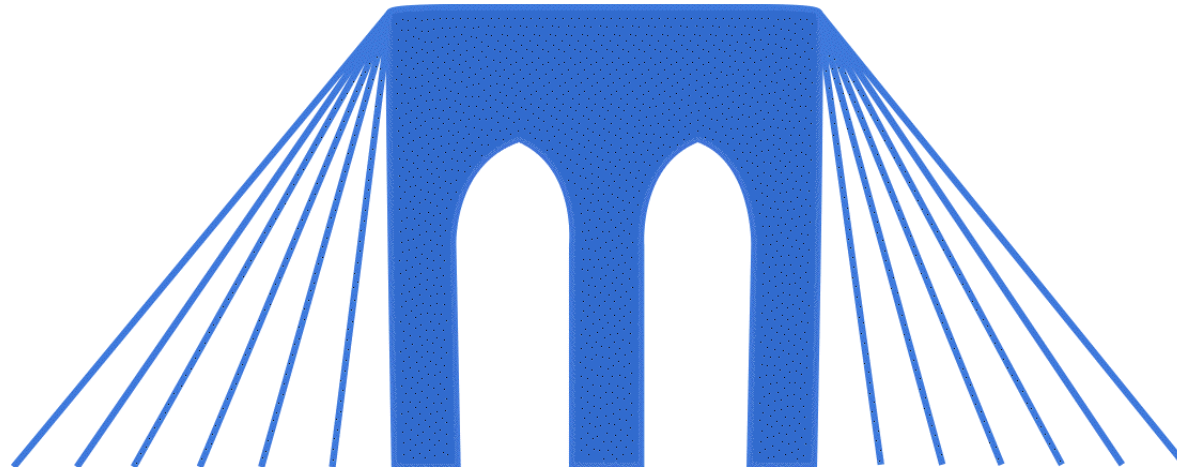


BRIDGES TO COMPUTING



General Information:

- This document was created for use in the "Bridges to Computing" project of Brooklyn College.
- You are invited and encouraged to use this presentation to promote computer science education in the U.S. and around the world.
- For more information about the Bridges Program, please visit our website at: <http://bridges.brooklyn.cuny.edu/>

Disclaimers:

- All images in this presentation were created by our Bridges to Computing staff or were found online through open access media sites and are used under the Creative Commons Attribution-Share Alike 3.0 License.
- If you believe an image in this presentation is in fact copyrighted material, never intended for creative commons use, please contact us at <http://bridges.brooklyn.cuny.edu/> so that we can remove it from this presentation.
- This document may include links to sites and documents outside of the "Bridges to Computing" domain. The Bridges Program cannot be held responsible for the content of 3rd party sources and sites.



Introduction to Game Programming & Design

Lecture 3: Game State and Game Mathematics

Content

1. Player Expectations

1. Efficiency

2. Game Mathematics

1. Collision Detection & Response
2. Complexity

3. Game State

1. Game state
2. Agent state

Player Expectations

- In general, games are held to a higher standard than other types programs.
- People expect "office applications" to fail, and don't expect 100% up-time from business websites.
- What we are willing to tolerate when "working" is wildly different then what we are willing to tolerate when "playing".

Efficiency

- Complexity and computability are concepts that are not normally taught on an undergraduate level.
- BUT game programmers need to consider "efficiency" in everything that they do.
- Studies have shown that if a player has to wait more than 30 seconds for levels to load in a game, their "review" of that will be greatly reduced.

Game Mathematics

- "Game Mathematics" refers both to areas of general mathematics (geometry, trigonometry, calculus) as well as specialized areas of mathematics (vectors, matrices).
- Graphic libraries, game libraries, 2D and 3D libraries exist for programming languages to help simplify the mathematical problems that you will face. But they can't be relied on to do everything.

