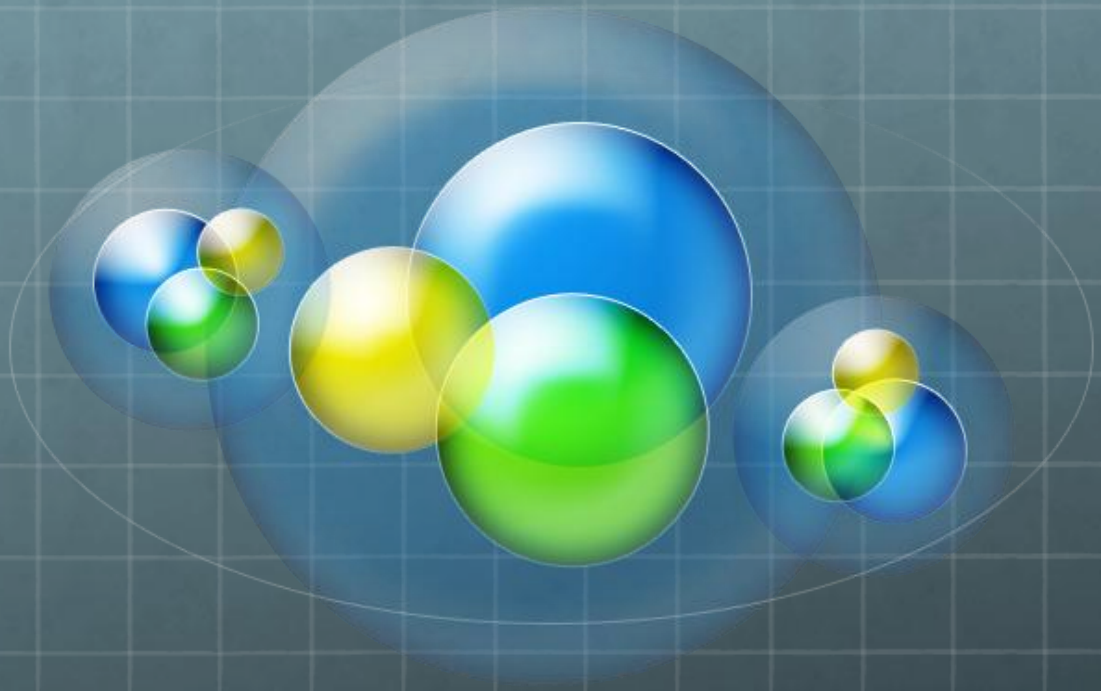


Collision Simulation

Eric Leong

Computer Programming

- "Software Development"
  - Process of creating *programs*
- Use programming *languages*
  - Java
  - C++
- Goal
  - Program accomplishes task

Programming Languages

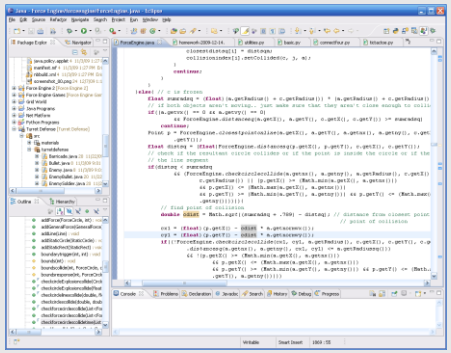
- Language human and computer can understand
- Tell the computer what calculations to do
- Algorithm: a set of instructions that solve a problem

Software Development

Idea

Software Development


```
def primeFactors(num):
    i, l = 2, []
    while i <= math.sqrt(num):
        while num % i == 0
            num = num // i
            l.append(i)
        i += 1 if i % 2 == 0 else 2
    return l + ([] if num == 1 else [num])
```



