CSE 167 Fall 2019

Discussion 5

Project 3

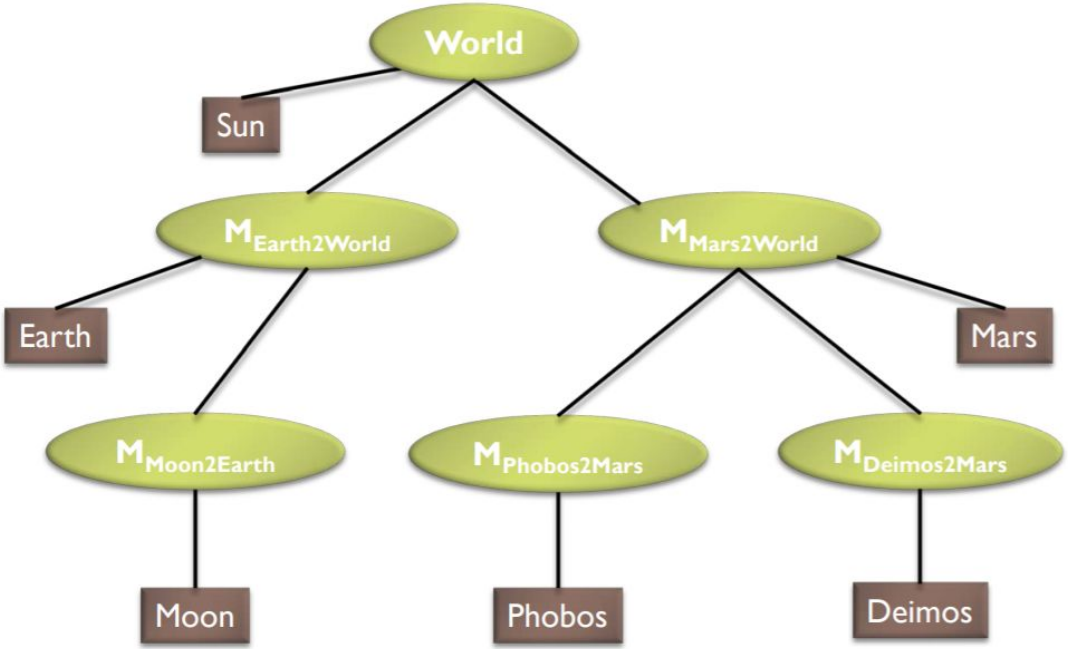
- Project specifications HERE
- DUE Friday Nov 1 at 2pm
  - CSE Basement 260/270
- Features to implement:
  - Scene Graph
  - Animated Robot
  - Robot Army
  - Culling

Overview

- Scene Graph
  - Review and Implementation
- Creating your Robot
- Animating Robot
- Creating Robot Army