Collision Detection

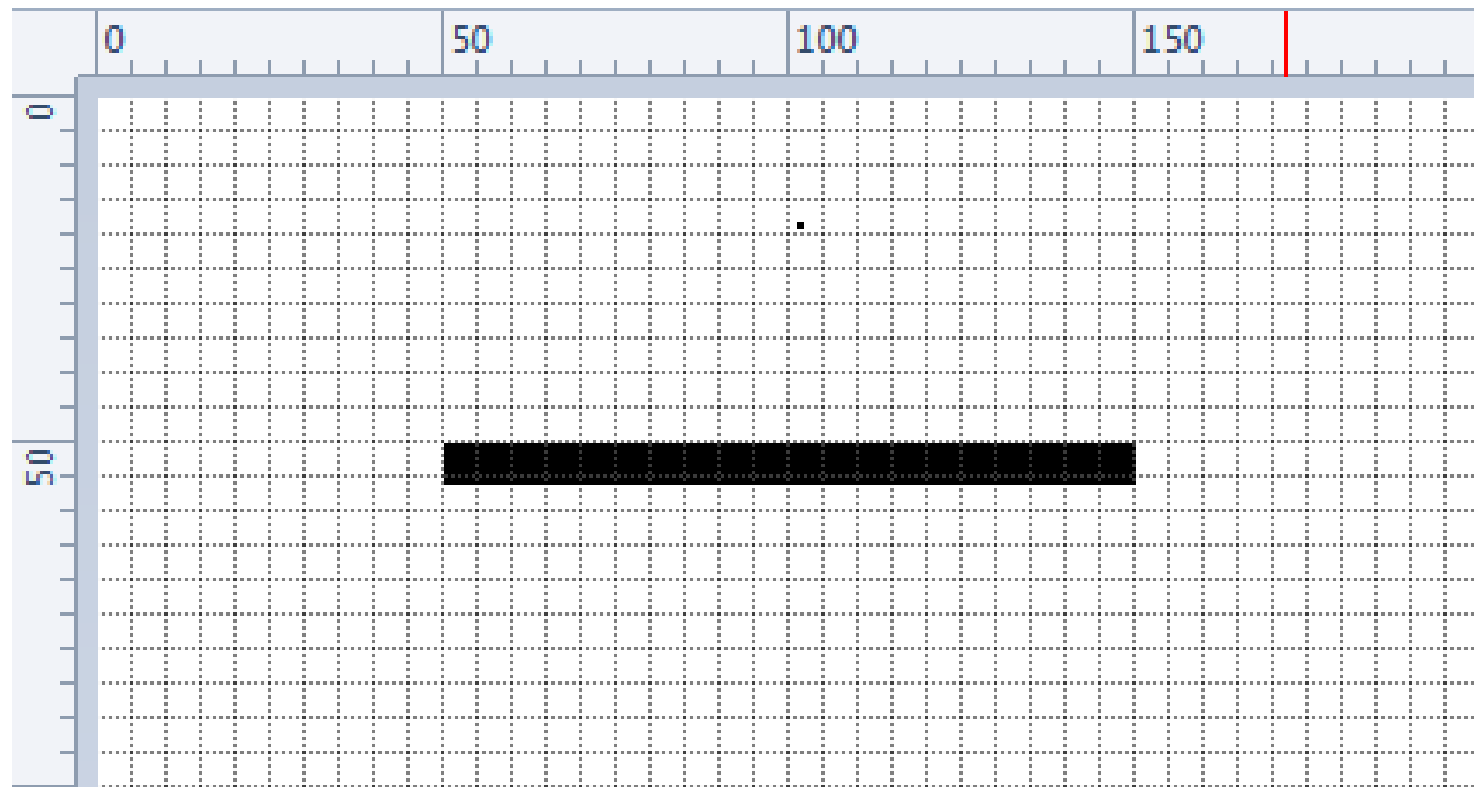
- Figuring out if two objects are touching incredibly common problem in a game:
 - Ball games (pong)
 - Shooting games.
- Two basic techniques:
 - Overlap testing
 - Detects whether a collision has already occurred
 - Intersection testing
 - Predicts whether a collision will occur in the future

Overlap Testing

- Facts:
 - Most common technique used in games
 - Exhibits more error than intersection testing
- Concept
 - For every simulation step,
 1. Move (update) all objects.
 2. Test objects to see if they overlap.
 3. If objects overlap, make adjustments or corrections.
 - Easy for simple volumes like dots, boxes and spheres, but harder for polygonal models.

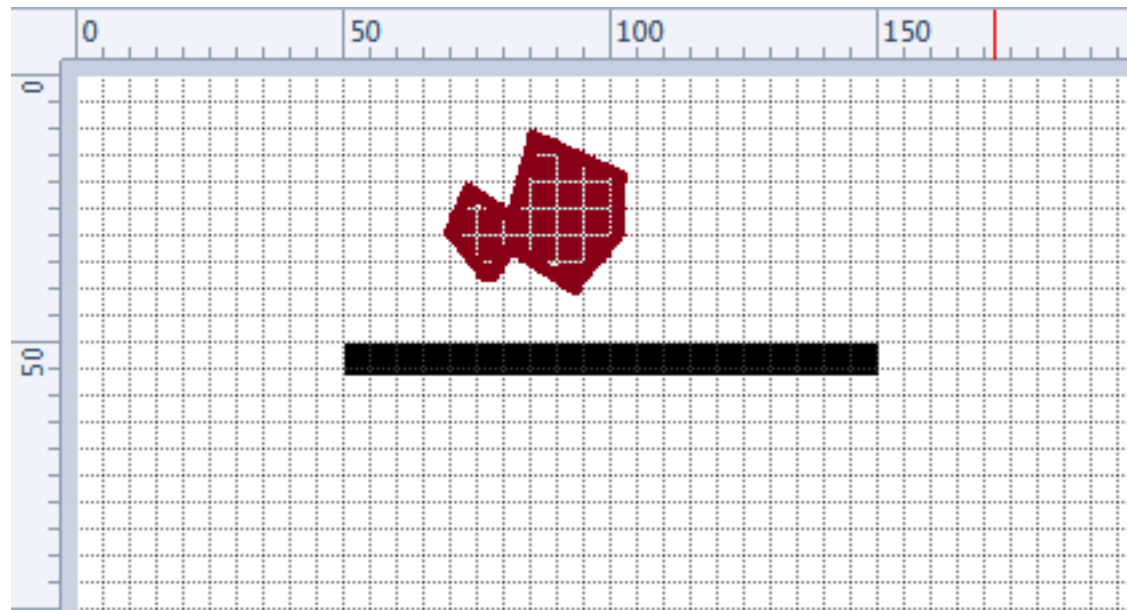
Simple Overlap Testing

- Easiest example is of a particle interacting with a square.
- This will still require 4 logical tests in a 2D game.
- Depending on the type of game that played, the order of those 4 tests can have a profound effect on efficiency.



