Announcements

- Continue to bring your laptop and power cord to class for the rest of the quarter.
- If you need to adjust the mechanics of your Hapkit, please do this before programming your virtual environments
- Aim to get checked off on **Lab 5** by the end of class Thursday.

ME 20N: Haptics: Engineering Touch
Autumn 2017

Week 5: Programming Virtual Environments

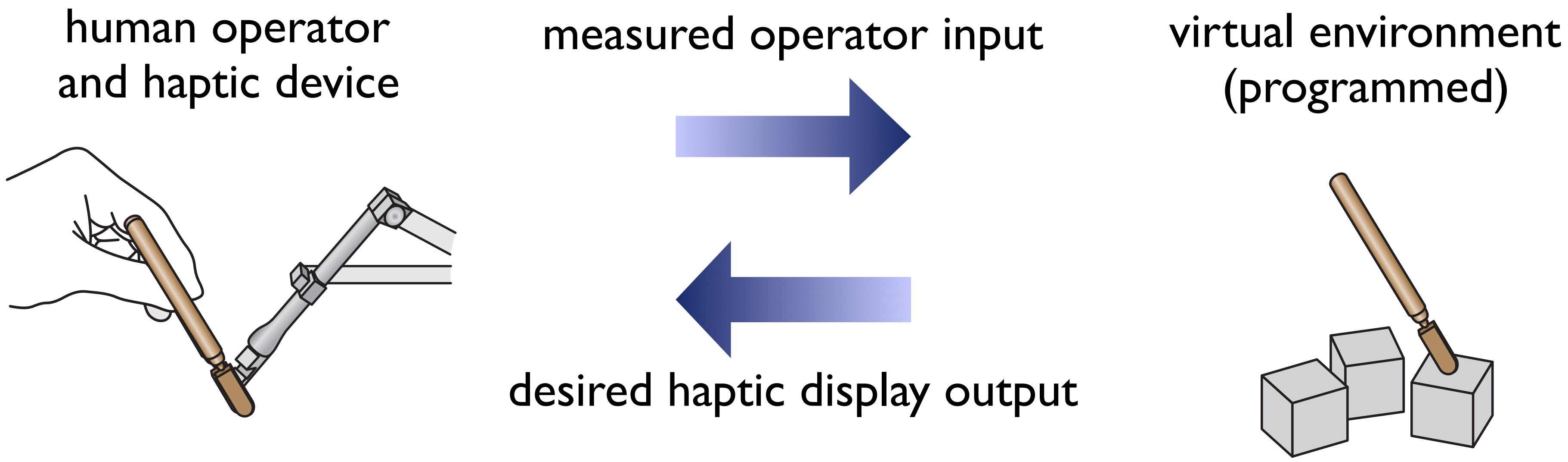
Allison M. Okamura
Stanford University

Haptic Rendering

Allison M. Okamura
Stanford University

Haptic Rendering

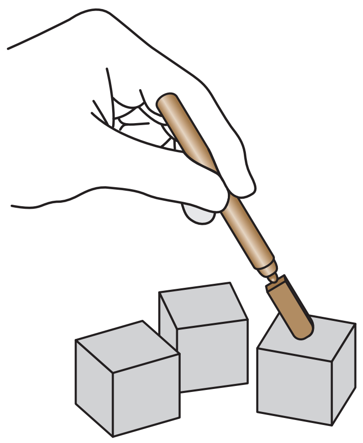
is the process of computing the force
resulting from contacts with virtual objects
based on measurements of the operator's motion.


human operator and haptic device
measured operator input
virtual environment (programmed)
desired haptic display output

Haptic Rendering

is the process of computing the force
resulting from contacts with virtual objects
based on measurements of the operator's motion.

what the user should feel:



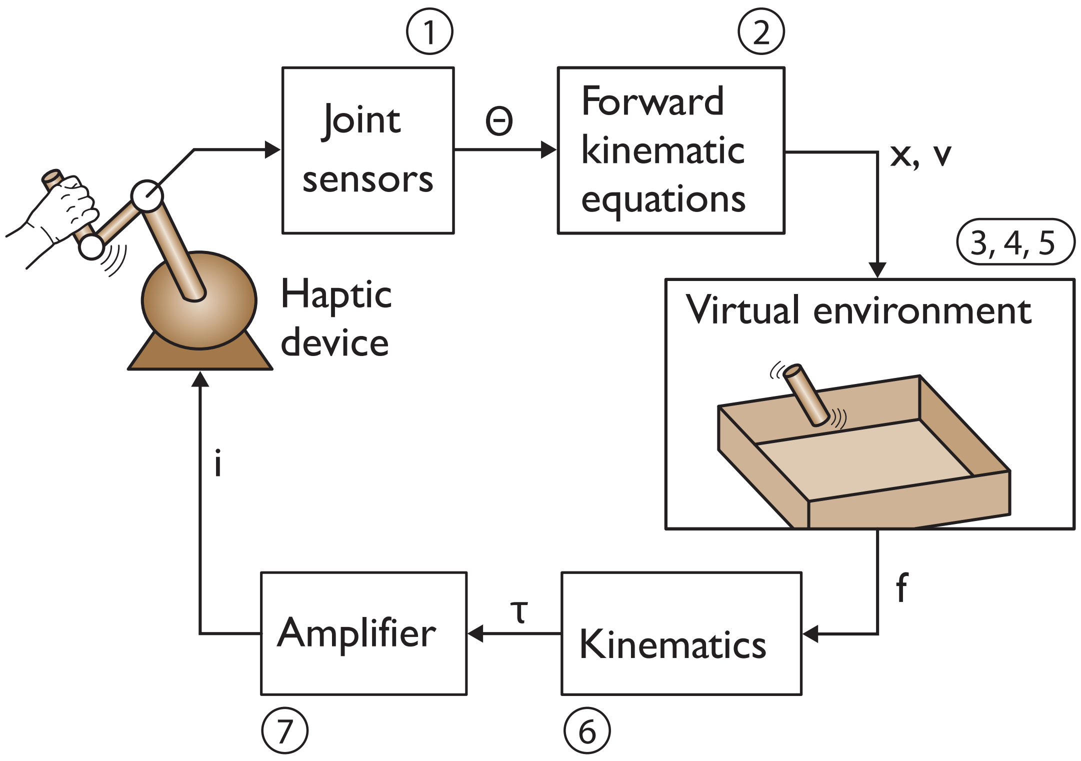
Transparency

is the idea that the user feels as if she is directly interacting with the virtual environment.

That is, she:

- Feels the virtual environment exactly as the designer/programmer intended
- Does not feel other forces, such as those arising from the mechanism of the haptic device (mass, friction, etc.)

The Haptic Loop



To begin, the user **moves** the haptic device

1. **Movement** of the device is sensed
2. **Kinematic equations** are used to find the motion of the haptic interaction point
3. If necessary, **contact** with object(s) in the virtual environment are detected
4. If necessary, the relevant point of the **surface** of the virtual object is detected
5. The **force** to be displayed to the user is calculated
6. Kinematics are used to determine **actuator commands**
7. An **amplifier** is used to send current/ voltage to the actuator

The user **feels a force** from the haptic device

Calculating the Force

The force is a function of user movement

You will decide this function
and program it in order to
create a compelling haptic virtual environment,
which may be:

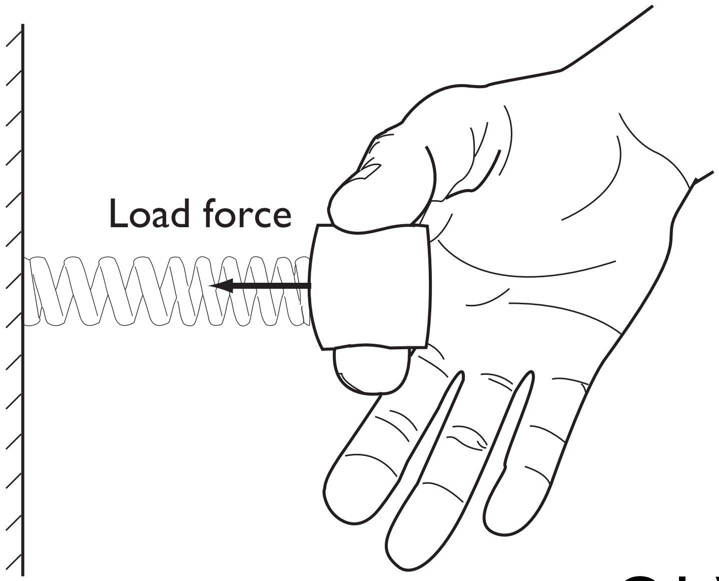
- Realistic
- Unrealistic
- Fun
- Educational
- Expected
- Unexpected

