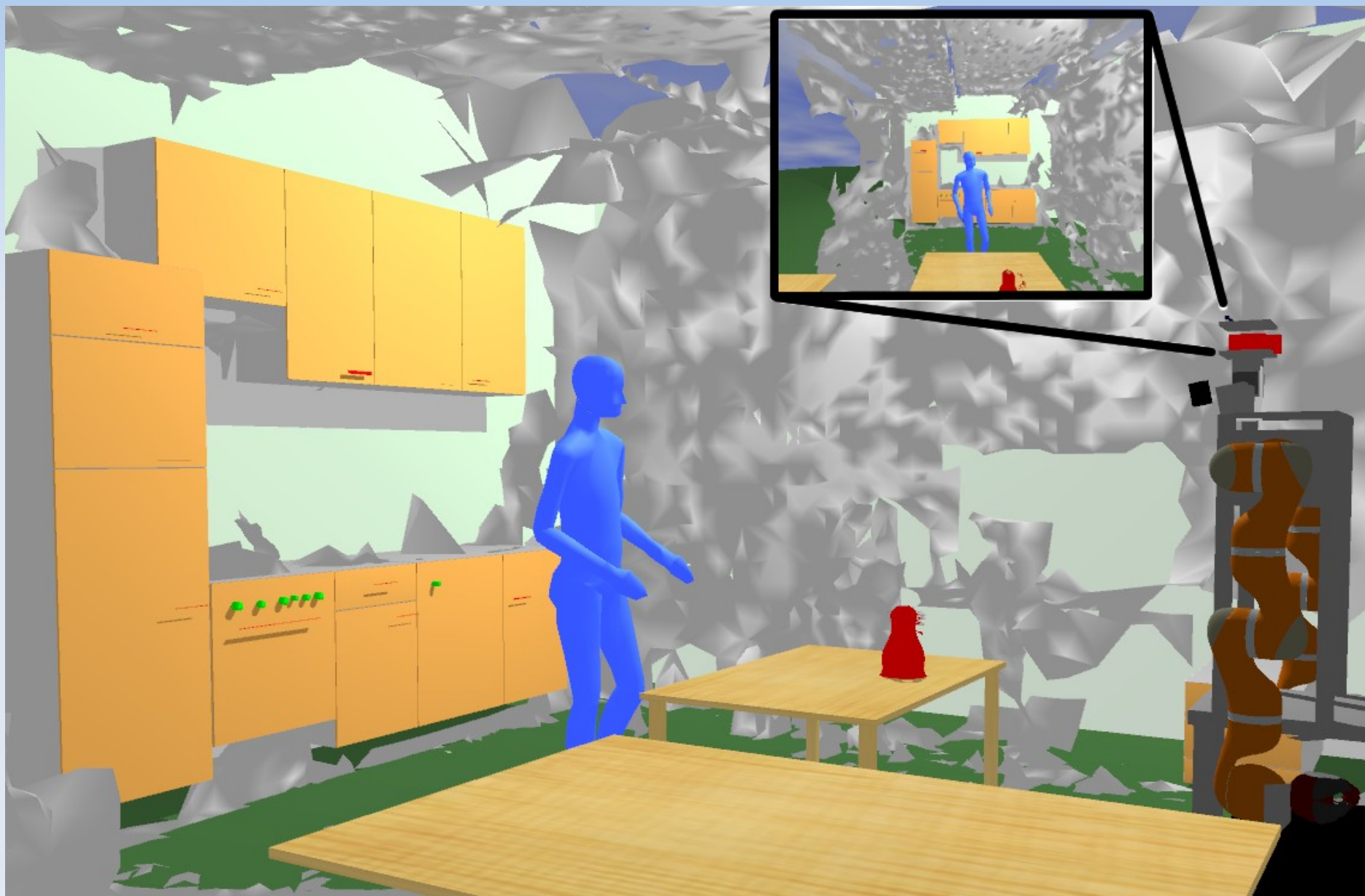


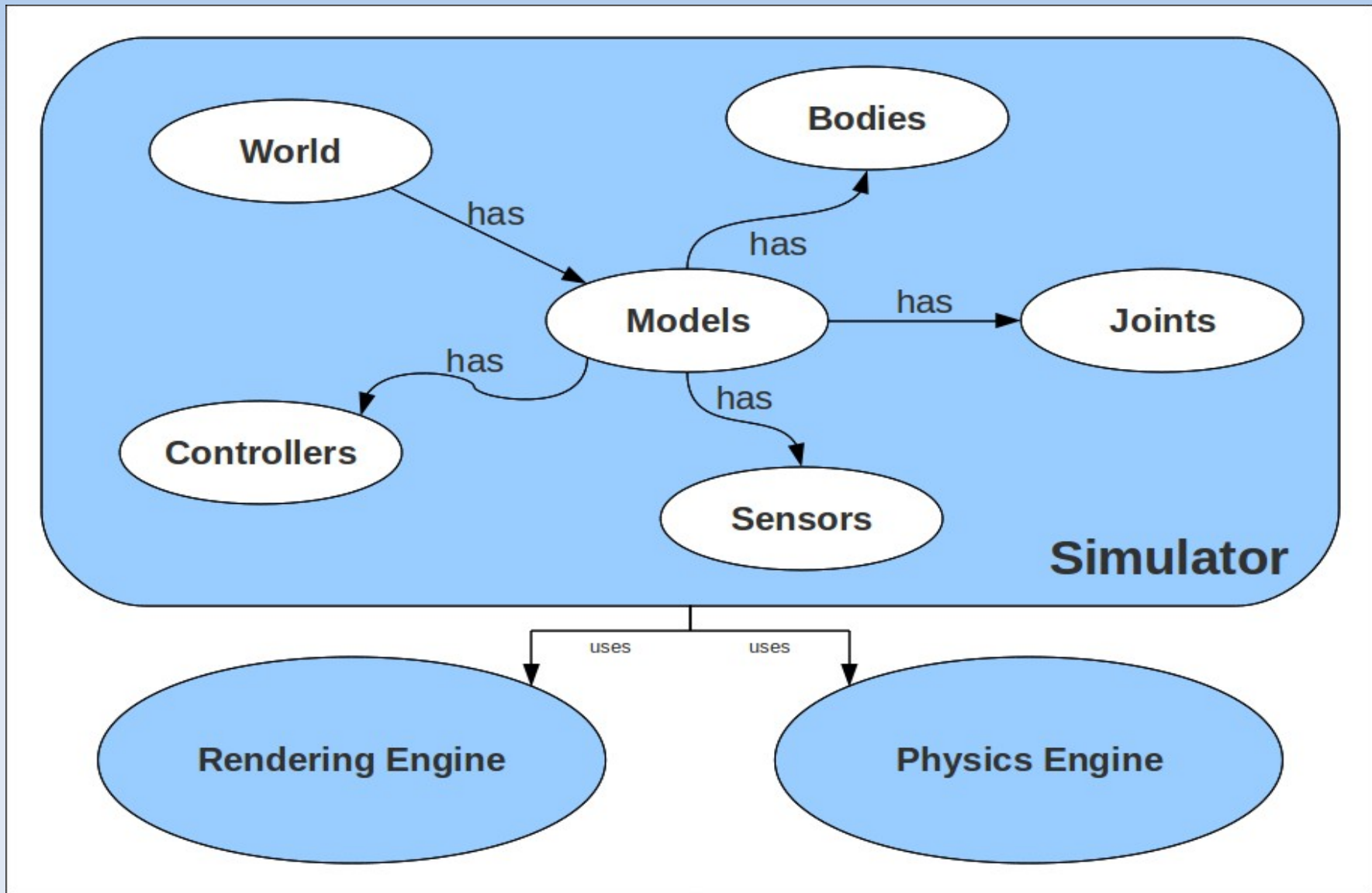
# Robotics simulation



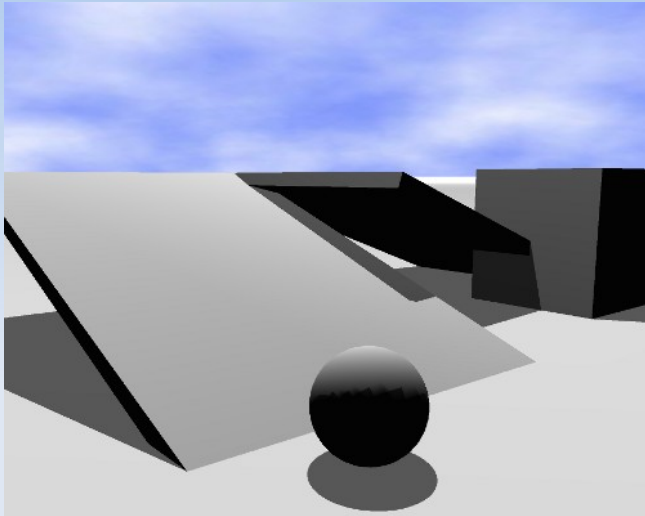
# Motivations

- Development tool
  - Interactive research development
  - Design, testing, optimization
  - Visualization
- System component
  - Knowledge source
  - Test-bed

