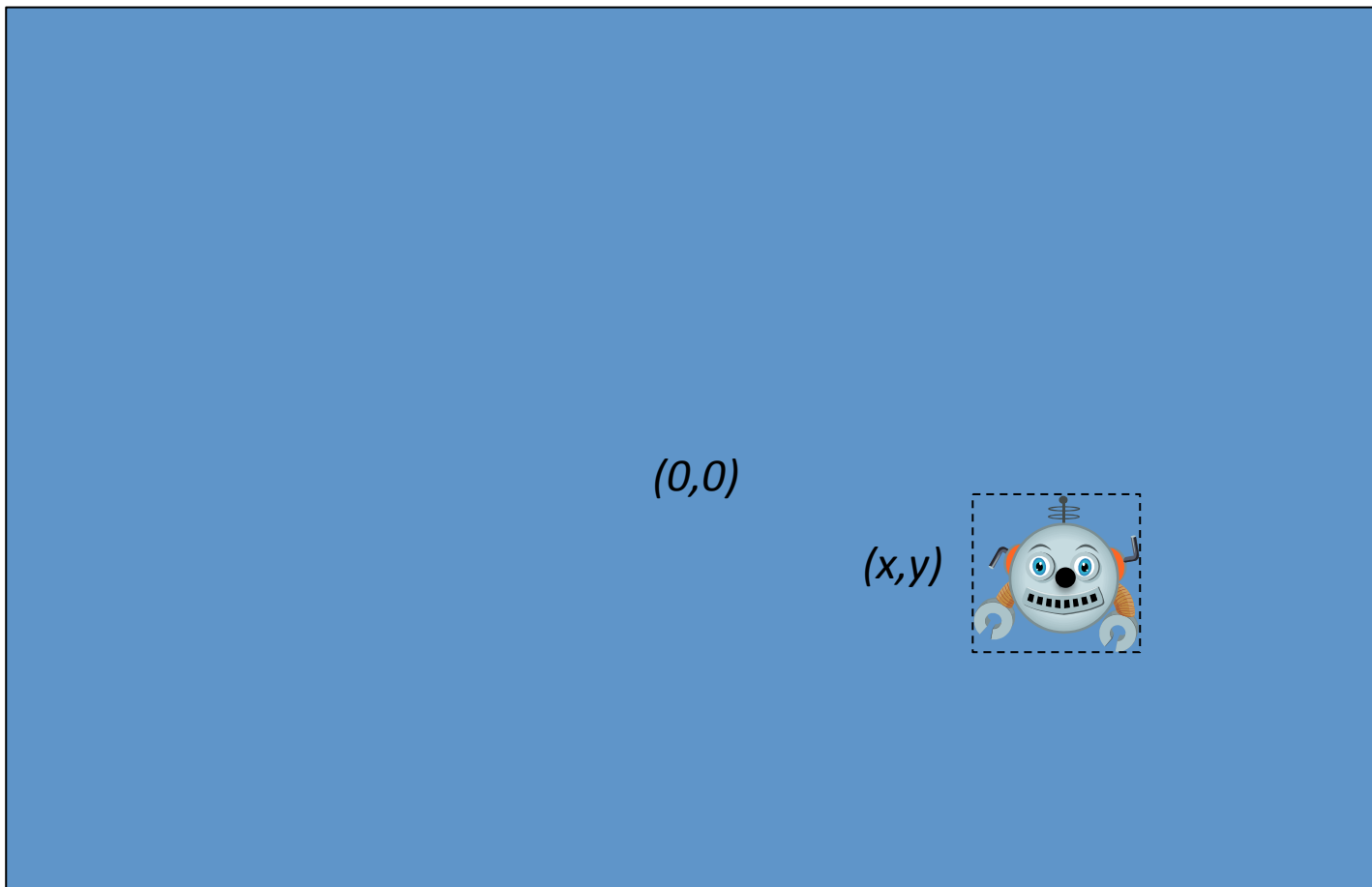
The Movie Metaphor

- Key Idea: Stage is updated 30 times per second
 - Broadcasts a FRAME event
 - All sprites are redrawn on the stage
- On the FRAME event, the sprites
 - Update their positions and properties
 - Add/remove sprites as needed
 - Update costumes as needed
- Idea: Change in a sprite's position from frame to frame looks like motion