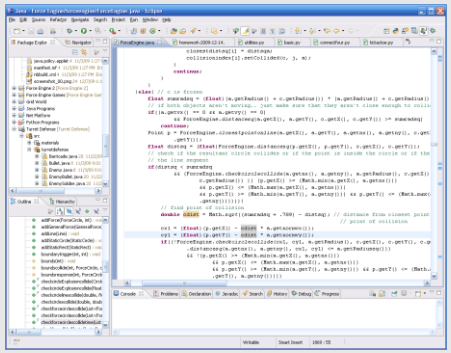
Software Development

Idea

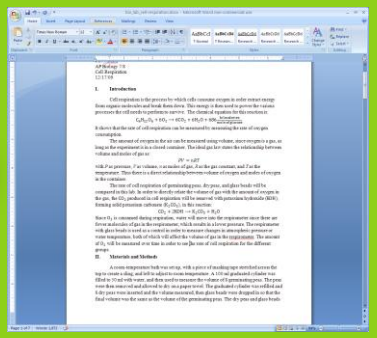
Programmer writes

Code

```
def primeFactors(num):
    i, l = 2, []
    while i <= math.sqrt(num):
        while num % i == 0
            num = num // i
            l.append(i)
        i += 1 if i % 2 == 0 else 2
    return l + ([] if num == 1 else [num])
```

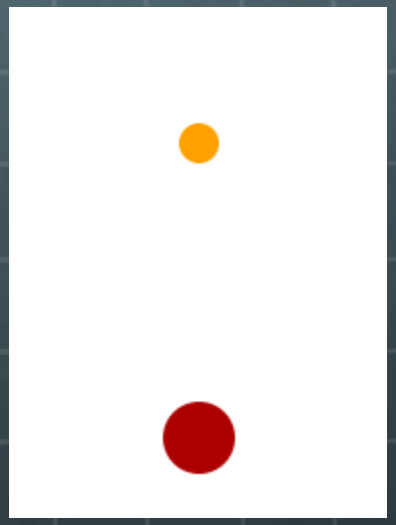


Run by computer



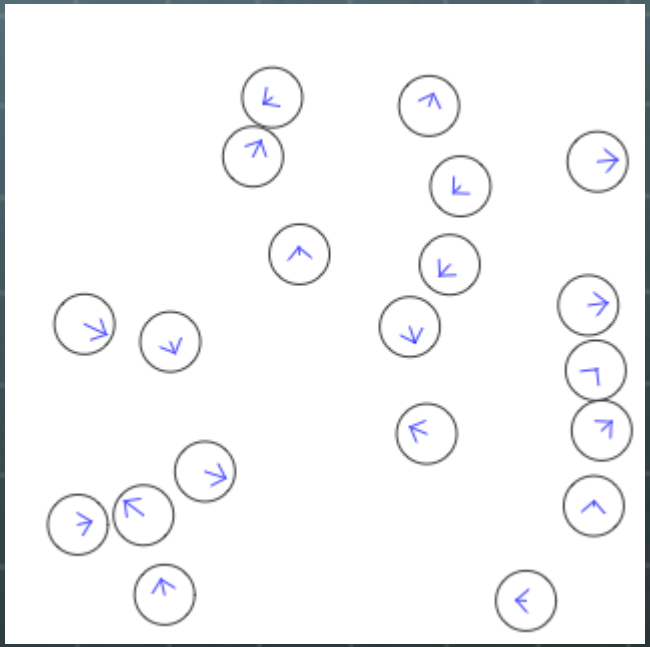
Computer Simulations

- Simulations are models
  - Realistic
  - Predictable (consistent)
- Objects represent physical things
- Interactions between objects
  - Gravity
  - Collisions



Simulation Specifics

- Objects move around by translocation
- Each object has a vector
  - Magnitude
  - Velocity
- Compare objects for interactions
  - Done in pairs

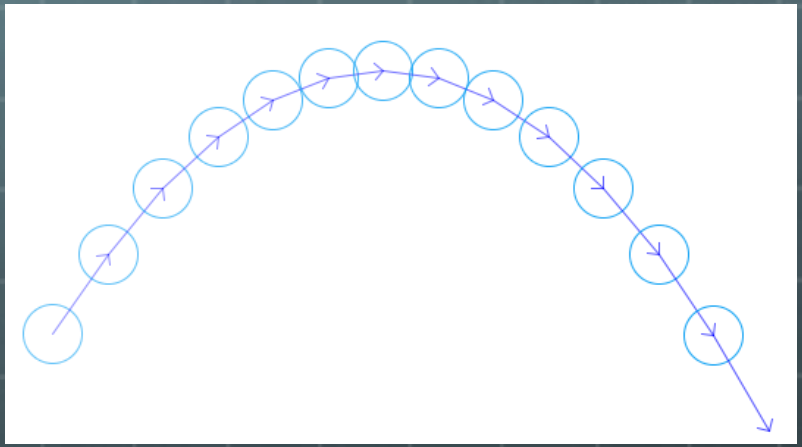


Real-time

- Time to compute <u>not</u> noticeable
- Appears smooth
  - No stuttering
- Instant reaction to user interaction
  - No “lag”

