PHYSICALLY REALISTIC HAPTIC INTERACTION WITH DYNAMIC VIRTUAL WORLDS

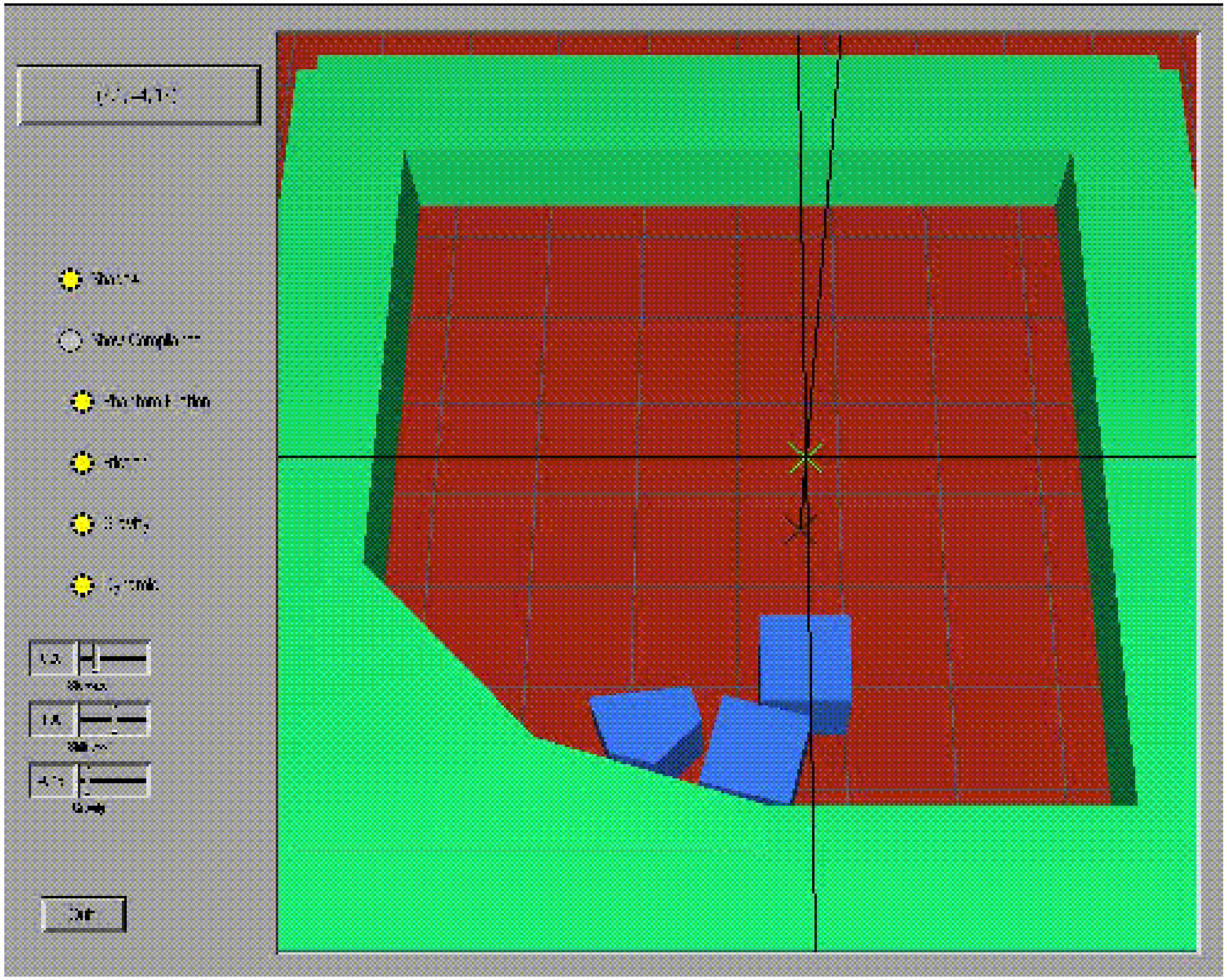
Sundar Vedula and David Baraff

The Robotics Institute

Carnegie Mellon University

Carnegie
Mellon

Virtual Environment Example



Carnegie Mellon

Goal

To be able to create an interactive dynamic virtual environment, whose entities have definite physical properties, and have them behave in a physically correct manner under the action of external forces.