# Overview

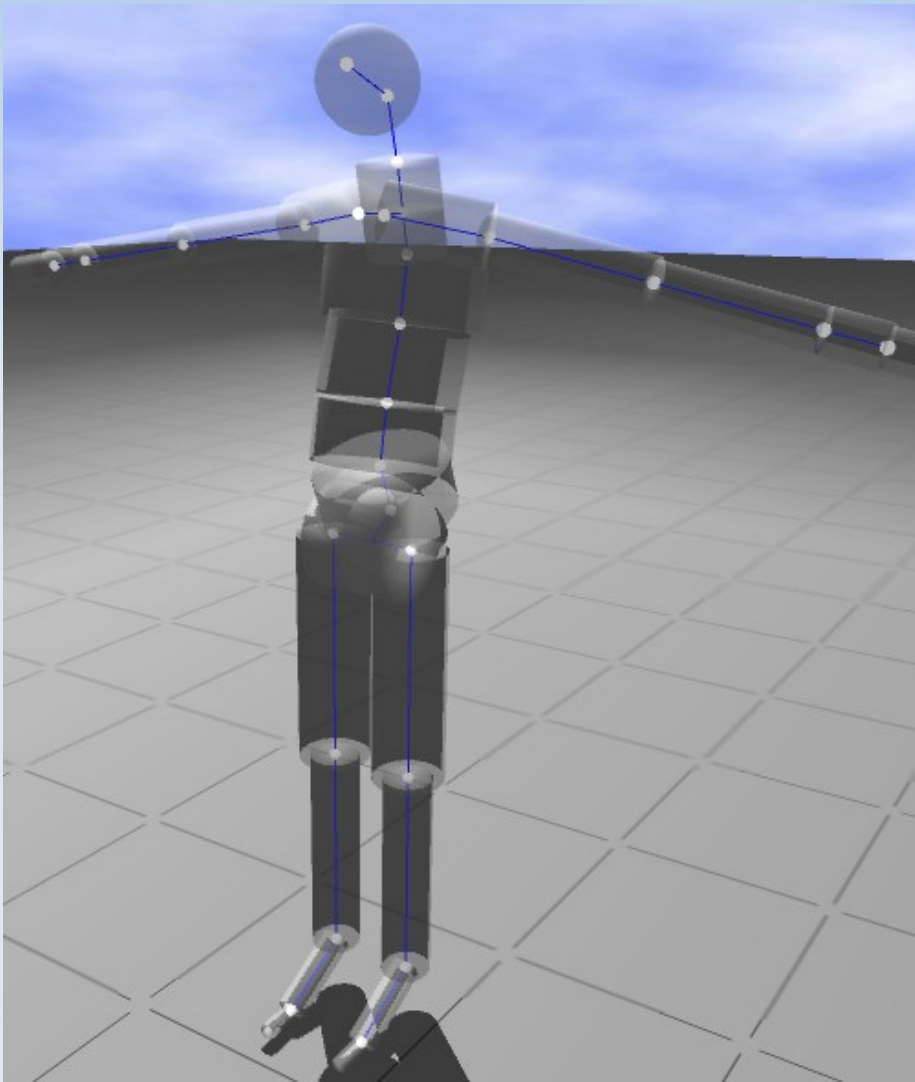


# Rendering Engines



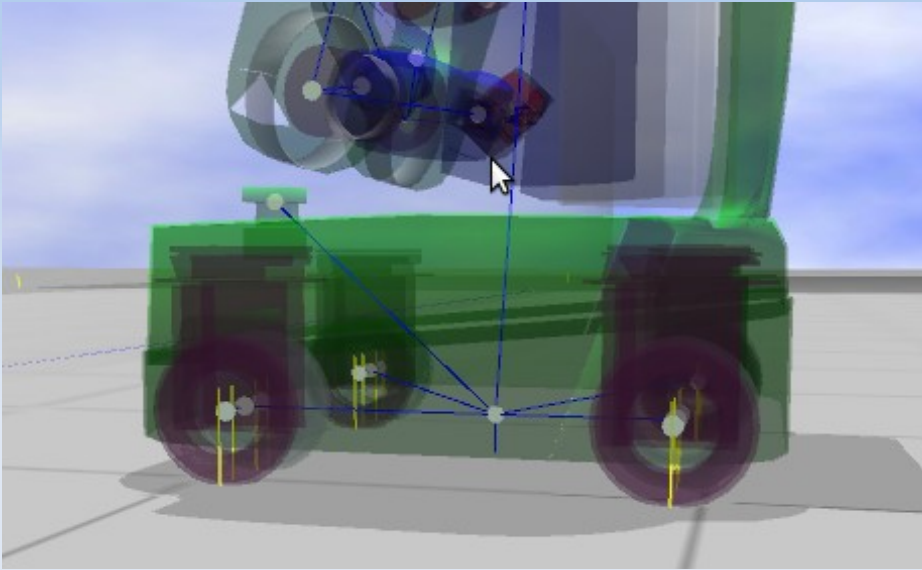
- Examples
  - Ogre3D
  - OpenGL
  - etc.
- Visualization
  - Sensors: camera, lasers
  - not always used

# Physics Engine



- Examples
  - ODE
  - Bullet
  - PhysX
- Rigid body, Soft body, Fluid simulation
- Components
  - Collision detection
  - Dynamics simulation

# Collision detection



- Two step process:
  - Broad phase (AABBs)
  - Narrow phase (collision geoms)

- 2 strategies
  - Discrete (a posteriori)
    - correction step, instability
  - Continuous (a priori)
    - more complex

# Collision response

- Discrete
  - Interpenetration error correction:
    - Force proportional to penetration depth
    - Stability problems
- Continuous
  - Use TOI (time-of-impact) predictions to avoid penetrations

# Dynamics simulation

- Orientation representation
  - Euler angles, quaternions, rotation matrices, axis angle
- Frame transformation
  - positions, velocities, efforts
- Newton's laws
- Constraints

# Dynamic simulation

- Particle model
  - Infinitesimal sized objects, finite mass
  - Newton's second law

$$\vec{F} = m \vec{a}$$

- Extended to finite sized, finite mass bodies
  - Provides linear momentum equations (COM)
- The simplest model
- No rotational motion of bodies

# Dynamics simulation

- Inertial model
  - Providing angular momentum of bodies
  - Models changes in position and orientation
  - Based on:

$$\frac{d(I\omega)}{dt} = \sum_{j=1}^n \tau_j$$

- $I$  = inertia tensor
- $\omega$  = angular velocity