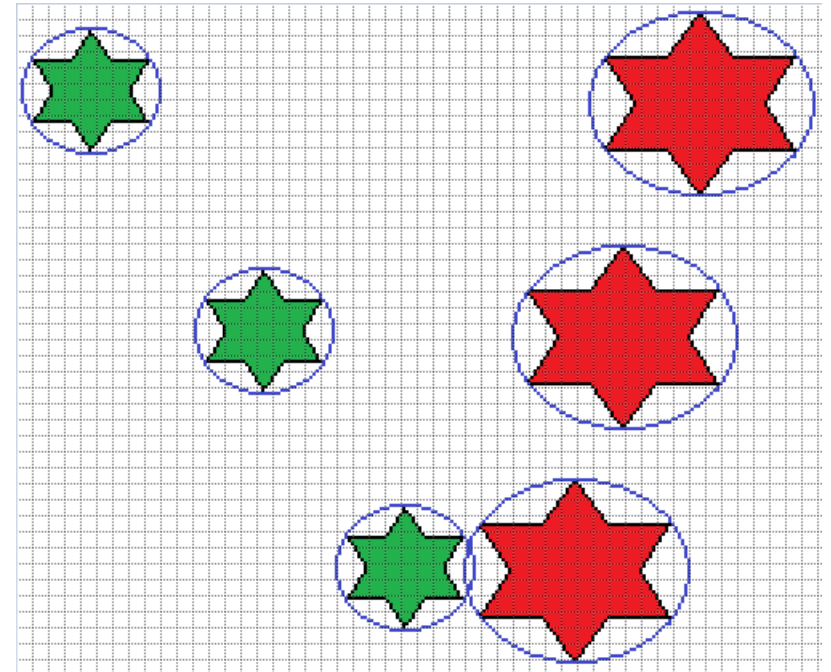
Complex Shape OOT

- How many tests would be required now?



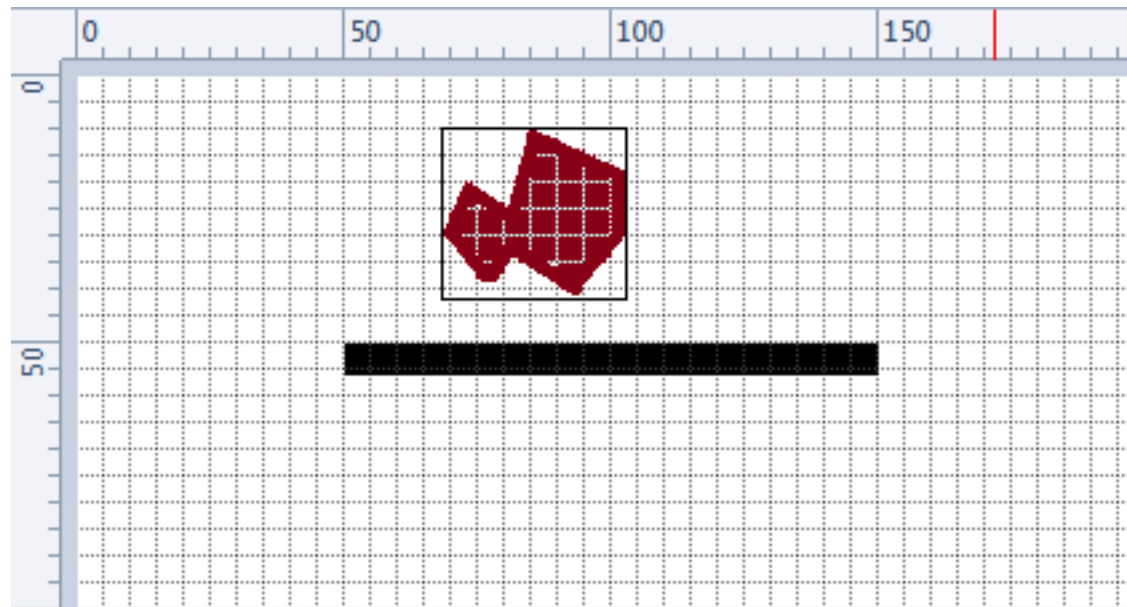
Bounding Circles

- For some complex shapes we can use bounding circles.
- To test whether or not two objects are colliding, just compare the distance between centers to the sum of the radii of the two circles.



