A NOVEL BROAD-PHASE CONTINUOUS-TIME COLLISION DETECTION ALGORITHM

Master’s thesis by Tarık Kaya

Motivation

- Discrete-time physics simulation is well researched
- Almost no studies exist for continuous-time physics simulation
- Proposed method is an attempt to bridge the gap

General Physics Simulation

- Physics simulation is a broad topic and useful in a diverse set of areas
- We focus on game physics simulation, which has its own special constraints

Game Physics Simulation

Only requirement for game physics simulation is to convince players of:

- Realism
- Predictable consistency (for innovative games)

Similar to game graphics

This is a reason for the popularity of discrete-time physics. It is simpler, more performant and looks “*good enough*”.

Phases of Physics Simulation

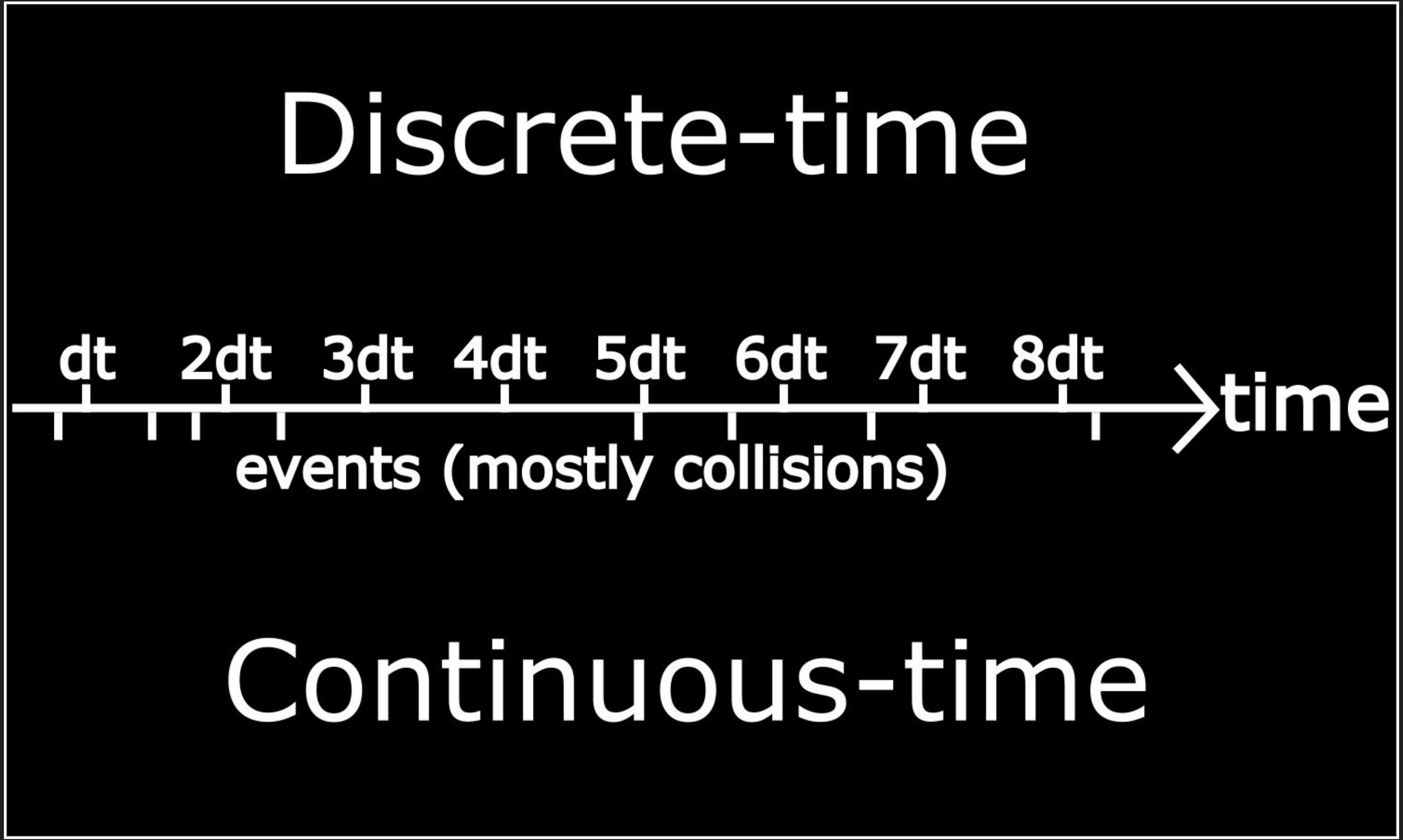
1. **Integration:** Update position and rotation according to linear and angular velocity
2. **Collision detection:** Detect all colliding pairs of objects
3. **Collision resolution:** Resolve all colliding pairs of objects by updating their positions and velocities

When to update simulation?