- $\tau$  = torque

# Dynamics simulation

- Results in 6 ordinary differential equations
- Newton's second law for motion without constraints:

$$\begin{bmatrix} f_x \\ f_y \\ f_z \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} m_x & 0 & 0 & 0 & 0 & 0 \\ 0 & m_y & 0 & 0 & 0 & 0 \\ 0 & 0 & m_z & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{xx} & I_{xy} & I_{xz} \\ 0 & 0 & 0 & I_{yx} & I_{yy} & I_{yz} \\ 0 & 0 & 0 & I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \begin{bmatrix} \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \\ \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix}$$

# Dynamics simulation

- Usually solved by iterative methods
  - Jacobi
  - Gauss-Seidel
- Alternatives:
  - Unconstrained dynamics: penalty methods, forcing function based on constraint violation
  - Hybrid simulations

# Gazebo

- Major components
  - World
  - Bodies
    - Geometries
      - Contact geometry
      - Visual geometry
  - Joints
  - Controllers
  - Sensors

# Gazebo World

- XML description of the simulated scenario

```
<gazebo:world>
  <physics:ode>
    <stepTime>0.001</stepTime>
    <gravity>0 0 -9.8</gravity>
    <cfm>10e-10</cfm>
    <erp>0.2</erp>
  </physics:ode>

  <rendering:gui>
    <type>ftk</type>
    <size>800 600</size>
    <pos>0 0</pos>
  </rendering:gui>

  <rendering:ogre>
    <ambient>0 0 0 0</ambient>
    <sky>
      <material>Gazebo/CloudySky</material>
    </sky>
    <shadowTechnique>stencilModulative</shadowTechnique>
    <grid>>false</grid>
  </rendering:ogre>
</gazebo:world>
```