Bounding Boxes

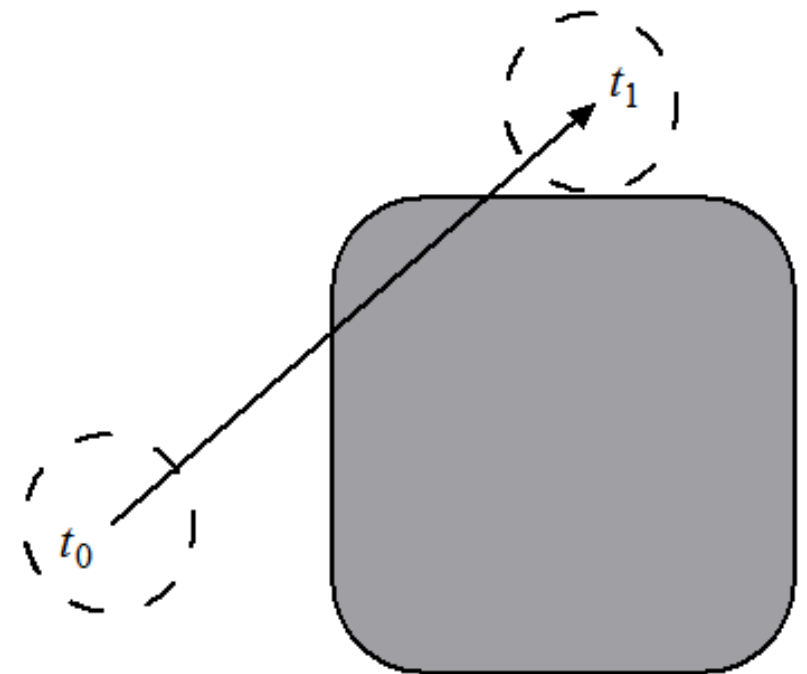
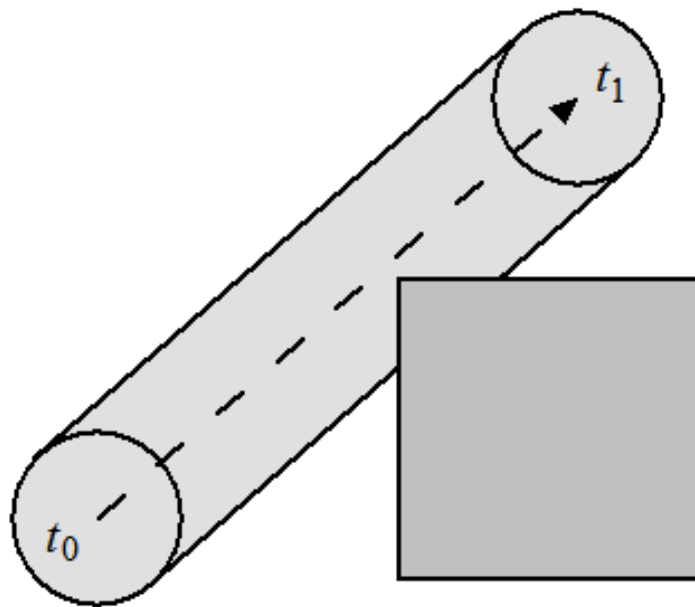
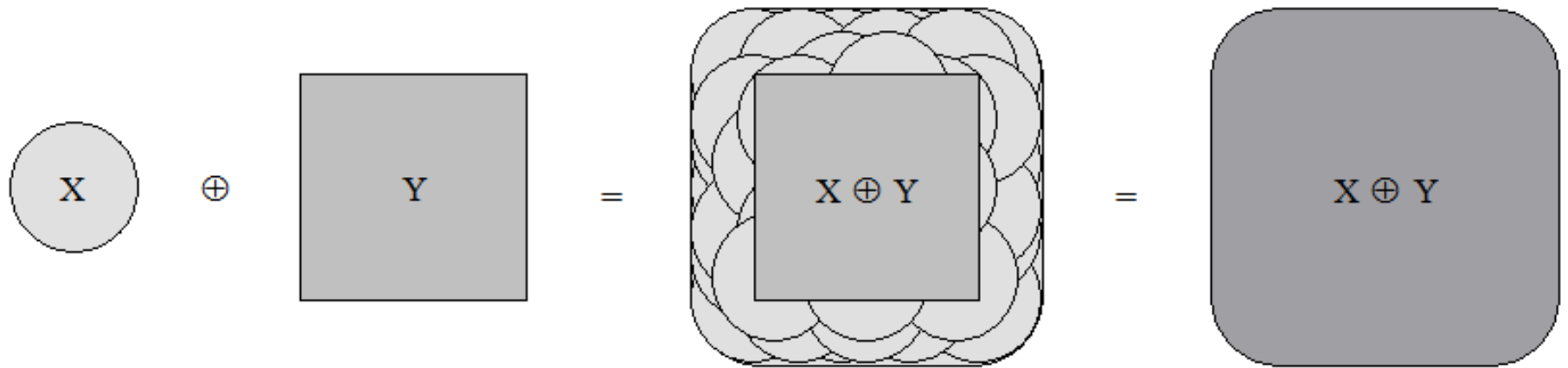
- Bounding Boxes can be used to reduce the complexity of shapes to simplify overlap testing.
- Note that secondary testing may need to be done if the bounding box is found to overlap.
- How many test now?