Real-time Modeling

- Must run fast
- Discrete velocities

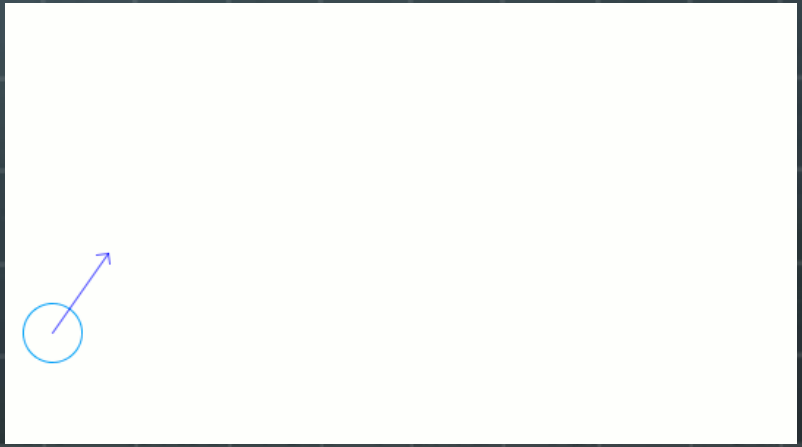


A Frame

Simulation Calculations

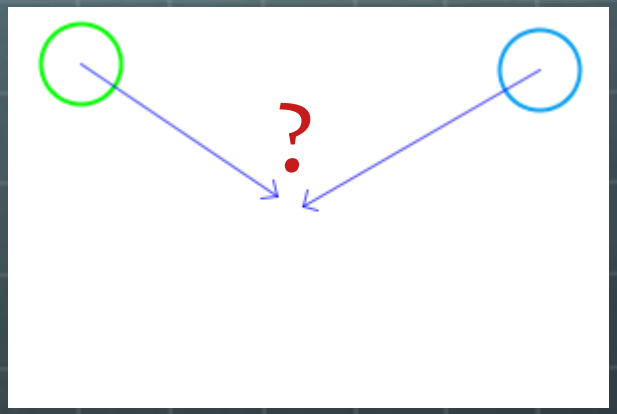
↓

Display to Screen

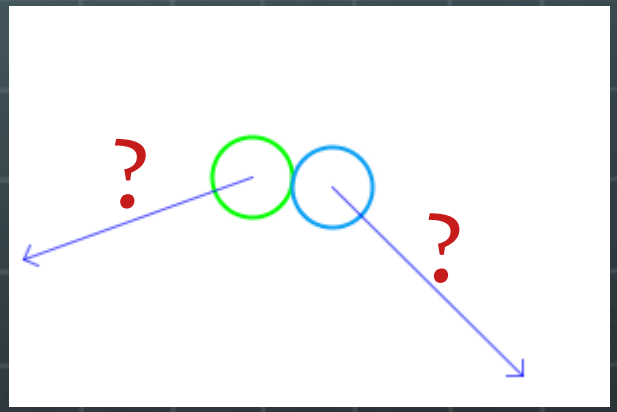


Collision Modeling

- Detection
  - Has a collision occurred?
  - When did it occur?
- Response
  - What is each object's new velocity?