# Gazebo Model

```
<model:physical name="plane1_model">
  <xyz>0 0 0</xyz>
  <rpy>0 0 0</rpy>
  <static>>true</static>

  <body:box name="box1_body">
    -----
  </body:box>
  -----
  <joint:hinge name="joint1">
    -----
  </joint:hinge>
  -----
  <controller:motor_contr name="contr1">
    -----
  </controller:motro_contr>
  -----
  <sensor:camera name="camera1">
    -----
  </sensor:camera>
  -----
</model:physical>
```

# Gazebo Bodies

```
<body:box name="body1">  
  <xyz>0 0 1</xyz>  
  <rpy>0 90 0</rpy>  
  
  <geom:box name="geom1">  
    -----  
  </geom:box>  
  
  -----  
  
  <geom:box name="geom_n">  
    -----  
  <geom:box>  
  
</body:box>
```

# Gazebo geoms

```
<geom:box name="box1_geom1">  
  <size>0.5 0.5 0.5</size>  
  <mass>0.1</mass>  
  
  <visual>  
    <size>0.5 0.5 0.5</size>  
    <mesh>unit_box</mesh>  
    <material>Gazebo/Rocky</material>  
  </visual>  
  
</geom:box>
```

# Gazebo Joints

```
<joint:hinge name="left_wheel_hinge">  
  <body1>left_wheel</body1>  
  <body2>chassis_body</body2>  
  <anchor>left_wheel</anchor>  
  <anchorOffset>0 0.04 0</anchorOffset>  
  <axis>0 1 0</axis>  
  <erp>0.8</erp>  
  <cfm>10e-5</cfm>  
</joint:hinge>
```

# Gazebo Controllers

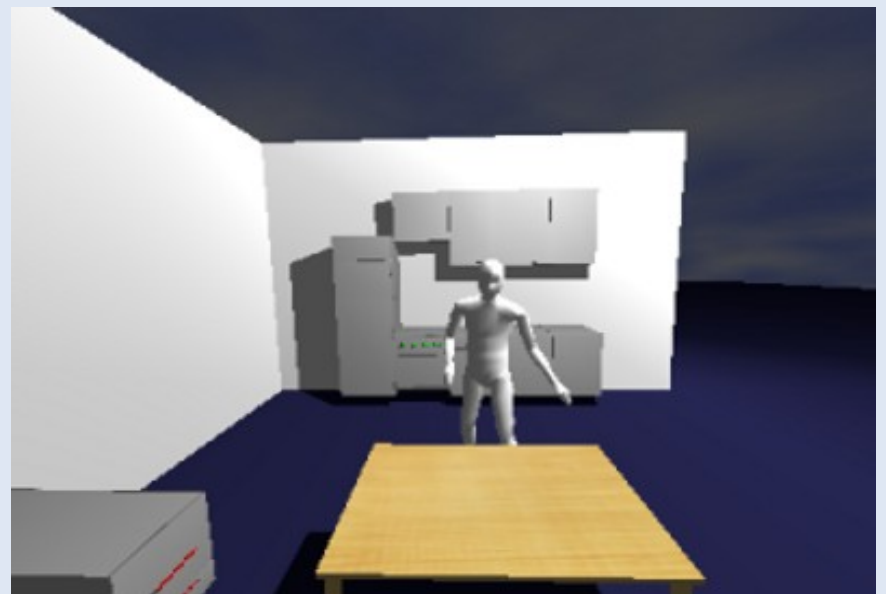
```
<controller:ias_human_controller
name="human_model_controller"
plugin="libias_human_controller.so">
  <alwaysOn>true</alwaysOn>
  <updateRate>100</updateRate>
  <!-- specific controller params
  -----
  -->
</controller:ias_human_controller>
```

# Gazebo sensors

```
<sensor:ray name="laser_1">  
  <rayCount>180</rayCount>  
  <rangeCount>361</rangeCount>  
  <origin>0.05 0.0 0</origin>  
  
  <displayRays>fan</displayRays>  
  
  <minAngle>-90</minAngle>  
  <maxAngle>90</maxAngle>  
  
  <minRange>0.1</minRange>  
  <maxRange>8</maxRange>  
  <resRange>.1</resRange>  
  
  <controller:sicklms200_laser name="laser_controller_1">  
    <interface:laser name="laser_iface_0"/>  
    <interface:fiducial name="fiducial_iface_0"/>  
  </controller:sicklms200_laser>  
</sensor:ray>
```