Minkowski Sum

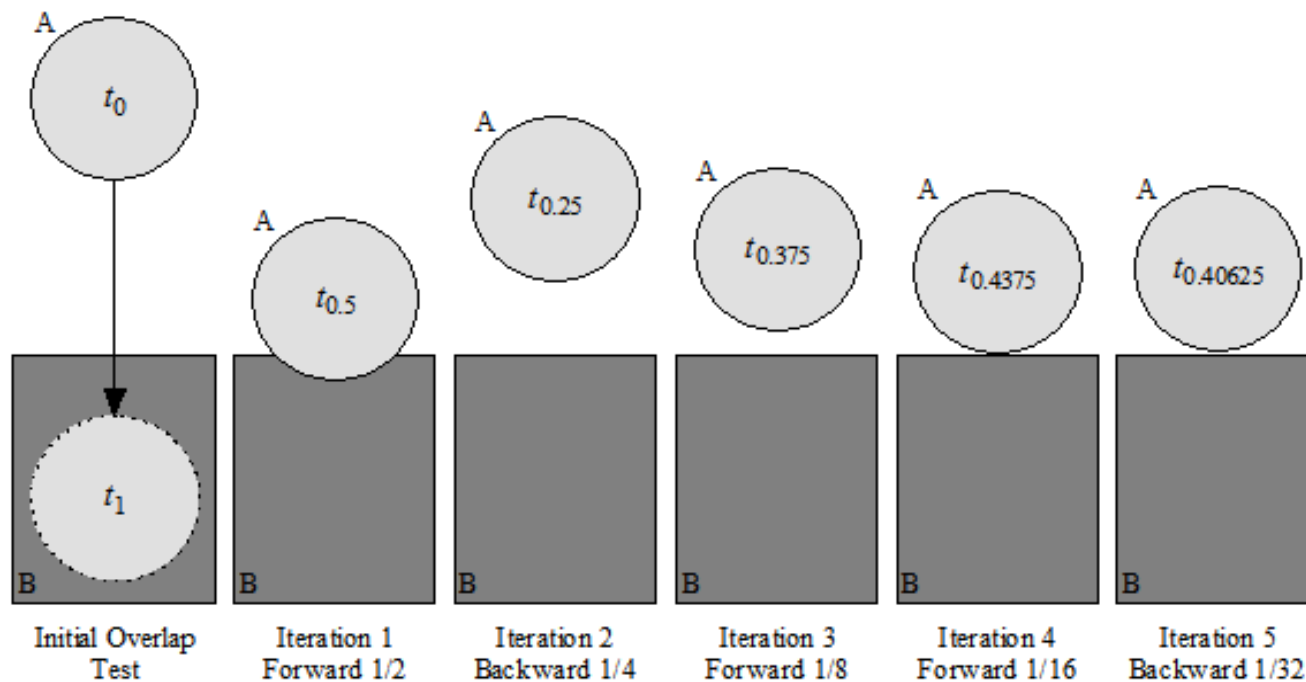
- Simple technique for reducing number of tests necessary.
- By taking the Minkowski Sum of two complex volumes and creating a new volume, overlap can be found by testing if a single point is within the new volume.
- Variations on the Minkowski Sum include calculating the x, y and z distances between the two objects that are being tested.

Minkowski Sum



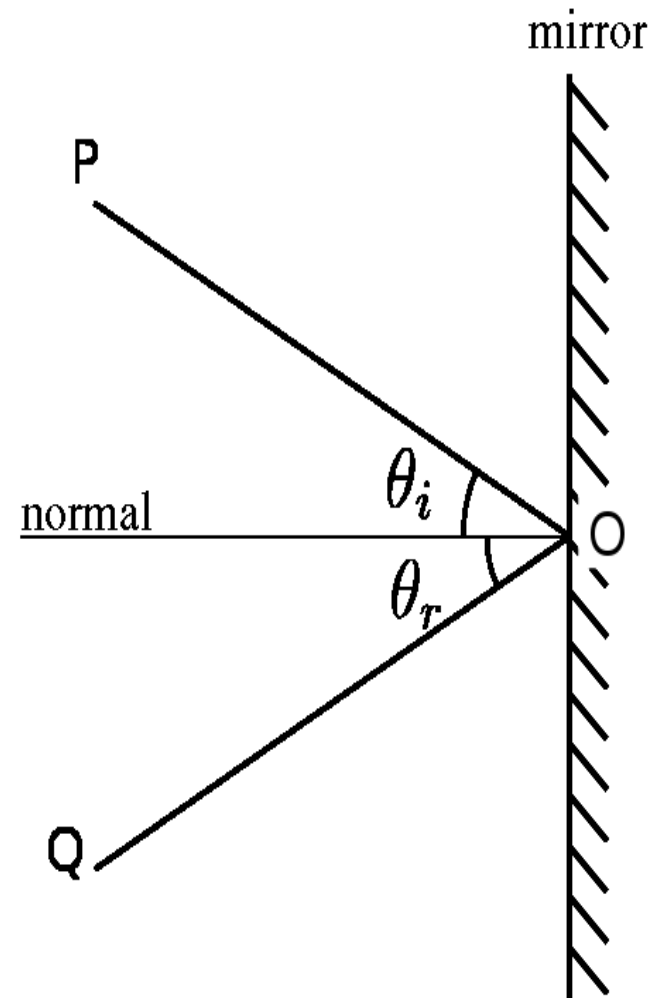
OT - Collision Time

- Collision time calculated by moving object back in time until right before collision.
 - Bisection is one effective technique. $\Theta(\log n)$
 - Minkowski values are another.