Two common answers:

1. **Discrete-time:** Fixed updates at fixed time intervals
2. **Continuous-time:** Updates as events occur

Difference between discrete and continuous simulation



Advantages of discrete-time

1. Ease of implementation
2. Better performance in general
3. More stable performance
4. Ease of synchronization with graphics render update
5. Works easily with arbitrary objects with rotational velocities
6. Many known broad-phase methods for optimization

Disadvantages of discrete-time

1. Collisions can be missed
2. Collisions are detected after touching and while overlapping
3. Collisions are handled too late and resolved incorrectly
4. Interdependent collisions are dismissed
5. Works as only an approximation continuous-time

Advantages of continuous-time

1. Simulation with a *theoretically* infinite accuracy
2. A collision is never missed
3. Physical objects never overlap
4. Collisions are resolved at the exact position and time of contact

Disadvantages of continuous-time

1. Implementation is difficult, especially for arbitrary objects with rotational velocities
2. Worse performance in general (Our work tries to overcome this)
3. Hardly any known broad-phase methods exist for optimization (Our work tries to overcome this)

Collision Detection

The most computationally expensive part of physics simulation

Our work tries to optimize this step

Examined under 2 main categories:

1. **Narrow-phase:** Collision detection between 2 objects
2. **Broad-phase:** Collision detection between more than 2 objects

Narrow-Phase Collision Detection

Many known methods such as: SAT, GJK, V-Clipping (*see thesis for the details*)

This thesis uses only simple circle-circle collision detection

Discrete-Time Circle-Circle Collision Detection

Algorithm with constant time and memory complexity:

Check if the distance between the circles is smaller than the total of their radii

$$(c1.x - c2.x)^2 + (c1.y - c2.y)^2 < (c1.r + c2.r)^2$$

Continuous-Time Circle-Circle Collision Detection

Algorithm with constant time and memory complexity, assuming that the square root operation is constant time:

The calculation of the exact time of collision between two moving circles has been simplified to a quadratic equation

Broad-Phase Collision Detection

Naive all-pairs-check leads to an obvious algorithm with quadratic time complexity

For discrete-time collision detection there are many known broad-phase techniques such as: “Spatial partitioning”, “Bounding volume hierarchies”, “Sort and sweep”, whereas hardly any for continuous-time

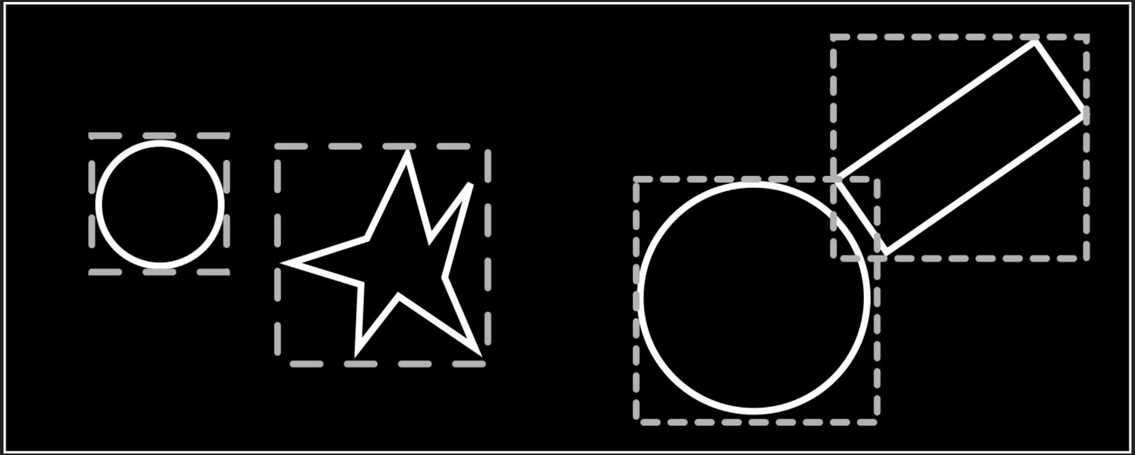
Our work uses only bounding volume hierarchies. We have figured out a way to apply bounding volume hierarchies to continuous-time collision detection.