The Setup

$(-240, 180)$



$(240, 180)$

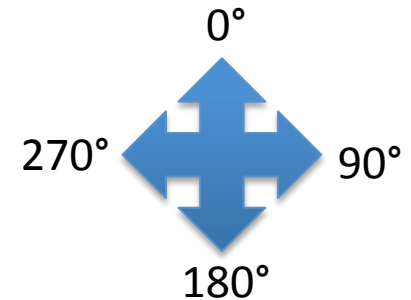


$(-240, -180)$

$(240, -180)$

Autonomous Motion

- Set the sprite's velocity
 - Number of steps (pixels) per frame
 - *Can be positive or negative*
- Set the sprite's direction property 
- Create a script to respond to the FRAME event
- On each frame change the position of the sprite by constant steps 
 - e.g. move 10 steps per frame at 90°




Issues with Motion

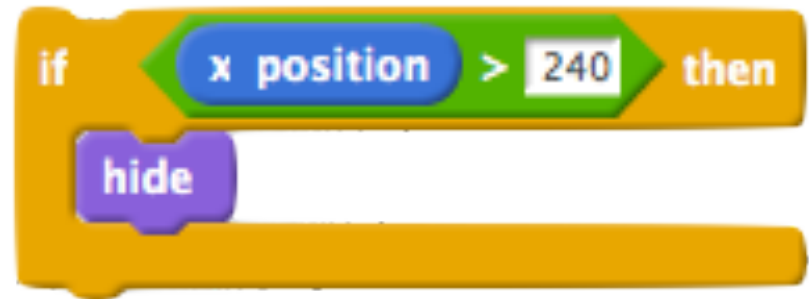
- Where should we set the sprite's velocity?
- What does it mean if the velocity is negative?
- What happens if the velocity is too great?
- Must the velocity be constant?
- What does it mean if the direction is negative?
- What happens if we hit the wall?

Hitting the Wall

- Fact: If the sprite 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 sprite is no longer on stage, hide it 
- How do we know when a sprite is no longer on stage?
 - Sprite is at the top wall:
`y position > 180`
 - Sprite is at the bottom wall:
`y position < -180`
 - Sprite is at the left wall:
`x position < -240`
 - Sprite is at the right wall:
`x position > 240`
- Where do we perform the test?
- If the test is positive: remove or hide the sprite
- Is there an easier way?



Falling Off when Touching the Edge

- Idea: If the sprite is touching an edge, hide it



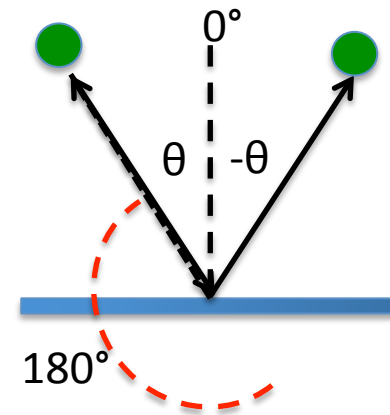
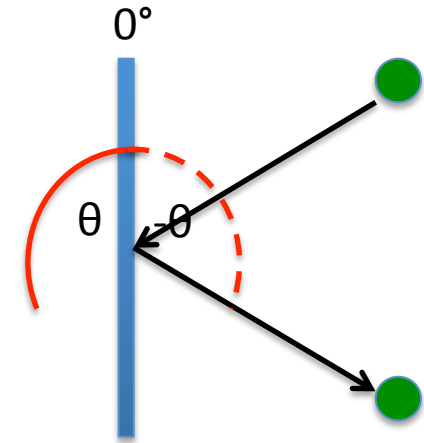
Bouncing of the Wall