Allow a user to manipulate this environment with a probe, and feel the corresponding forces back.

Carnegie Mellon

Physically Based Modeling

- Model objects as polyhedral, rigid bodies
- State = position and velocity for all degrees of freedom
- Given:

    Object geometries, initial state, External Forces

  Solve for:

    New state

  Subject to:

    kinematic constraints

Carnegie Mellon

Physically Based Modeling(cont.)

- In short, solve

$$\frac{d}{dt}\begin{pmatrix} x(t) \\ \theta(t) \\ mv(t) \\ I\omega(t) \end{pmatrix} = \begin{pmatrix} v(t) \\ \omega(t) \\ F(t) \\ \tau(t) \end{pmatrix} \quad \textbf{(1)}$$

- Constraints can be equalities or inequalities
- Permanent and Transient Constraints enforced by additional constraint forces (not penalty)
- Coulomb friction model used
- Step through time, backtrack on collisions
- Display updated world model
- Key requirement: Display at real-time, i.e. 30Hz
- Plenty of published work

Distributed Processing

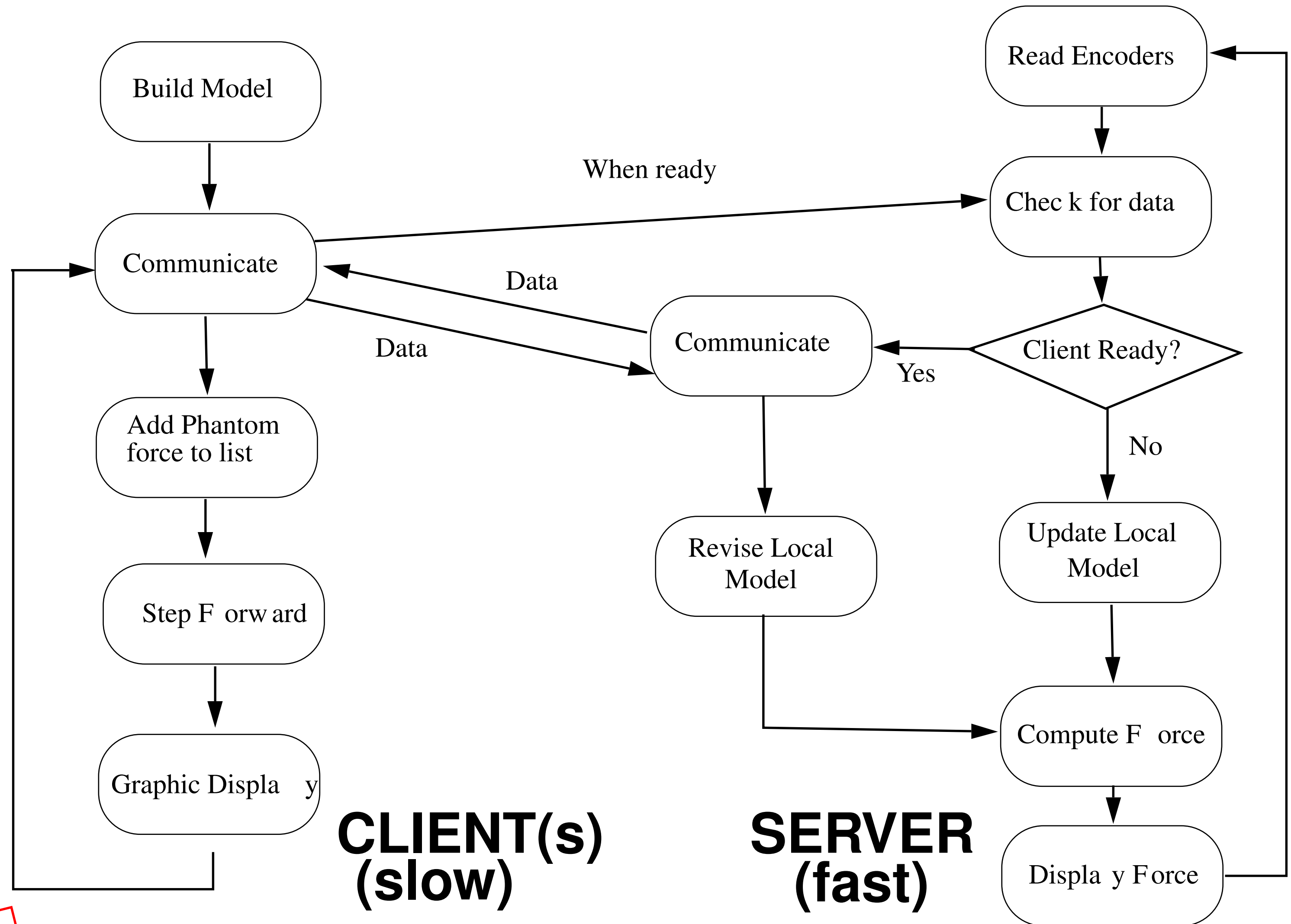
Why?