Bounding Volumes

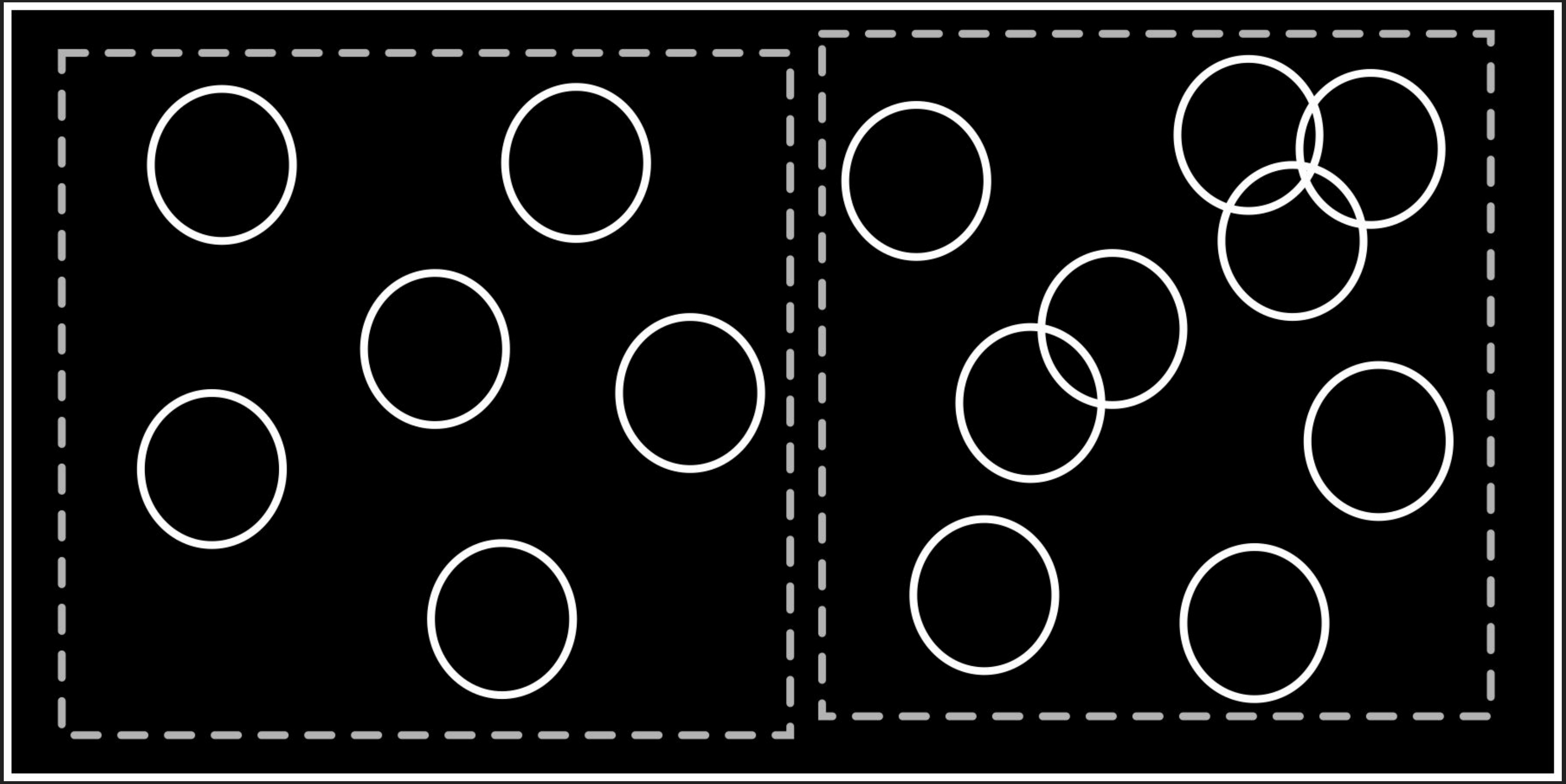
Bounding volumes contain all their underlying object(s).