Scene Graphs

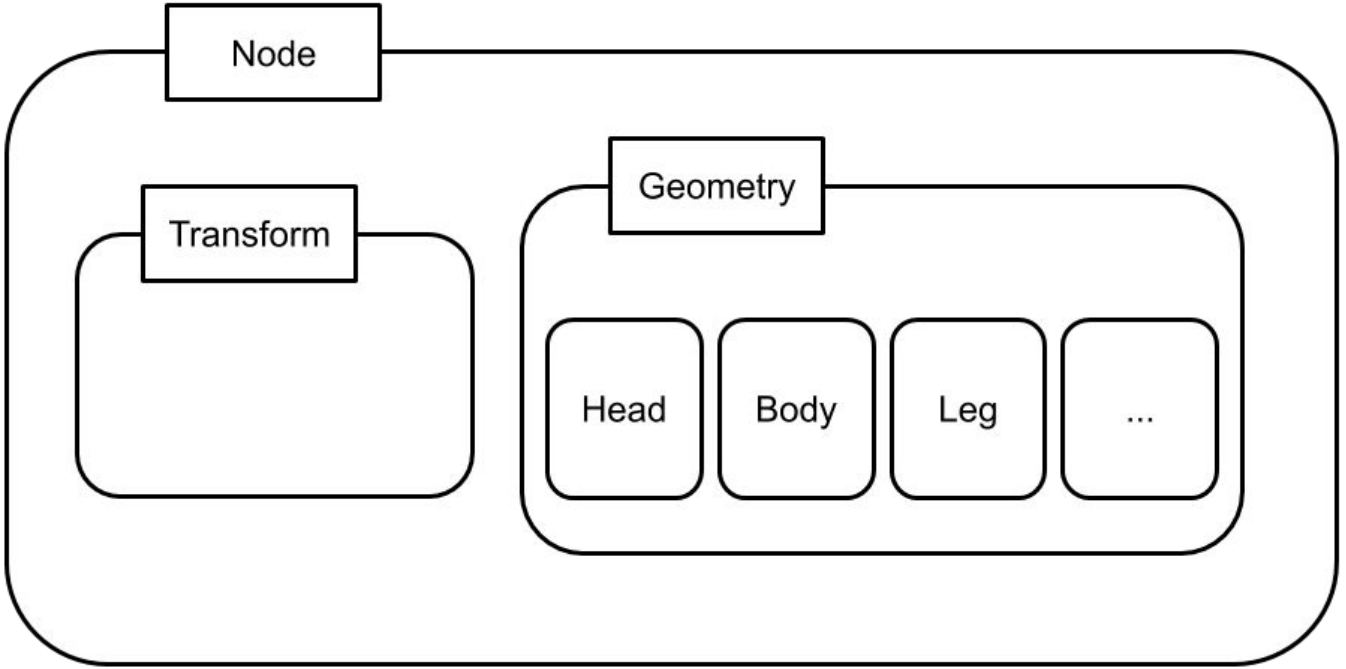
Scene Graph


World
Sun
M Earth2World
M Mars2World
Earth
Mars
M Moon2Earth
M Phobos2Mars
M Deimos2Mars
Moon
Phobos
Deimos