- Generalized physical simulation, with display can only go at around 30Hz
- Better modularity
- Force server doesn't need to know about model away from the probe location
- Servo rate independent of model complexity (easier real-time control)

How?

- Use one processor for dynamic simulation and graphics, and one for low-level force control of each haptic interface
- Have simulator communicate with each force server separately

Multi-processor model



Carnegie Mellon

Simulation algorithm

- **Simulation processor acts as haptic *client***
- **Query each force server for position and force information (what object, where, and how hard the user pushed on during the last cycle)**
- **From position, determine appropriate local model for each probe, and communicate to server**

  force(object->probe) = - force(probe->user)
  = - force(probe->object)
- **Add haptic forces to list of external forces for simulation**
- **Step forward in simulation and display new state**
- **Display <u>constrained</u> probe position**

Carnegie Mellon

Force Server Algorithm

- Check to see if simulator has new local model
- If yes,
  - send current probe position, amount and location of force applied during previous cycle
  - Receive new local model (position, velocity, and orientation of relevant plane, if any)
- If not,
  - Update local model appropriately (more later)

- Compute normal force from penetration using a simple penalty method
- Compute frictional force using Coulomb model
- Display force to user

Local Model Schemes

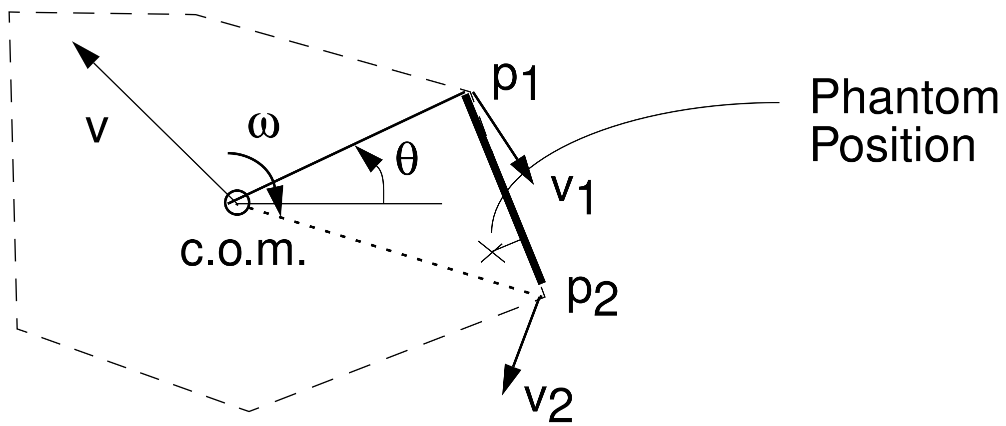
1. Closest Plane
   - Use closest plane for force computation
   - Position Discontinuities at client update rate (nasty clicking at 30 Hz)

2. Plane with velocities of end-points
   - Approximate velocity of contact point by linear interpolation between edges
   - In 2-D, edge length changes
   - In 3-D, nasty things happen to plane dimensions
   - Works for objects with high moments of inertia, i.e., ones that don't rotate easily

Carnegie Mellon

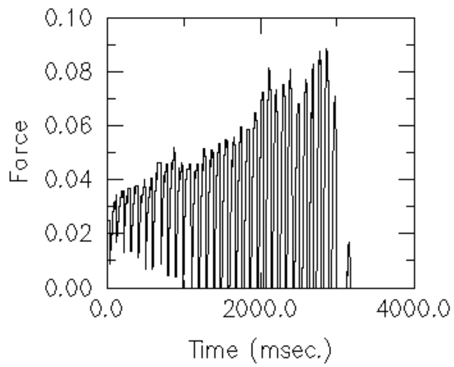
Local Model Schemes (cont.)