Rendering Specific Haptic Effects

Allison M. Okamura
Stanford University

Virtual Spring

$$F = -kx$$

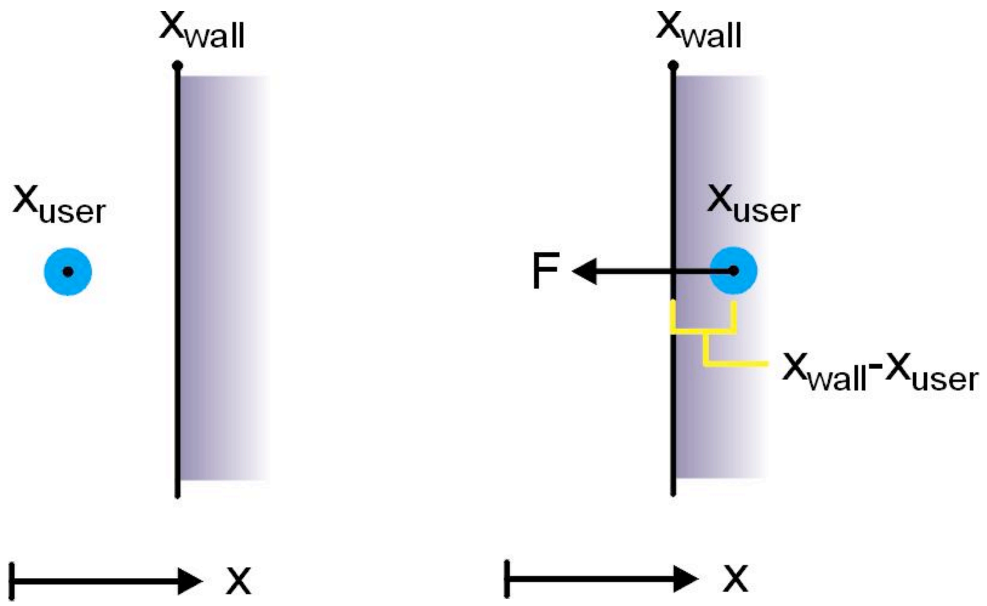


f is the force to be felt by the user
k is the stiffness of the virtual wall
x is the position of the handle

$x = 0$ at the equilibrium point of the spring

Q1: What do you expect to feel when you move your hand back and forth?
Q2: What would happen if you get the sign wrong?

Virtual Wall

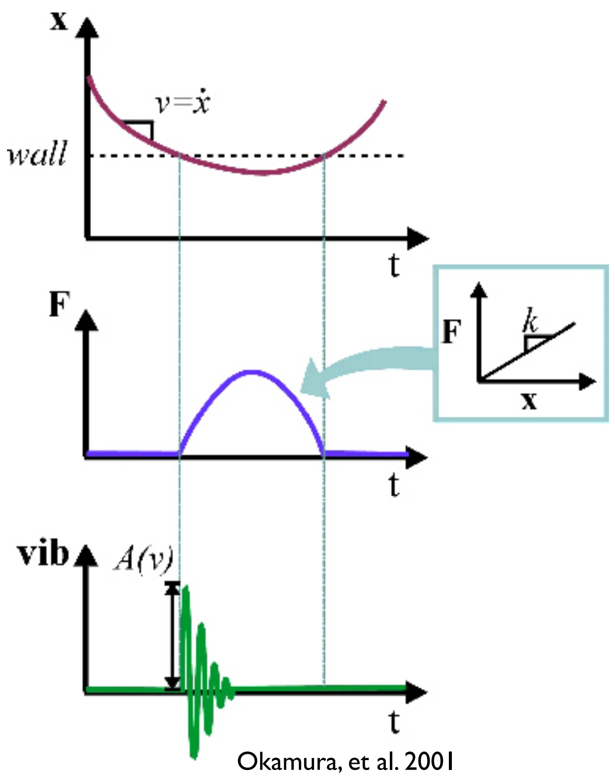


$$\text{If } x_{user} > x_{wall}, F = k(x_{wall} - x_{user})$$
$$\text{stiffness } k > 0$$

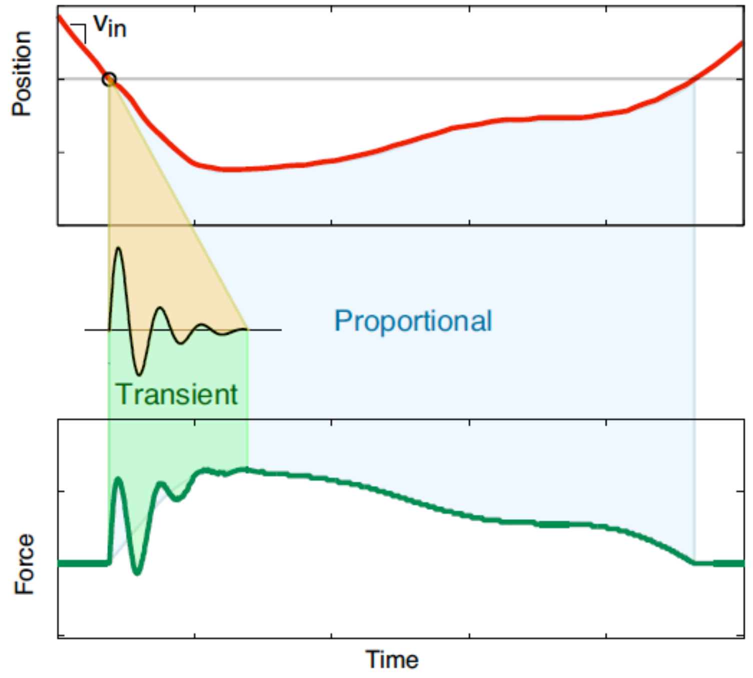
Q1: What do you expect to feel when you move your hand back and forth?
Q2: What would happen if you get the sign wrong?