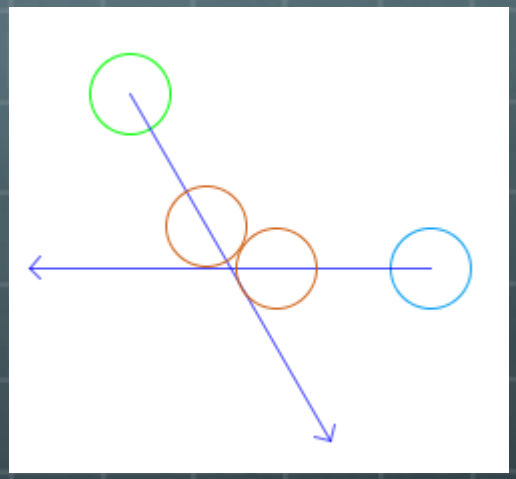


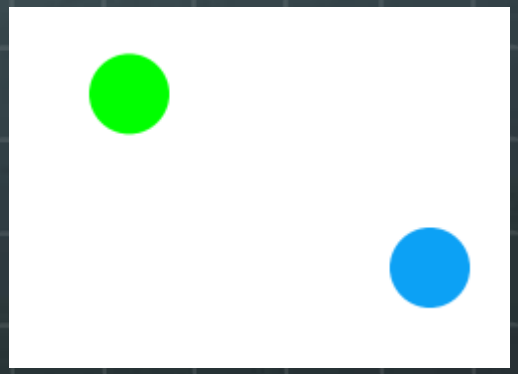
Collision Algorithm →



Collision Detection

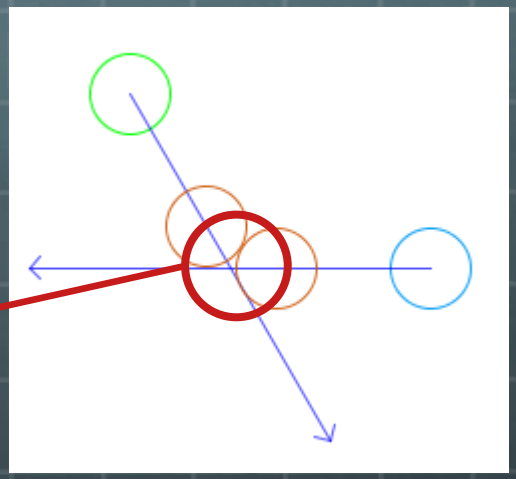
- Collision occurs when:
  - Objects intersect
  - Objects touch
- Time of collision is when objects first touch
  - Where paths intersect

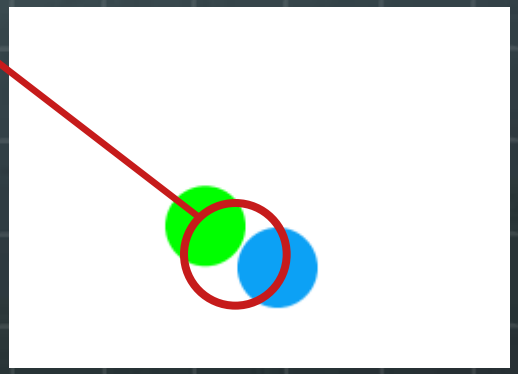




Collision Detection

- Collision occurs when:
  - Objects intersect
  - Objects touch
- Time of collision is when objects first touch
  - Where paths intersect





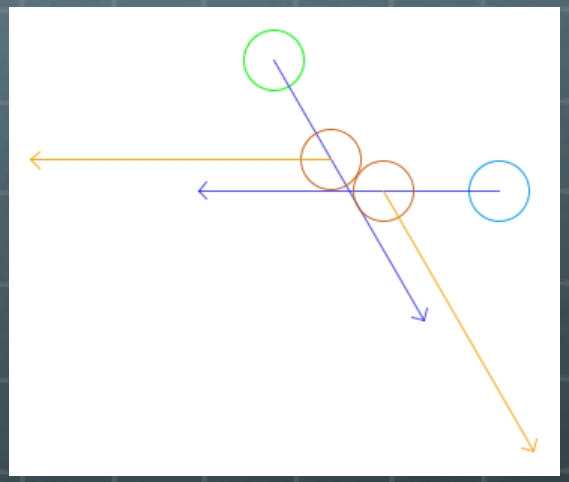
Collision Response

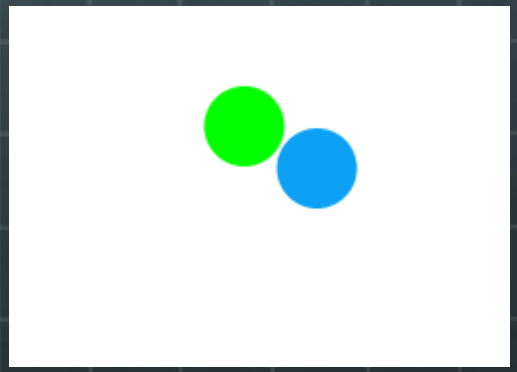
- Conservation of Momentum

$$\sum mv_i = \sum mv_f$$

- Resultant velocity depends on on impact angle
- Energy may be lost
  - Inelastic collision