3. Full Kinematic Interpolation

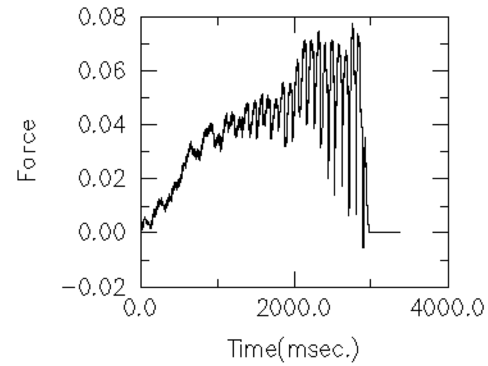


- **Position, Velocity, Orientation, and Angular Velocity of c.o.m.**
- **Local co-ordinates of plane boundaries**

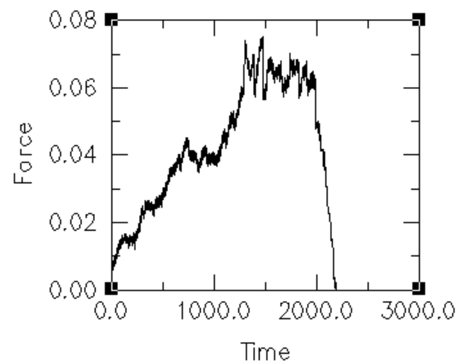
Comparison of Local Models



Plane Interpolation



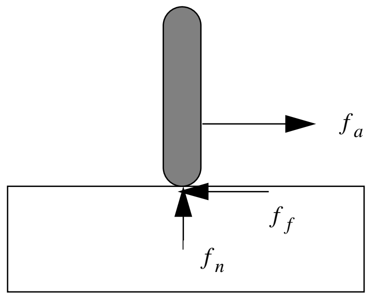
Velocity Interpolation

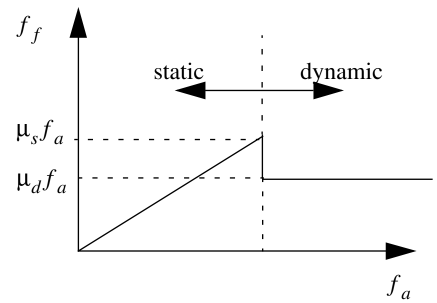


Kinematic motion model

Carnegie
Mellon

Coulomb friction model





- Static friction is a function of x, $x_{stopped}$, and k
- Dynamic friction is a function of v, v_l, and v_h
- Switching between friction modes avoided by a hysteresis technique
- Switching from dynamic to static friction should not involve force discontinuity, so $x_{stopped}$ and k are set appropriately

Carnegie Mellon

Why is Coulomb friction hard?

- Firstly, we have no force sensors to measure force, to determine impending direction of motion
- Secondly, no tachometers to measure velocity, to decide between static and dynamic friction
- Workarounds:

    Exponential differentiating filter on position to approximate velocity

    Threshold velocity to determine if one has *stopped*
- Reasonable results with right threshold and spring constant for static friction
- Strongly dependent on good velocity estimate near zero

Carnegie Mellon

Results

- Dynamic simulation - 40Hz on 250MHz R4400 SGI Indigo Extreme
- Phantom™ haptic interface controlled at 10kHz on 200MHz Pentium Pro
- 3-D haptic display with 2-D dynamic simulation, i.e. objects can be pushed in the plane, but their top and bottom surfaces can be felt but not pushed
- Preliminary manipulation experiments with two probes demands experience, and finite sized fingers
- Can assign physical properties to probe and make it a part of the simulation - stability issues become significant

Conclusions

- Framework for effective haptic interaction using multiple force feedback devices with single physically based simulation system
- Ability to work around measurement of force and velocity
- Local model update strategy for good dynamic performance at low communication rates
- Coulomb friction model for surface properties
- Support for multiple, distributed users manipulating one virtual environment

Carnegie Mellon

Research Issues

- How much knowledge on the force server?
- Choices for concave surfaces
- Better control strategies
- Is second-order dynamics really good with a point tool? (especially in 3-D)
- Ways of constraining motion
- Interfaces in 3-D for better interaction
- Manipulation and HCI issues - what can you really do with a point?
- Ergonomic considerations