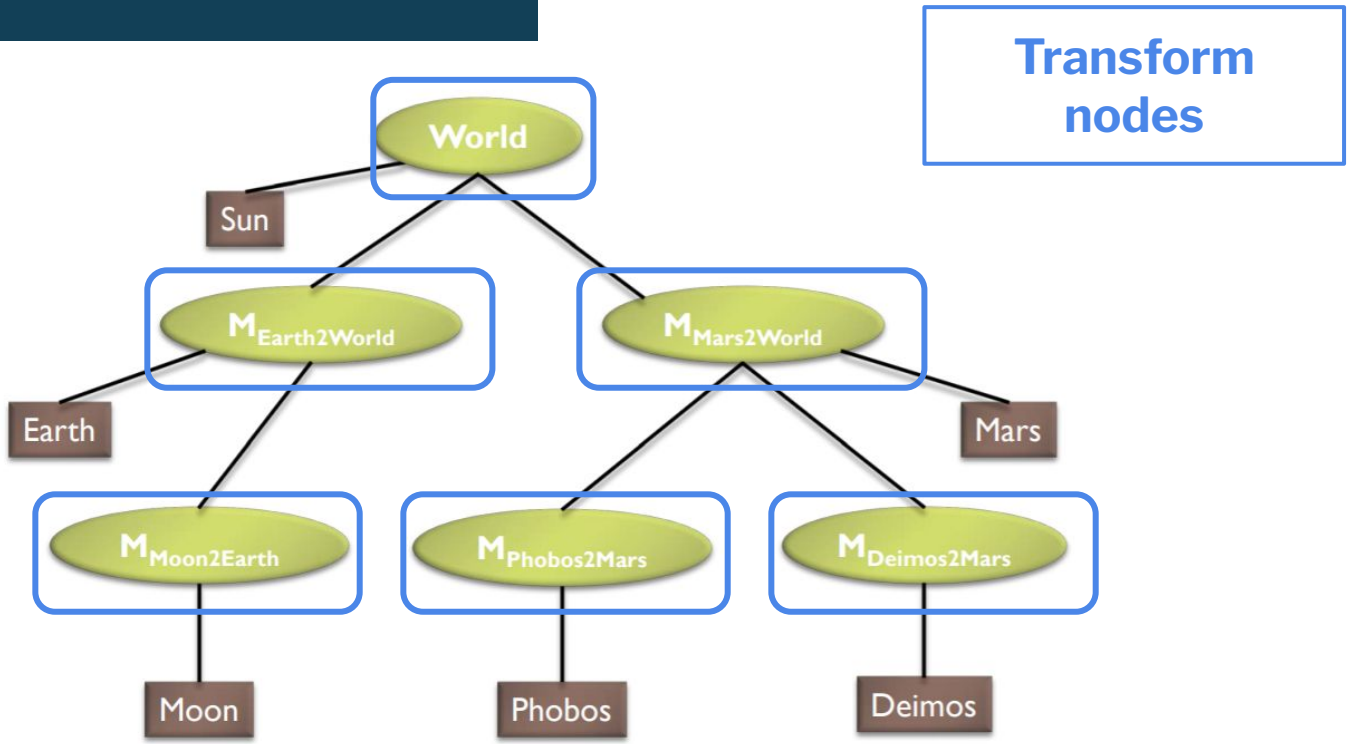
Scene Graph

- Need 3 classes:
  - Node class
    - Base class with a virtual void draw and update functions
  - Transform class
    - Responsible for transformations
  - Geometry class
    - Similar to your PointCloud class
    - Responsible for drawing the objects
- Will create either a Transform or Geometry type object

Scene Graph
Hierarchy

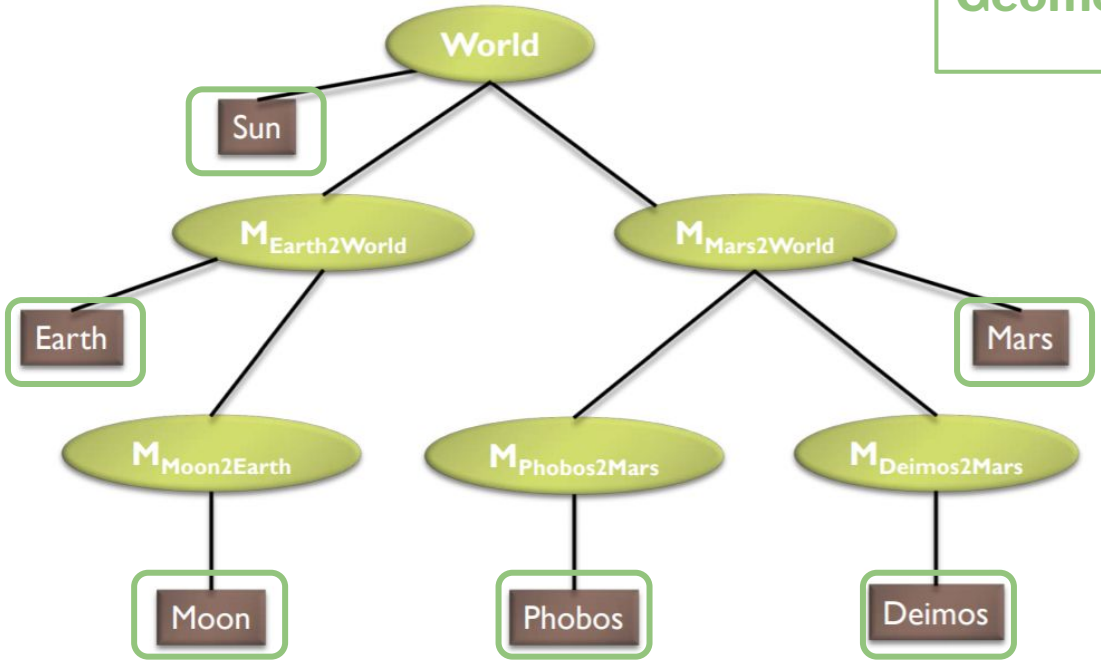


Scene Graph


Transform nodes
World
Sun
M Earth2World
M Mars2World
Earth
Mars
M Moon2Earth
M Phobos2Mars
M Deimos2Mars
Moon
Phobos
Deimos

Scene Graph

Geometry nodes



Node Class

- Abstract base class
  - Need to set up the functions that you want both Geometry and Transform classes to have

```
class Node {
public:
    virtual void draw(GLuint shaderProgram, glm::mat4 C) = 0;
    virtual void update(glm::mat4 C) = 0;
};
```