If the bounding volumes do not collide, then the underlying objects can not be colliding.

If the bounding volumes collide, then the underlying objects may be colliding.



Main Idea of Bounding Volume Hierarchies



A Top-Down Construction of BVH

1. If the number of objects is less than 2, then finished
2. Partition all objects according to a chosen dimension into two equal sets
3. Set bounding volumes of these two sets
4. Recursively apply to both sets

Discrete-Time BVH

Bounding volume hierarchies are invented and used for discrete-time collision detection.

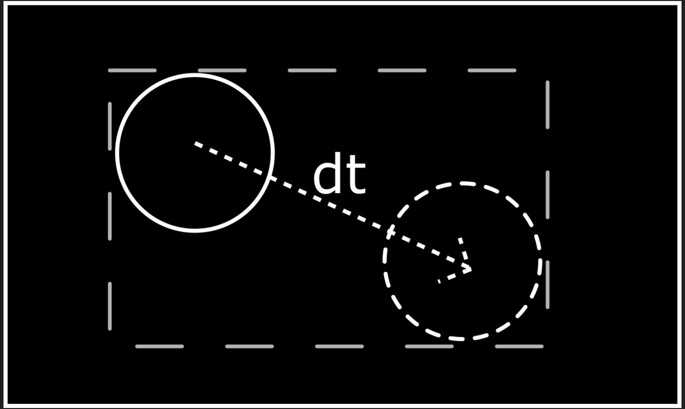
Bounding volumes only cover static objects

We will apply them for continuous-time collision detection

Continuous-Time BVH

Bounding volumes will cover objects at current time and objects at some time step later

This time step is not fixed as in the case of discrete-time collision detection, it will be self adjusting based on the occurrence of collisions



Self Adjusting Time Step

Setting the time step to infinite works, but provides a quadratic-time algorithm.

Our algorithm uses one-sided binary search to determine the optimal time step.

As long as the algorithm does not detect a collision in BVH, we double the time step.

After finding the next collision, our algorithm halves the time step for later collision detections.

Therefore it becomes self adjusting.

The Algorithm

1. Construct the hierarchy with the time step
2. Traverse hierarchy for the next collision
3. If next collision is found: Halve the time step and finish
4. Otherwise: Double the time step, update hierarchy and go to step 2

Complexity Analysis

Our algorithm has quadratic time complexity in the worst case, as in any other broad-phase collision detection methods.

Worst case: all objects move towards a single point to arrive at the same time.

But our algorithm runs much faster than all-pairs check in practice, despite a much bigger constant factor. Our algorithm behaves close to $O(n \log n)$ time in practice.

Proving this average case complexity mathematically is out of the scope of this study, as it depends on the physical setup including problem domain, as well as its initial state (i.e. the positions, velocities, and sizes of objects) Empirical evaluation via simulations has been the best approach we could afford.

Complexity Analysis: Constructing the Hierarchy

Constructing the hierarchy is $O(n\ \mathit{log}n)$

At every level of the hierarchy, our algorithm partitions the objects into two equal halves. This results in log n levels. Partitioning an array is proved to be O(n).

$T(1) = 1$
$T(n) = 2T(n/2) + n = O(n\ \mathit{log}n)$

Complexity Analysis: Updating the Hierarchy

Updating the hierarchy is $O(n)$.

The n physical objects are leaves of the hierarchy-tree. There are 2n-1 bounding volumes.

Updating every bounding volume takes constant time, as the hierarchy is a binary-tree.

Complexity Analysis: Hierarchy Traversal

Best case: No pairs of bounding volumes collide = $O(n)$

Worst case: Every pair of bounding volumes collides = $O(n^2)$

Thus the average time complexity depends on the physical setup and the quality of the hierarchy.

Empirical average case = ~$O(n\log n)$

Complexity Analysis: Number of Time Step Changes

We use one-sided binary search to determine the time step.

This leads to a $\mathrm{O}(log\mathrm{t})$ algorithm, where t stands for time of next collision.

In practice this step behaves much closer to $\mathrm{O}(1)$, because of the spatial and temporal coherence.

