CMPUT 350 - AI Part 3
Path Planning Odds and Ends

Michael Buro, Jake Tuero

Department of Computing Science, University of Alberta
Edmonton, Canada
tuero@ualberta.ca

September 1, 2024

Change Log

Outline

1. Multi-Agent Path Planning
2. Fast Collision Tests
3. Collision Resolving Algorithm

Multi-Agent Path Planning

We can get one object from location A to B quickly

What about groups of units?

New challenges

- choke points, units get clogged up
- how to stay in formation?
- opponent denying access

Multi-Agent Path Planning

Naive approach: every unit finds path individually

For squads even simpler: one unit finds path, the others follow

Myopic self-centered view creates serious problems in confined areas

Worst-case: multiagent pathfinding problem turns into sliding tile puzzle

Therefore, finding the shortest solution is NP-hard in general

Handling Collisions

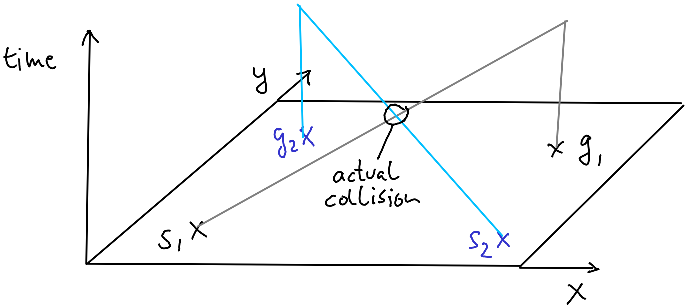
Simple approach: assume world is static, plan paths with A^*, and replan if something changes

But this can fail miserably. Doesn't take into account moving obstacles

By the time we are there, the objects has moved on - so that wasn't really a collision we had to plan for

Space-Time Collisions

Instead, we must avoid collisions in space-time, i.e. prevent objects to be at the same location at the same time



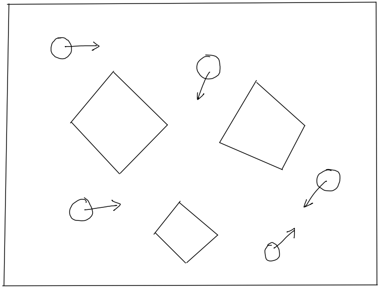
This can be done with A*. But computing optimal paths is very costly: we need to look at the joint-action space!

Many publications deal with this problem. See [3], for example

ORTS Kinematics System

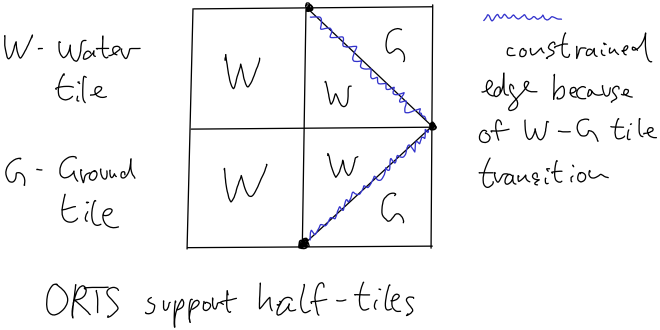
This section describes the kinematics system implemented in the ORTS (Open Real Time Strategy) game engine

- The game server sees the world as set of line segments and circles
- Segments are obstacles for moving objects (circles)
- Circles move in straight lines
- When circles collide with segments or other circles, they just stop



ORTS Kinematics System

To simplify creating maps, tiles are used to define ORTS worlds

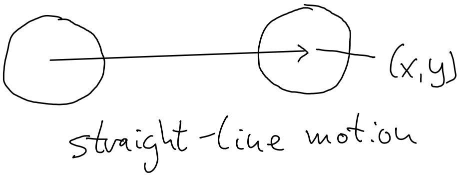


Server generates line segments from tiles

Object Motion

Move commands set objects in motion: obj.move(x, y, s),

where (x, y) is the goal location and s is the speed

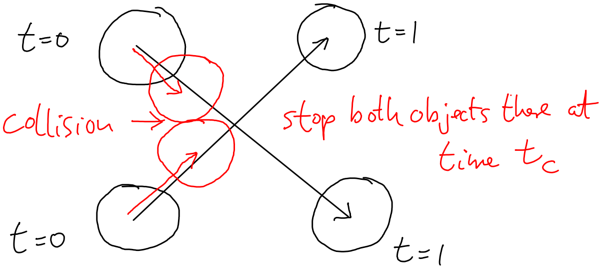


$\leadsto$ paths must be broken down into straight-line segments

Collisions

In each simulation cycle, moving objects are advanced

They stop when colliding with segments or other objects



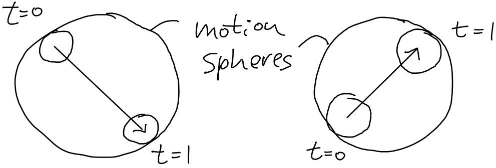
Collision time results from solving quadratic equation

Pairwise collision test is expensive

n moving objects $\rightsquigarrow$ $\Theta(n^2)$ collision tests

Faster Collision Tests – Using Motion Spheres

Objects don't collide if their motion spheres don't intersect



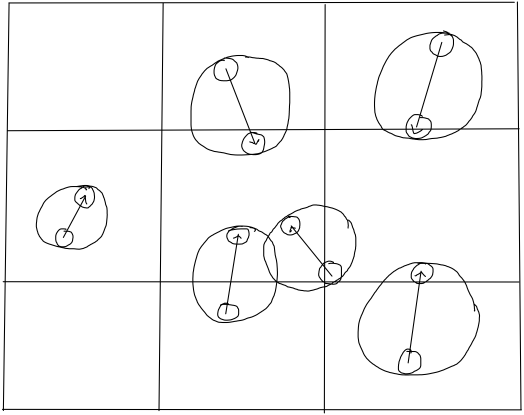
Motion spheres contain all points the object is passing on the way to the goal location

Any bounding shape will do, but using spheres simplifies pairwise intersection tests

How?

Sector-Based Collision Test

Superimposing sectors can reduce the number of pairwise intersection tests considerably



Then the pairwise test only has to be executed for all objects whose motion spheres intersect with a sector

Efficient Collision Resolution

The following collision resolution algorithm computes all collision times first

Beginning with the earliest it

... advances all objects to that point in time ...

... stops the colliding objects

... re-computes nearby objects' collision times

... and continues

If n objects are spread out well, this algorithm runs in $O(n \log n)$ time

Collision Resolving Algorithm

```
compute motion spheres
populate sectors with motion spheres
construct motion sphere intersection graph G
    from sectors
compute connected components of G

for each connected component do
    create priority queue with collision times earliest first
    while (queue not empty) do
        pop (collision time t, edge)
        resolve collision
        advance every object to time t
        update collision times in neighborhood
    end
    move non-colliding objects
end
```

For more details, see [1][2] below

References

[1] M. Buro, ORTS: A Hack-Free RTS Game Environment, Proceedings of the International Computers and Games Conference 2002, Edmonton, Canada. pp. 280-291

[2] M. Buro and T. Furtak, RTS Games and Real-Time AI Research, Proceedings of the Behavior Representation in Modeling and Simulation Conference (BRIMS), Arlington VA 2004, pp. 34-41

[3] N.R. Sturtevant and M. Buro, Improving Collaborative Pathfinding Using Map Abstraction. Proceedings of the AIIDE conference, Marina del Rey 2006, pp. 45-50