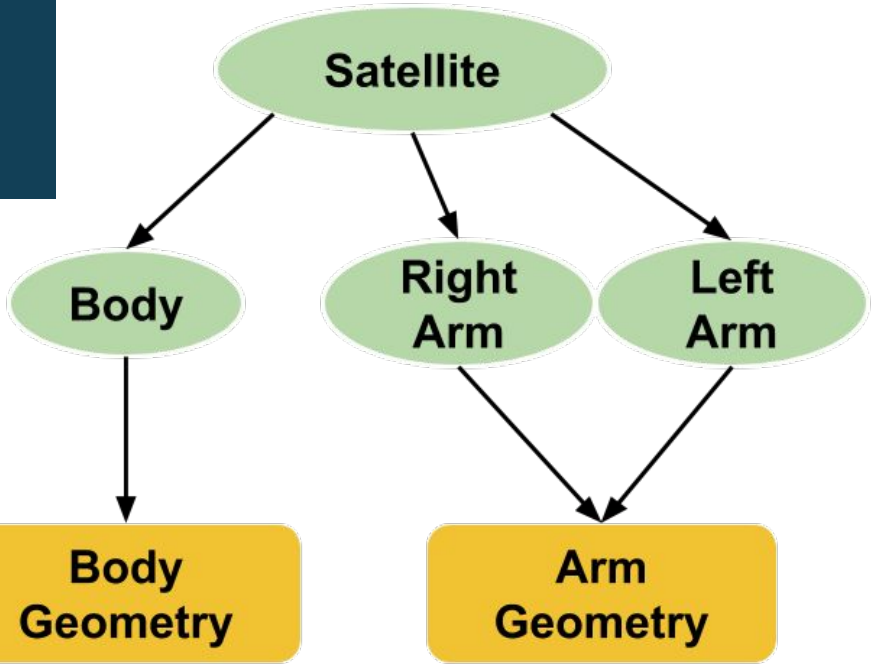
Transform Class

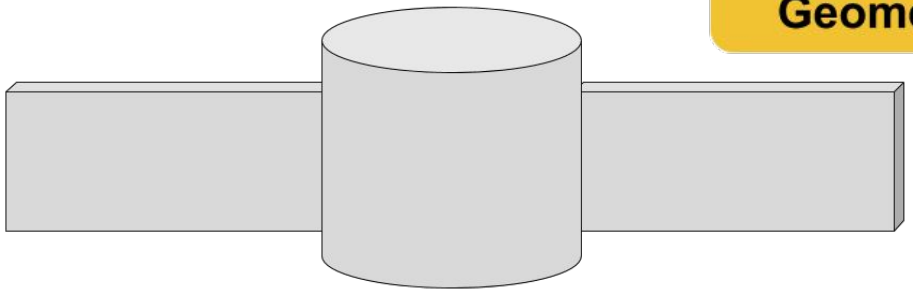
- Derive from Node class
- Functions:
  - draw & update (b/c inheriting from Node)
  - addChild
- Member variables:
  - Transform matrix
    - Matrix that places object relative to parent
  - List of child Nodes

Geometry Class

- Derive from Node class
- Can take straight from PointCloud.cpp
- Functions:
  - draw & update (b/c inheriting from Node)
  - Load, parse… any helper functions you may have had
- Member Variables:
  - model
  - VAO, VBO(s), EBO…
  - Points, normals, indices…

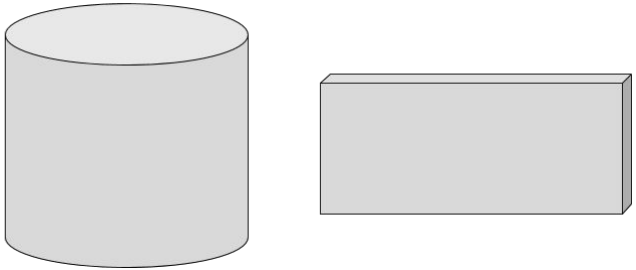
Scene Graph


Satellite
Body
Right Arm
Left Arm
Body Geometry
Arm Geometry



Scene Graph
Building

```
ArmGeo = new Geometry("arm.obj")
BodyGeo = new Geometry("body.obj")
```