Complexity Analysis: In Total

We construct the hierarchy once for each collision detection = $O(n\ log n)$

We update time step and hierarchy logt times = $O(n\ log t)$

After the first successful hierarchy traversal we finish = ~$O(n\ log n)$

In total: $O(n\ log n + n\ log t)$

Execution times are in agreement with this finding

Memory complexity: n bodies + $(2n - 1)$ bounding volumes = $O(n)$

Running Times

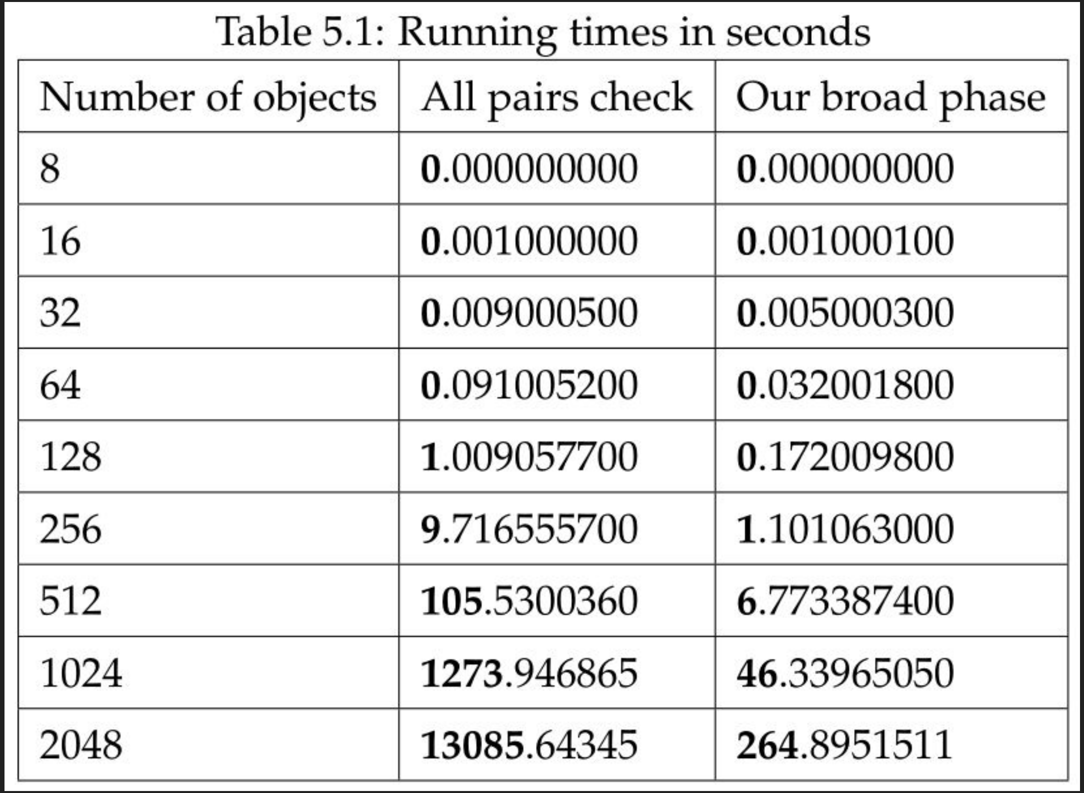


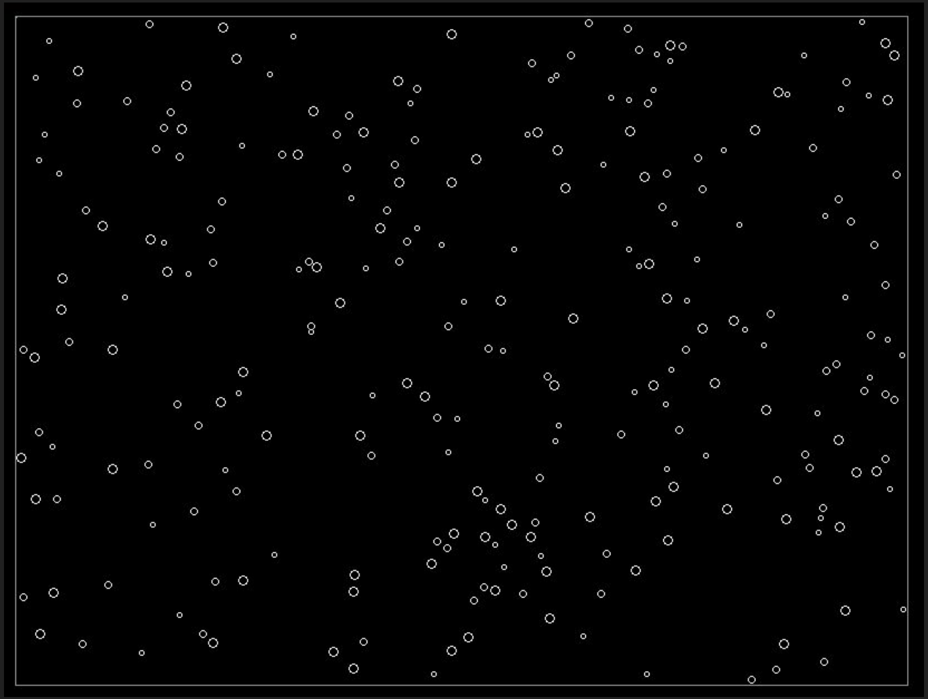
Table 5.1: Running times in seconds

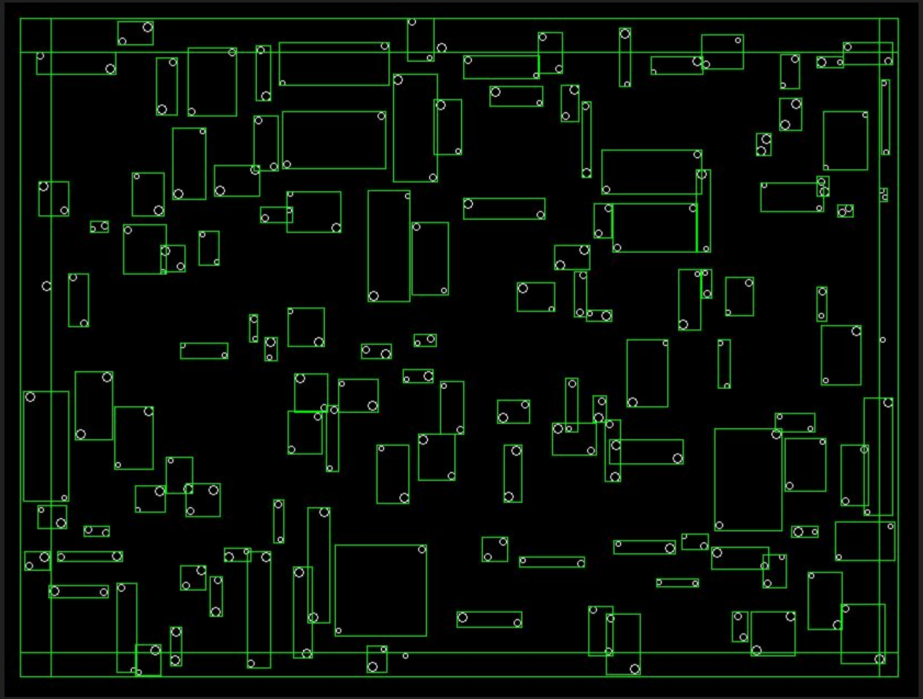
Number of objects	All pairs check	Our broad phase
8	**0**.000000000	**0**.000000000
16	**0**.001000000	**0**.001000100
32	**0**.009000500	**0**.005000300
64	**0**.091005200	**0**.032001800
128	**1**.009057700	**0**.172009800
256	**9**.716555700	**1**.101063000
512	**105**.5300360	**6**.773387400
1024	**1273**.946865	**46**.33965050
2048	**13085**.64345	**264**.8951511

Shortcomings

1. We have provided only a broad-phase method for the continuous-time collision detection. We have used only circles to simplify the narrow-phase process. For a general continuous-time collision detection, new narrow-phase methods have to be invented, so that they can handle arbitrary objects with angular velocities.
2. We have disregarded air friction (damping) in our simulation, because there are many different models of damping. Some damping models allow constant time narrow-phase collision detection, whereas some may not.

Screenshots from our Visualization Tool





Thanks

Any questions?