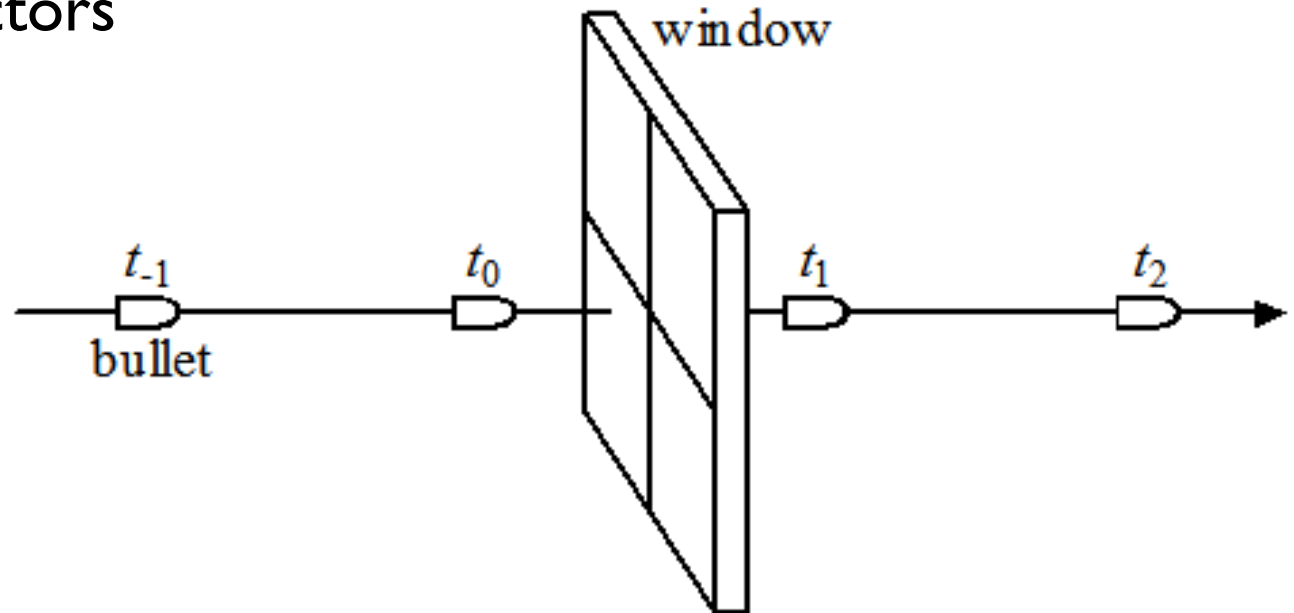
Collision Response

- Having captured the exact moment and position of collision, geometry, and trigonometry can be applied to calculate new trajectories.



Limits of OT

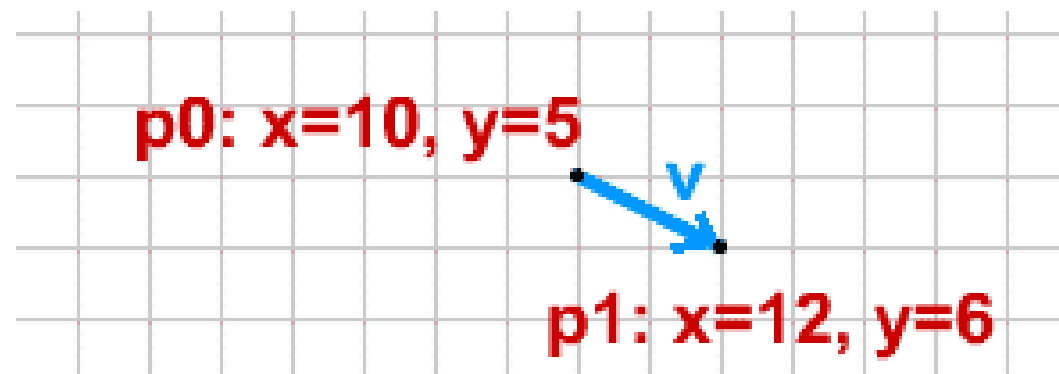
- OT is easy, but limited.
 - Fails with objects that move too fast
 - Unlikely to catch time slice during overlap
- Possible solutions
 - Design constraint on speed of objects
 - Reduce simulation step size
 - Use Vectors



Vectors

- You already know of "vector images", images represented by a mathematical formula.
- We can represent entire objects (and their movement) with formula's as well.
- Vector (an matrice) mathematics can then be applied to reveal information about where objects will be and whether or not they will collide (at any time, past or future).

`PI [x1, y1, x2, y2];` // A particle vector.



