Client-side motion and physics scripting in distributed virtual worlds

Will Monroe
CURIS Summer 2011
Mentors: Philip Levis, Ewen Cheslack-Postava, Behram Mistree

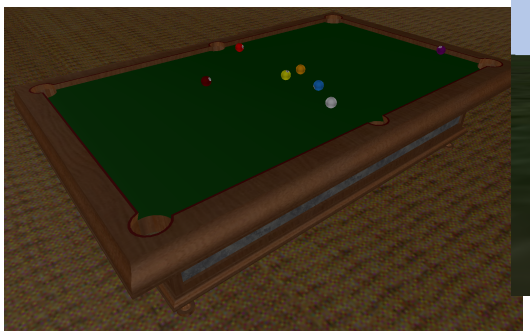
Problem

Scripting complex motion in a distributed virtual world is tedious and error-prone. A centralized approach (one script controlling many objects) is simpler, but doesn't leverage the ability to distribute computation among multiple clients.

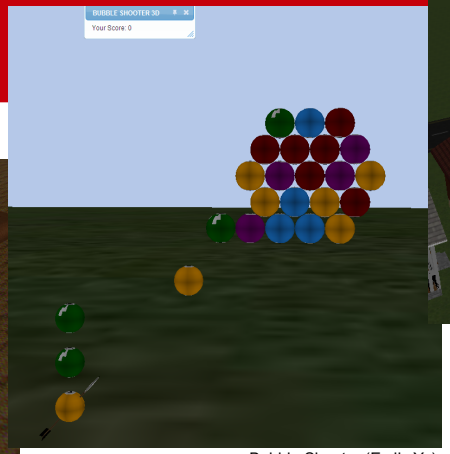
What is needed

A library to abstract away the burdensome details of motion control while still allowing customization of each object's motion on the client.

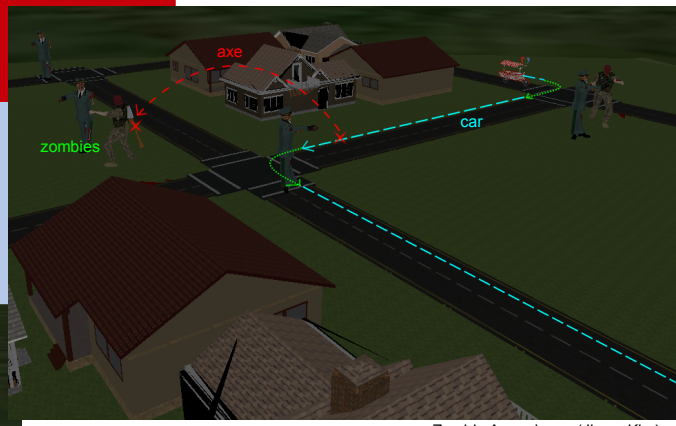
```
function rollingVelFn(pres) {
    if(pres.velocity.length() < 0.004)
        return <0, 0, 0>;

    return pres.velocity -
            pres.velocity.normal() * 0.004;
}

for(var i in balls)
{
    // make balls bounce off each other
    balls[i].coll = new motion.Collision(
        balls[i],
        coll.TestSpheres(balls),
        coll.Bounce(0.8)
    );
    // confine balls to within the table
    balls[i].bounds = new motion.Collision(
        balls[i],
        coll.TestBounds(
            TABLE_BOUNDS.max,
            TABLE_BOUNDS.min
        ),
        coll.Bounce(0.8)
    );
    // make balls gradually slow and stop
    balls[i].friction = new motion.Velocity(
        balls[i],
        rollingVelFn
    );
}
```



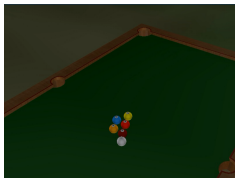
Billiards (Will Monroe)

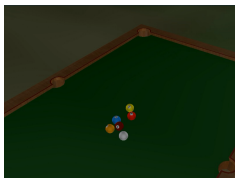


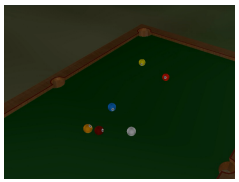
Bubble Shooter (Emily Ye)

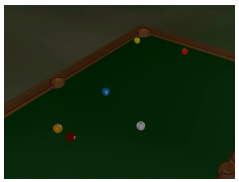


Zombie Apocalypse (Jiwon Kim)









Challenges

- **Reduce complexity for users**
- **Correct for network latency**
- **Properly handle decentralization**

Implementation

Polling

A polling-based system with regular queries and updates allows the user to painlessly modify the properties of an object's motion in response to events in the virtual world.

Sensible defaults

The library assumes reasonable default values for most aspects of motion, so the user needs to give less information. For example, gravity assumes a downward acceleration of 9.8 m/s^2, unless the user specifies something different.

A strong supporting library

Behind the highly customizable core is a library of ready-made functions that implement types of motion anticipated to be common.

Hooks for custom callbacks

Allow the user to write her own routines to build new types of motion on top of the basic library functionality.

Send a snapshot of the world with messages

Store positions and velocities computed during collision detection and send them with collision notification messages, so the circumstances of the collision can be reproduced after the message has traveled across the network.

Filter out duplicate collisions

Compare the velocities in the message to locally stored velocities, and if the local velocity has already changed, ignore the message.

Future improvement

Sensitivity to polling rate

Optimize the polling system so slowdowns caused by rendering of complex scenes or collision detection among large numbers of objects don't cause accuracy to degrade.

Multipresencing: centralize when possible

Adapt by interacting across the network only when necessary, so hosting many objects on the same client can improve performance.