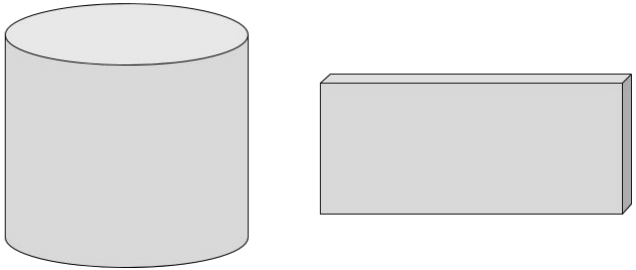


Body Geometry

Arm Geometry

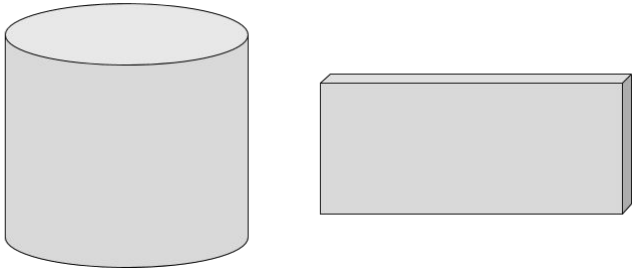
Scene Graph
Building

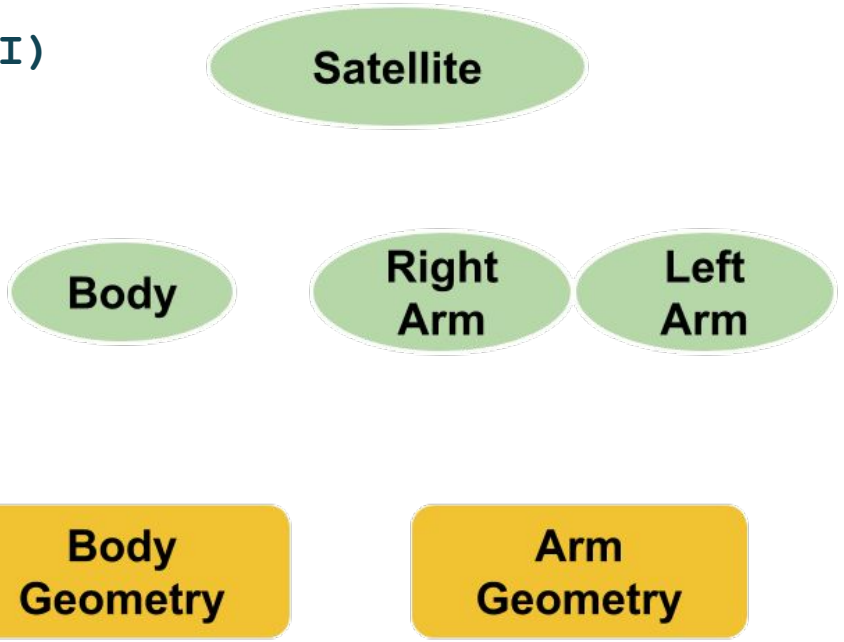
```
Satellite = new Transform(I)
```



Scene Graph
Building

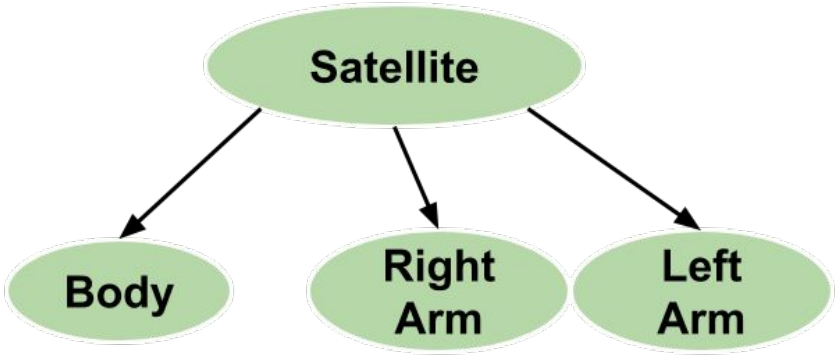
```
Satellite = new Transform(I)
Body = new Transform(T1)
Rarm = new Transform(T2)
Larm = new Transform(T3)
```

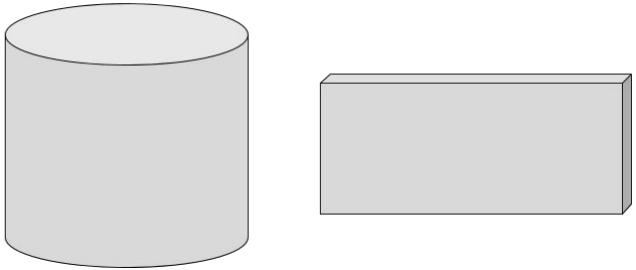




Scene Graph
Building

```
Satellite.addChild(Body)
Satellite.addChild(Rarm)
Satellite.addChild(Larm)
```