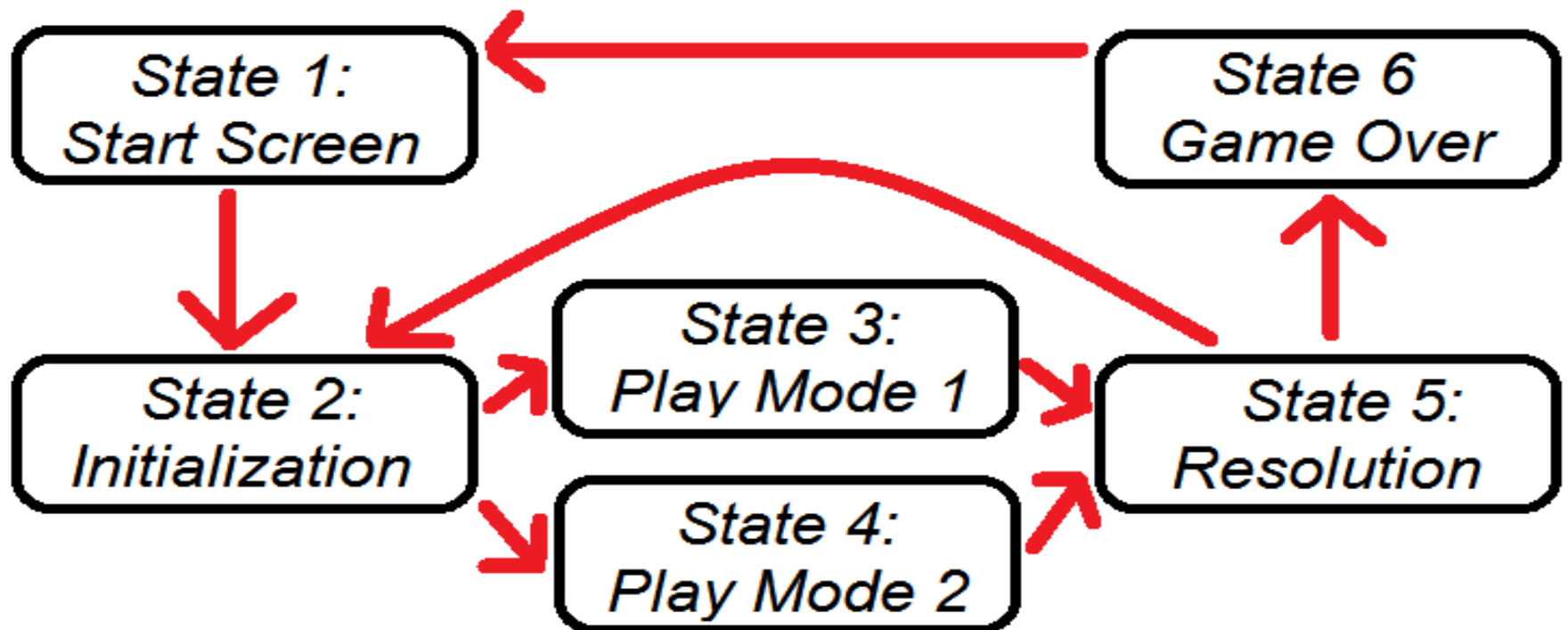
Game Mathematics In SCRATCH

- SCRATCH simplifies game mathematics for you, with a couple of hand blocks.
 - Object collision detection in SCRATCH can be done with a simple "is touching block".
 - Objects can be kept on screen with a simple "if on edge bounce" block.



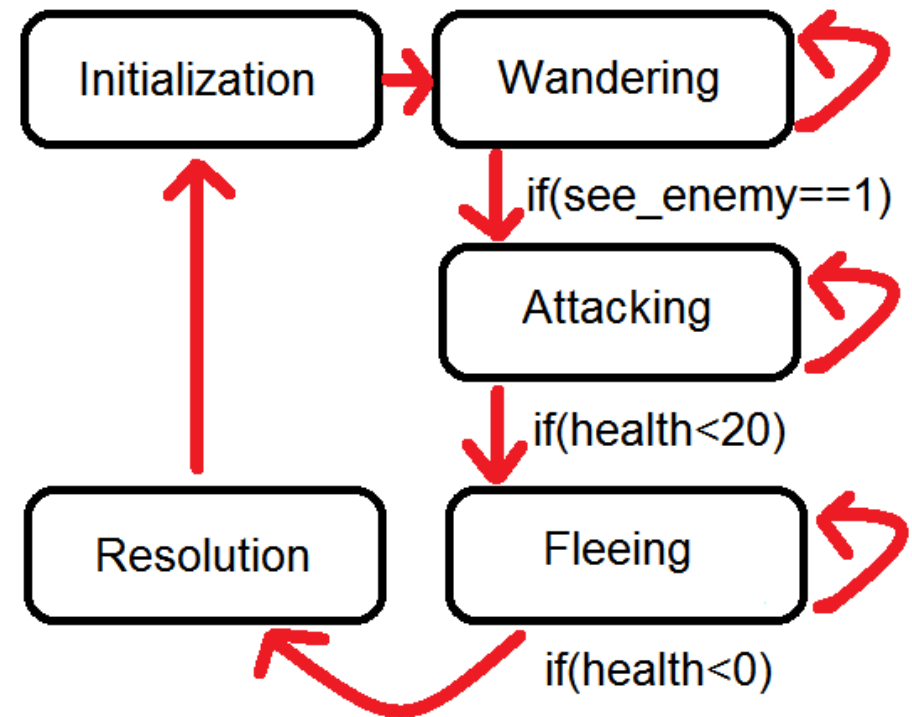
Game State

- All games consist of a sequence of states.
- Each state is characterized by a combination of visual, audio and/or animation effects, as well as a set of rules that are being applied.