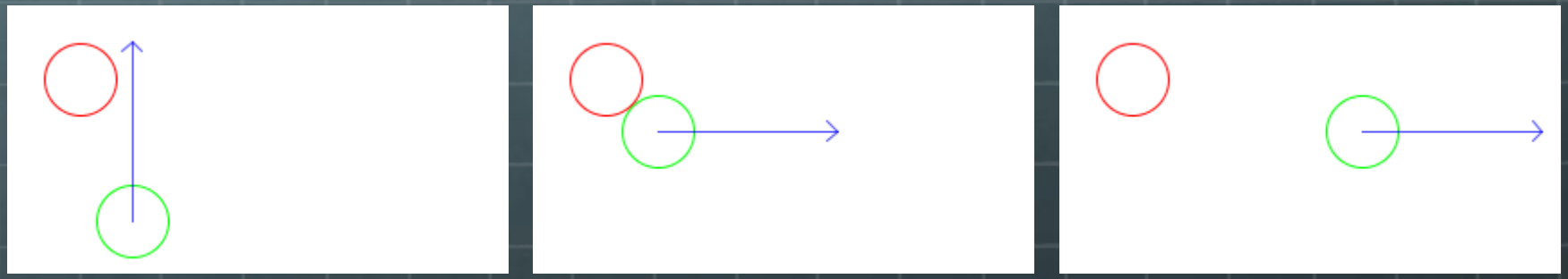
$$v = C_R v_f$$





Single Collision Algorithm

- Common method
- Not realistic



Frame 1 → Collision Algorithm → Frame 2 → Collision Algorithm → Frame 3

My Project

- Increase accuracy by addressing problems
  - More realistic
- Minimize extra computation time

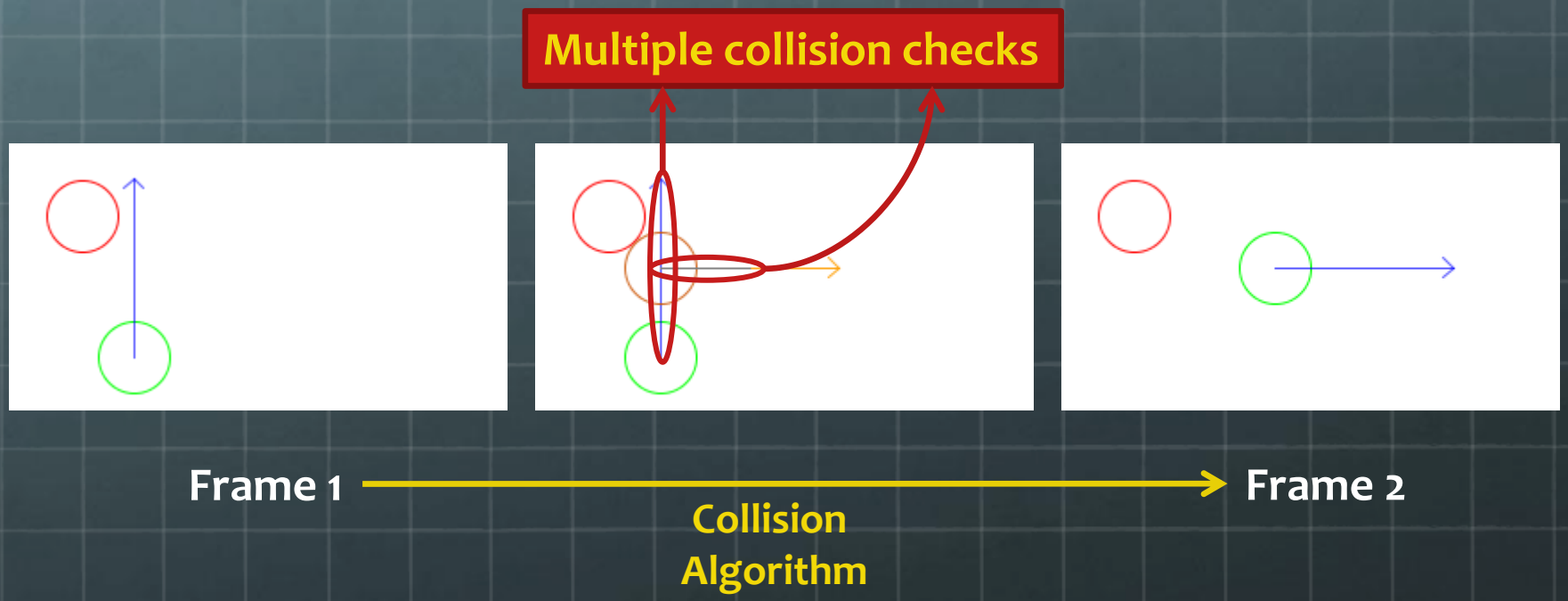
My Approach

- Assume collisions are independent
- <u>Repeat</u> collision algorithm for successive collisions
- After first collision:
  - Avoid repeating calculations
  - Only test distance left to travel