- Idea: Once a sprite touches a wall, reverse velocity
- How do we know the new direction?
- Two scenarios
 - Vertical wall

point in direction **0** - direction

- Horizontal wall

point in direction **180** - direction



An Easier Bounce of the Wall

if on edge, bounce

Collision Detection

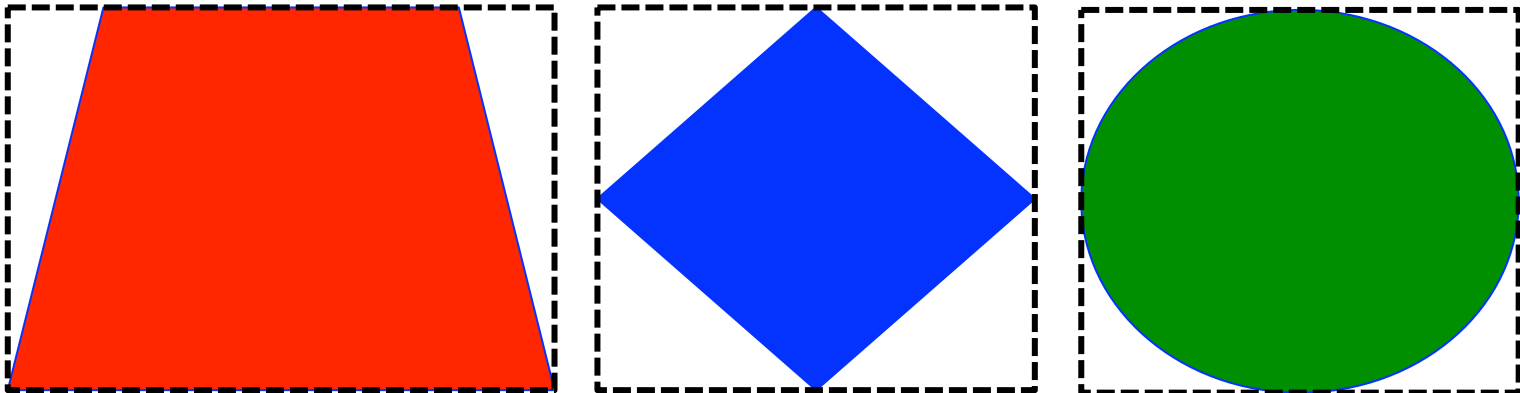
- Obs: We just described a special form of *collision detection*
- In general, *collision detection* is needed to detect if two or more sprites are intersecting or touching in some way
- Why is this useful?

Mechanisms for Collision Detection

- Four ways to detect collisions:
 - Cheap and fast: Check if bounding boxes overlap
 - Expensive and slow: Check if the points of one sprite intersect with the other
 - Fast but specialized: Use geometry
 - More complicated and fast: Use invisible sprites
- For most purposes, the second way suffices

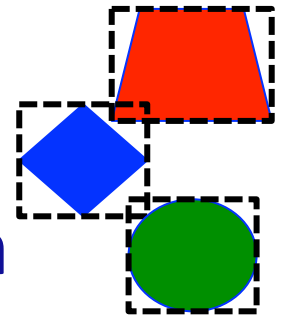
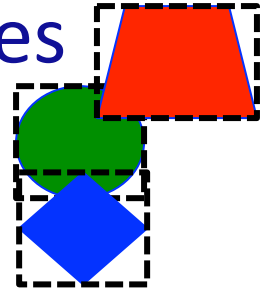
Bounding Boxes

- Defn: A *bounding box* of a sprite is the smallest orthogonal rectangle that can contain the sprite



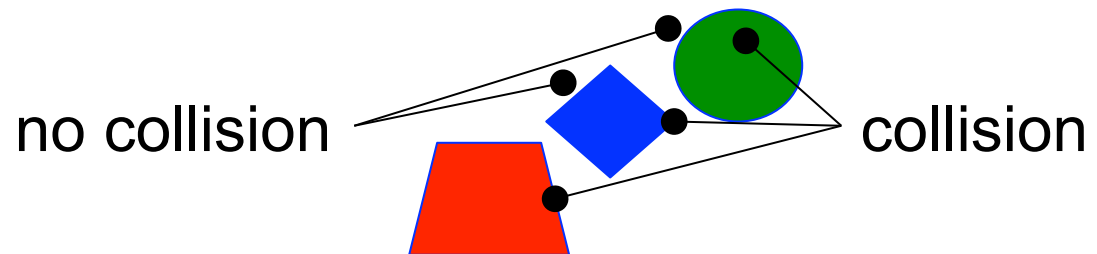
Bounding Box Collision Detection

- Idea: If the bounding boxes of two sprites intersect, a collision has occurred
- Pros: Fast, cheap, simple to use
- Cons:
 - Cannot determine where the collision occurred
 - Irregularly shaped sprites have large bounding boxes
 - False positives
- Obs: Need finer granularity mechanism