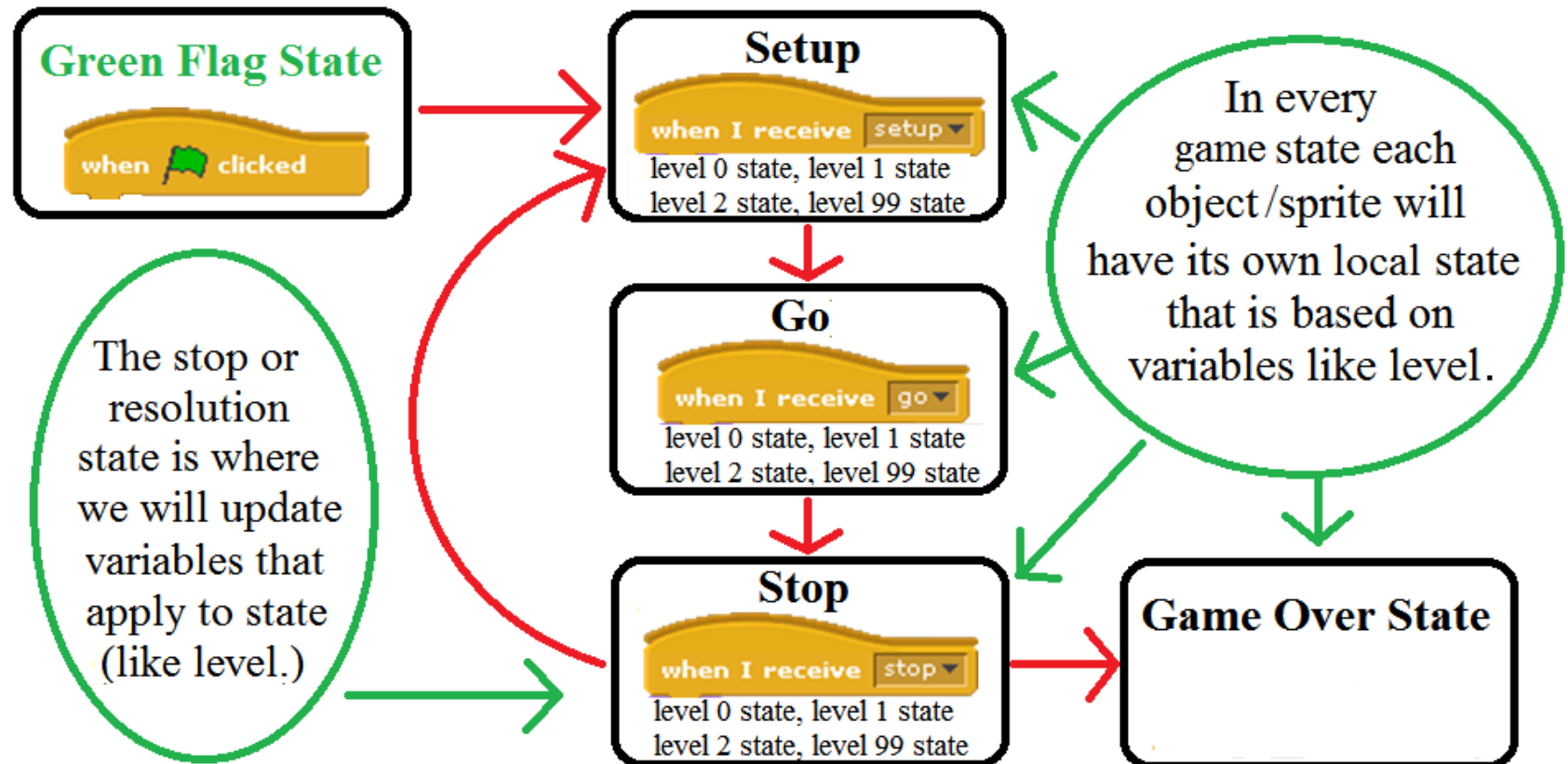
Object State

- Objects in the game proceed through their own states as well.
- These states are defined by the behavior and functionality applied at that time.



Game State in Scratch

- At typical game in Scratch might use the following state transition diagram.



Object State in Scratch

- In a typical game in Scratch, all of your normal sprites (not stage or any control sprites like buttons) will work very well with only 4 scripts.
- These scripts (and any associated variables) will control the objects state.

```
when clicked
hide
stop script
```

```
when I receive setup
point in direction 90
go to x: -120 y: 0
set size to 40 %
switch to costume costume1
if level = 0
hide
stop script
```

```
when I receive go
if level = 0
stop script
```

```
when I receive stop
hide
stop script
```



The End