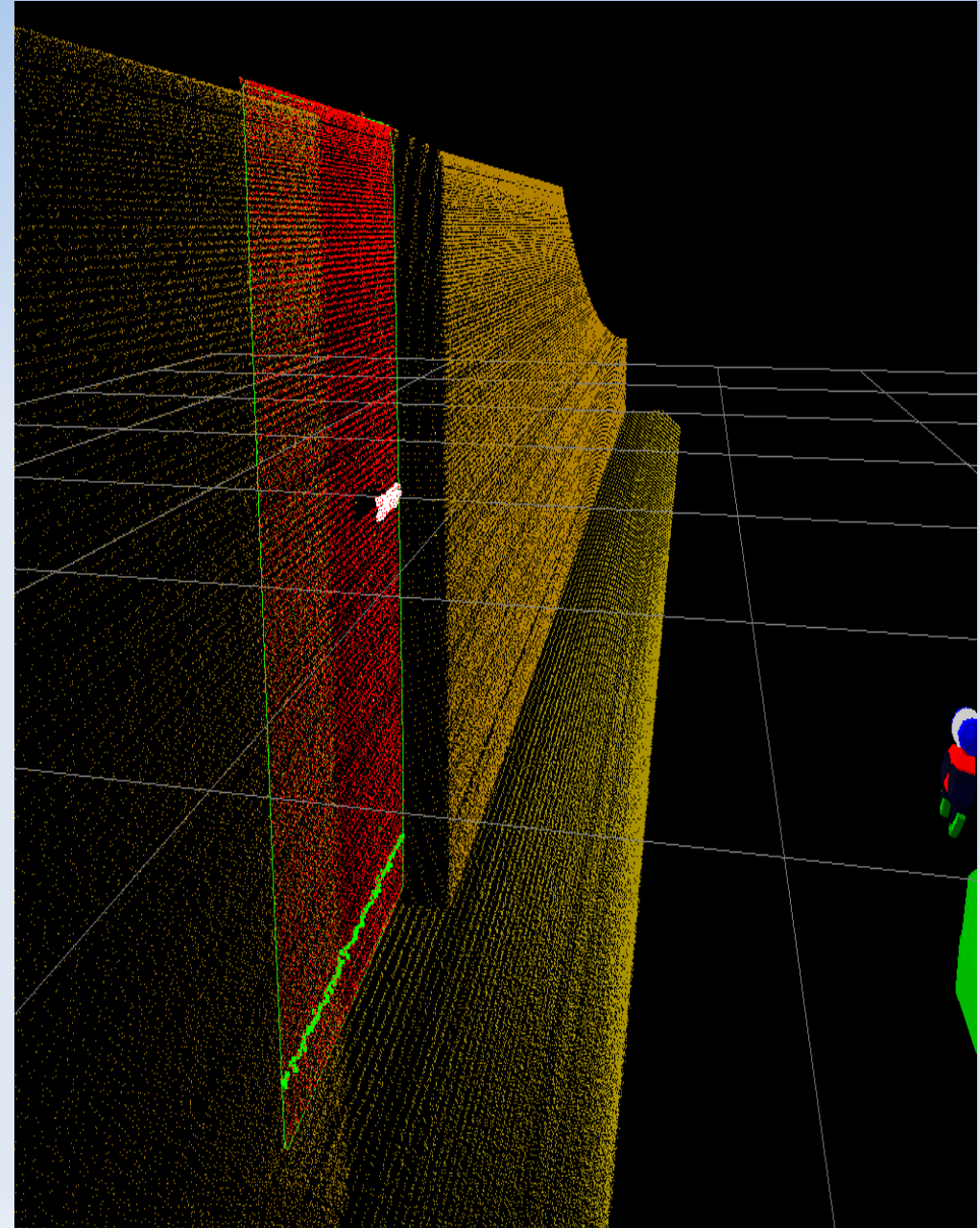
# Camera sensors

- Get the image from the rendering engine
- Ideal pin hole cameras
- No distortions
- Noise-free
- In focus
- No need to calibrate



# Laser range sensors

- Raycasting
- Physics engine based
- High resolution (no. of rays) is expensive
- Plans to use rendering engine raycasting
  - Faster
  - More accurate
- No noise



# Other sensors

- Contact sensors
- Stereo camera sensor
- IMU's
- etc.

# Conclusions

- Contacts
  - Default inelastic collisions
  - Not always stable
- Contact friction
  - not accurate if resulting friction force is not aligned with the principle directions.
- Camera sensors
  - Not very realistic, even with textures
  - Shading is interpolated across meshes
- Ideal joint motors and encoders
  - Friction and efficiency not modeled

# Conclusions