How to make a virtual wall feel stiffer?

One way: Impact Vibrations



Okamura, et al. 2001



Kuchenbecker, et al. 2006

Virtual Damper

$F = -bv$

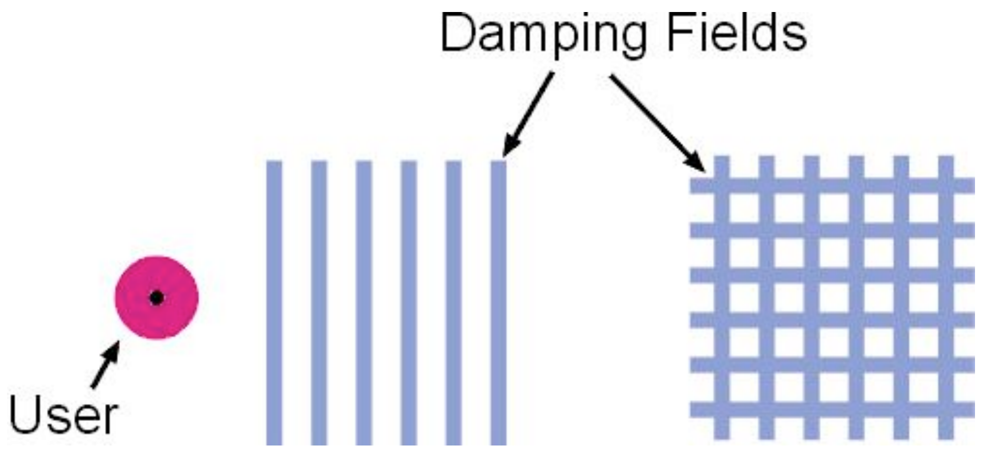
f is the force to be felt by the user
b is the damping coefficient / viscosity
v is the velocity of the haptic interaction point (user)

Q1: What do you expect to feel when you move your hand back and forth?
Q2: What would happen if you get the sign wrong?

Virtual Texture

There are many possible ways to do this, here is one:

if x is inside a damping area,
$F = -bv$



note that vibrations may occur due to discontinuity in force

Step 1: Render a Virtual Spring

We already have the code to measure the user's position and output a force.
Now we will connect the two by rendering a virtual spring.

1. Comment out any serial monitor print lines, which slow down the loop and may make your spring feel discretized.
2. Program the spring into your Hapkit code, in the function called `hapkitRenderSpring`.
3. Start with a small stiffness and gradually increase until you feel something. Think about how stiff your choice of stiffness (in units of N/m) feels -- does it make sense?
4. Test that the Hapkit can return to center by itself with an appropriate value of stiffness. It is okay to have a slightly oscillatory response as long as it settles at vertical.
5. Let an instructor feel your virtual spring.

Step 2: Render a Virtual Wall

Follow the instructions in the lab handout to add appropriate code to the function `hapkitRenderVirtualWall`

Let an instructor feel your virtual wall.

If you have time at the end, try some other methods, like making the wall feel stiffer by adding vibrations or damping upon contact.

Step 3: Render a Virtual Damper

Now we will measure the user's velocity and output a force in order to render a virtual damper.

1. Notice that we compute the velocity of the handle via a second-order filter in the file haplink_position.cpp:
   ```
   dxH = 0.9*((xH-xH_prev)/0.000001) + 0.1*dxH_prev;
   ```

2. Program a linear damping into your Hapkit code, in the function `hapkitRenderLinearDamping`.

3. Start with a small damping coefficient and gradually increase until you feel something.

4. Let the instructor feel your virtual damper.

Step 4: Render a Virtual Texture

Follow the instructions in the lab handout to add appropriate code to the function `hapkitRenderTexture`.

Let the instructor feel your virtual texture.

If you have time, try some other methods.
There are a lot of ways to do this, so be creative!

Step 5: Render Something Else!

???