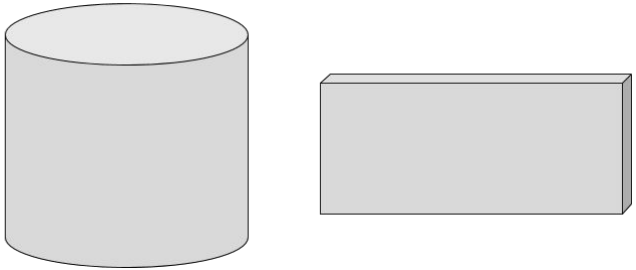


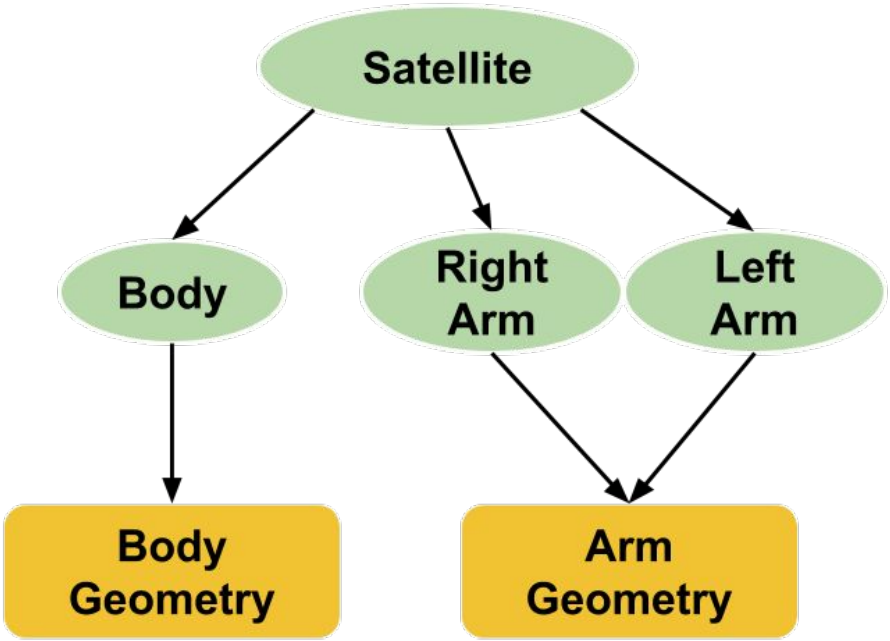
Body Geometry

Arm Geometry

Scene Graph
Building

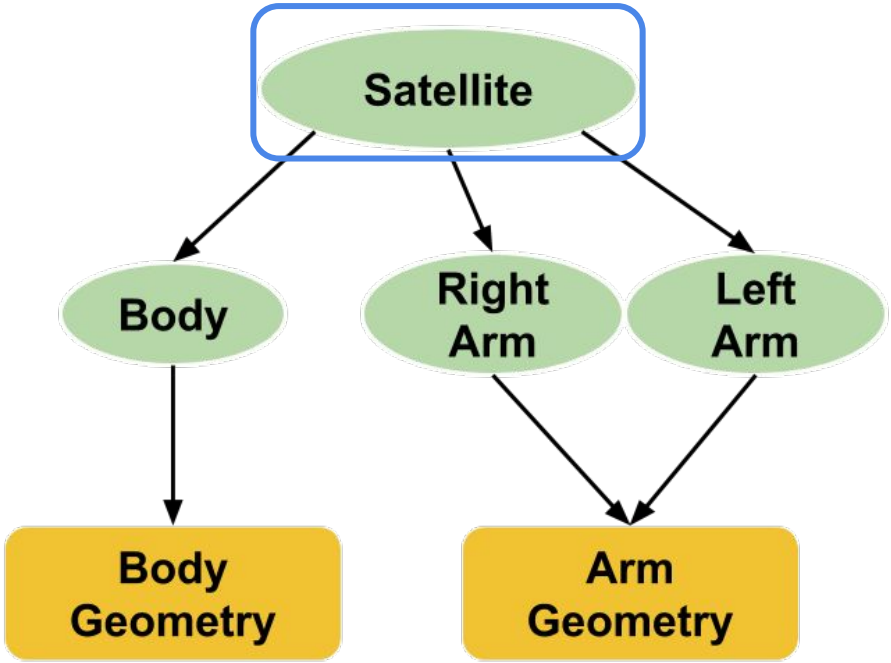
```
Body.addChild(BodyGeo)
Rarm.addChild(armGeo)
Larm.addChild(armGeo)
```