$$x_f = x_i \cdot (1 - timepassed)$$

New Result

- More realistic



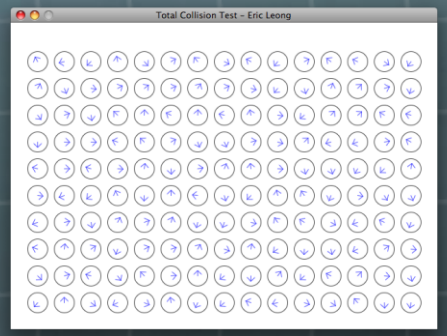
Experiment

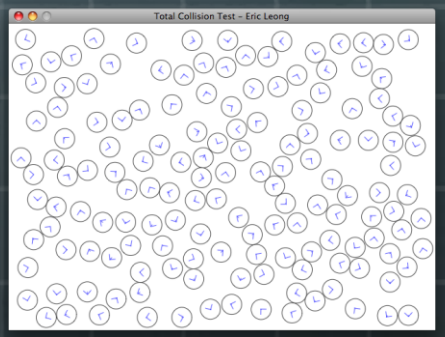
Initial State:

- 150 circles in a 30×50 grid
- Random velocities

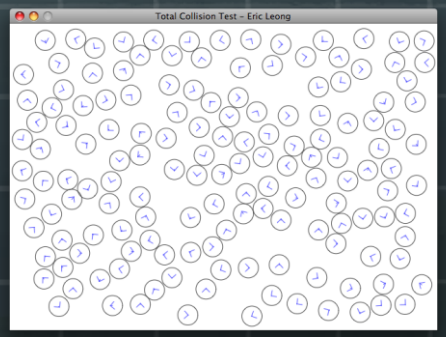
Experiment:

- Run for 600 frames
- Measure time elapsed
- Vary number of collisions



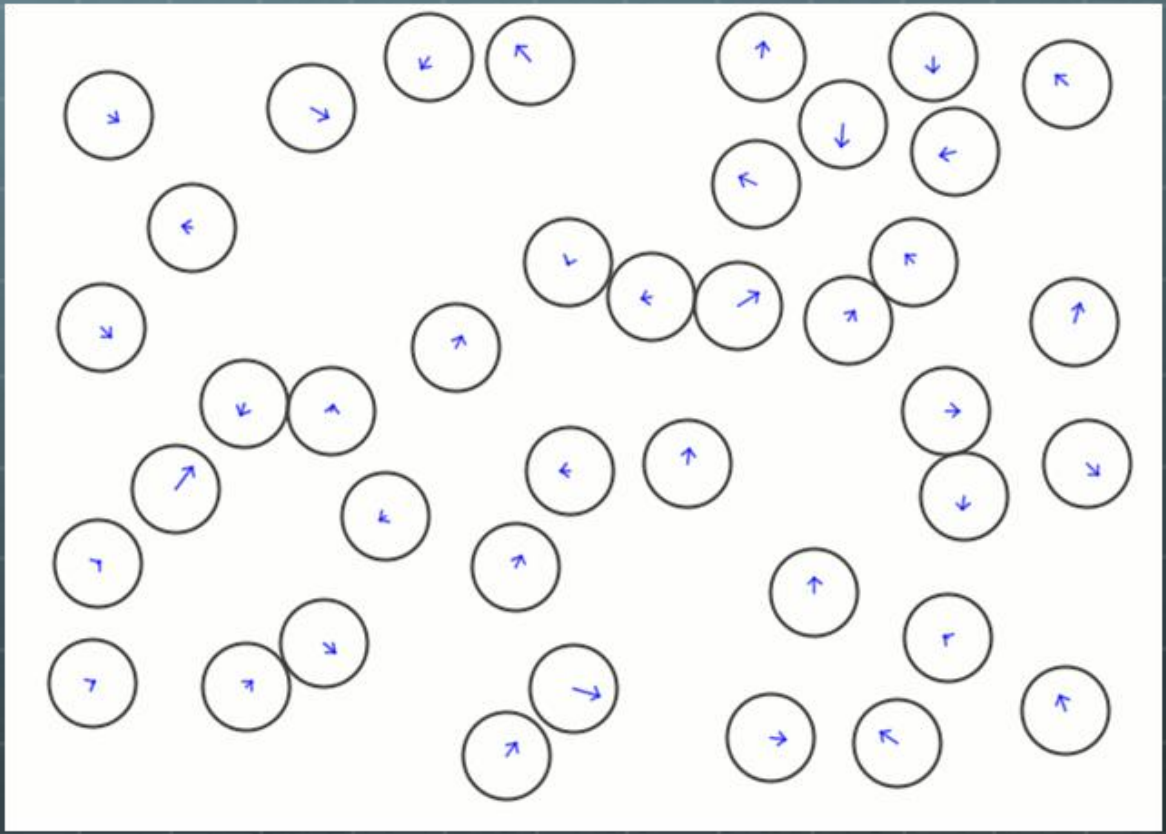


Single Collision Algorithm



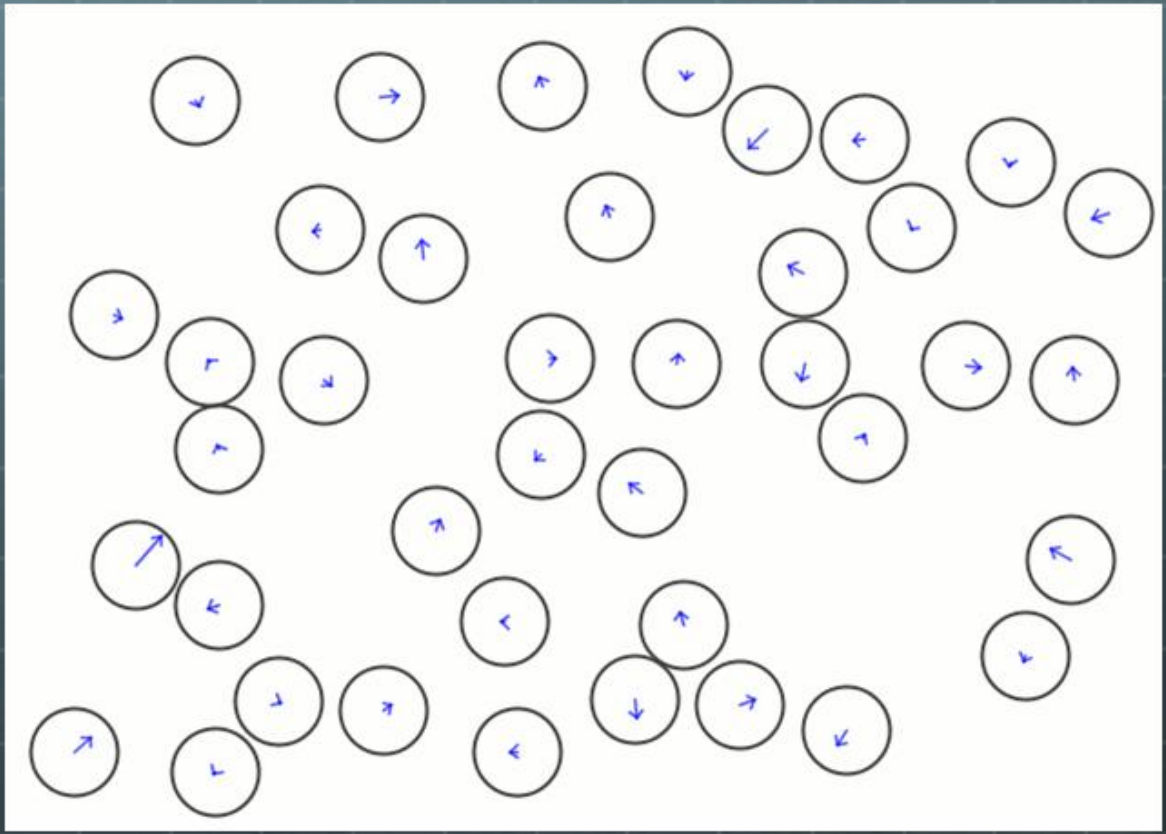
Multiple (2) Collision Algorithm

Single Collision



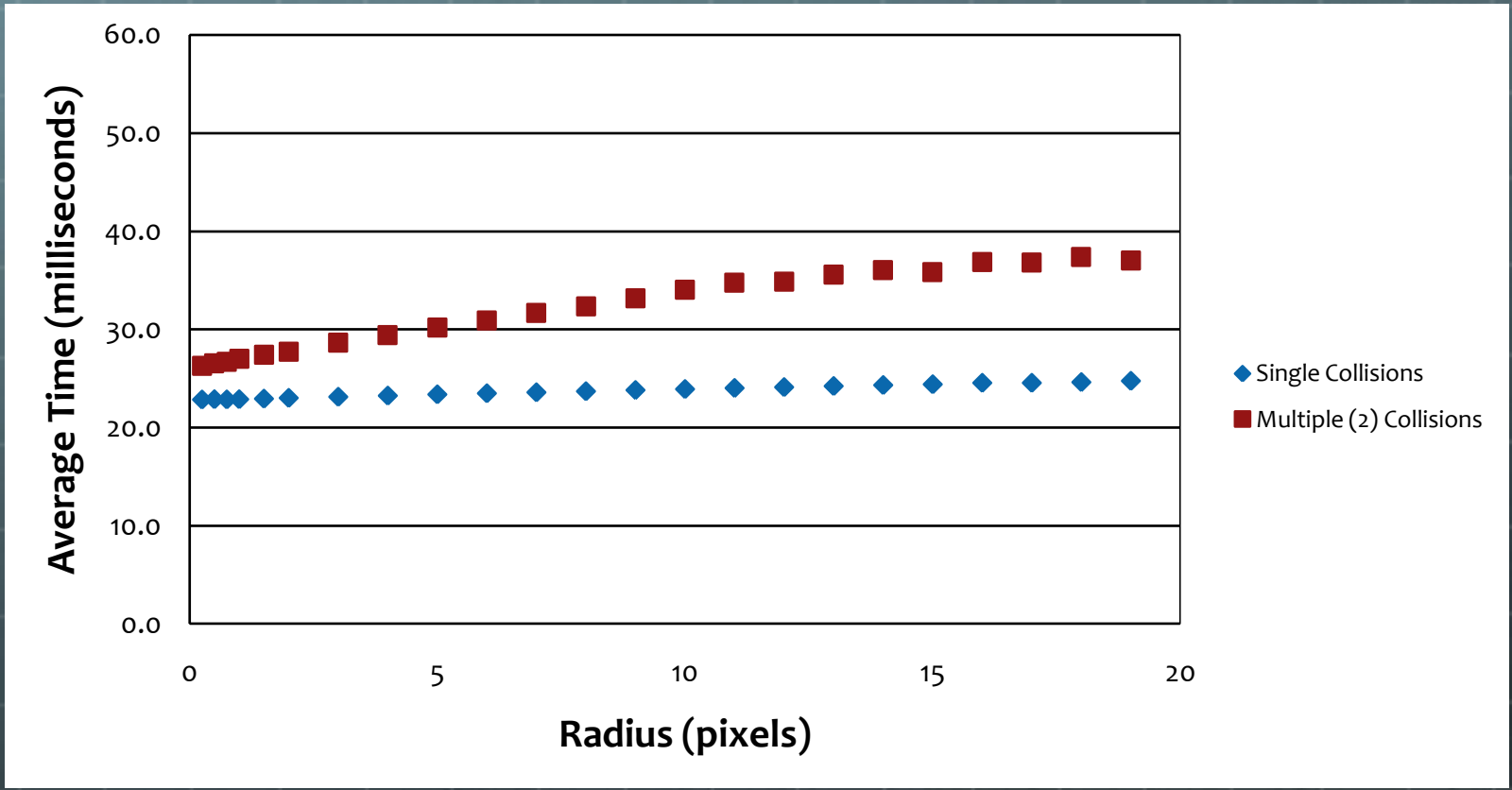
240 frames

Multiple (2) Collisions



240 frames

Computation Time per Frame vs. Frequency of Collisions



Conclusion

50% speed decrease

- Better than running original algorithm twice
- Speed decrease depends on frequency of collisions

Greater accuracy

- Objects do not “stick” to each other
- Objects more evenly distributed
- More realistic

Applications

- Physical Simulations
- Virtual Reality
- Video Games
- Computer-Assisted Design (CAD)
- Animation

BallHead