Point Based Collision Detection

- Ideas:
 - Detect whether a specific point is within the shape of the sprite
 - Only the drawn part is checked for overlap with the point
 - The bounding box isn't considered!
- Pros: More accurate than bounding box
- Cons: Sprites comprise many points so collisions require multiple checks



A Compound Approach

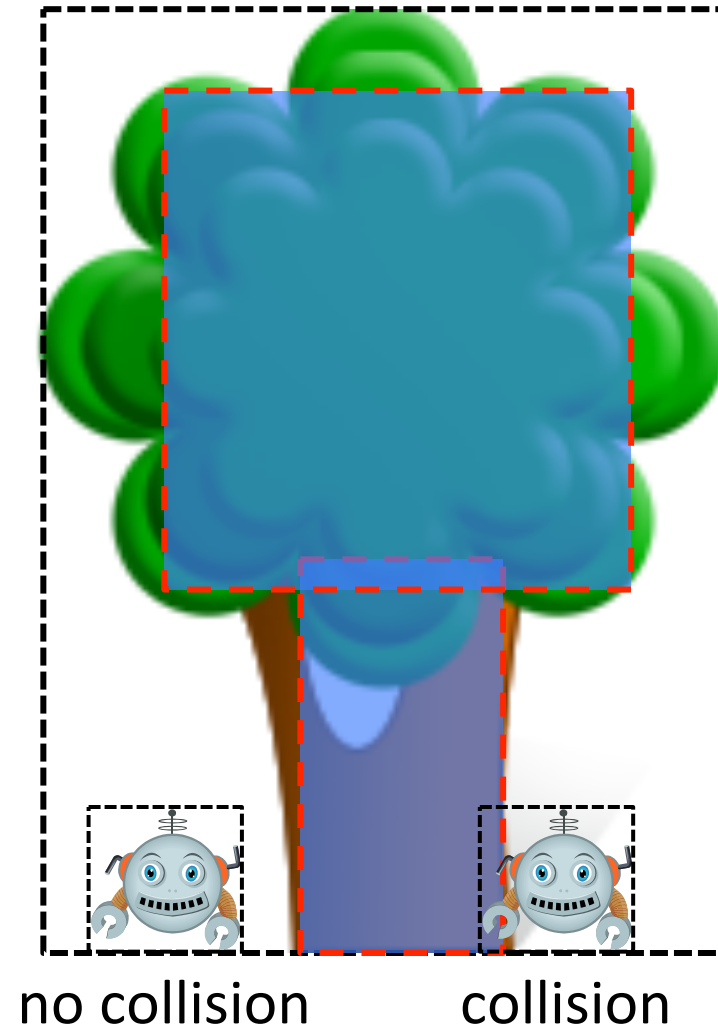
- Obs 1:
 - Bounding boxes are fast but inaccurate
 - Point-wise detection is accurate but slow
- Obs 2: Collisions are rare compared to FRAME events
- Idea: Use a two-step process
 - Check if bounding boxes overlap
 - If yes, perform point-wise collision detection
 - If no, then no collision has occurred

Geometry

- Using geometry for finding collision
- i.e. Circle Collision
 - taking the centre points of the two circles
 - ensuring the distance between the centre points are less than the two radius added together.

Use invisible sprites

- Problem:
 - Want to use bounding box collision detection on irregular shaped sprite
 - Bounding box of sprite differs from its shape
- Solution:
 - Create invisible sprites within this sprite with smaller bounding boxes
 - Use the smaller bounding boxes to detect collisions



Variables

- Idea: A *variable* is a changeable value recorded in Scratch's memory.
- If a property or value in your program will change during the execution of your program, you will likely need a variable to keep track of it.
- Local (or private/personal)
 - This sprite only
- Global (or public)
 - All sprites or stage
- Cloud
 - Stores on server.
 - Allows for data from a project to be saved and shared online.
- List (array)
 - Is made of items like a variable
 - Can be useful when many variables are needed
 - Local or Global

Tomorrow's Tutorial

- Extra Features
 - Keeping Score
 - Playing Sounds
 - Fixing Bugs