Scene Graph
Drawing

```
Satellite->draw()
```

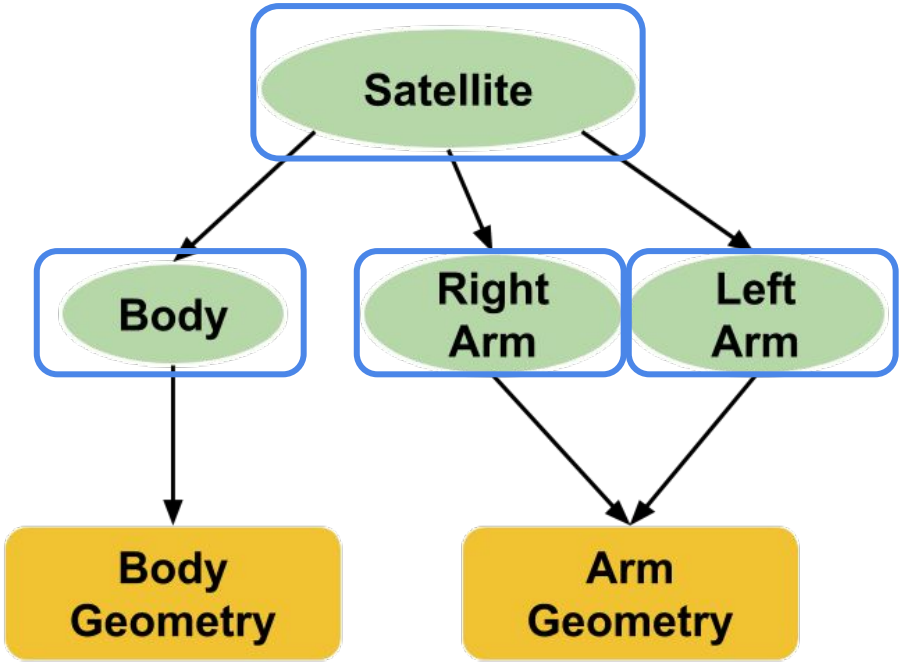


Scene Graph
Drawing

- Job of Transform's draw call is to make sure that all its children get drawn
  - Loop through all kids
  - Call draw on all kids

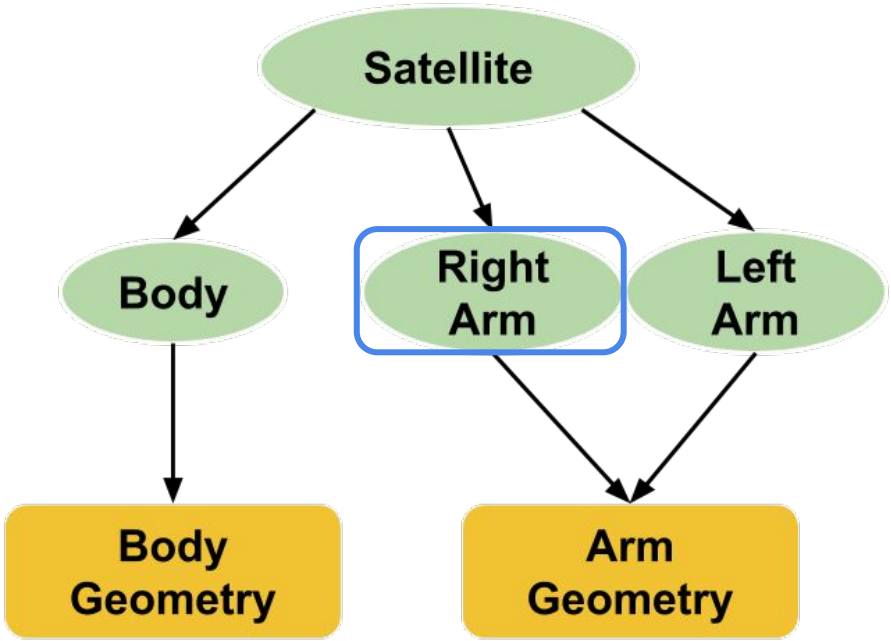
Scene Graph
Drawing

```
Satellite->draw()
⇒ Body->draw()
⇒ Rarm->draw()
⇒ Larm->draw()
```



Scene Graph
Drawing

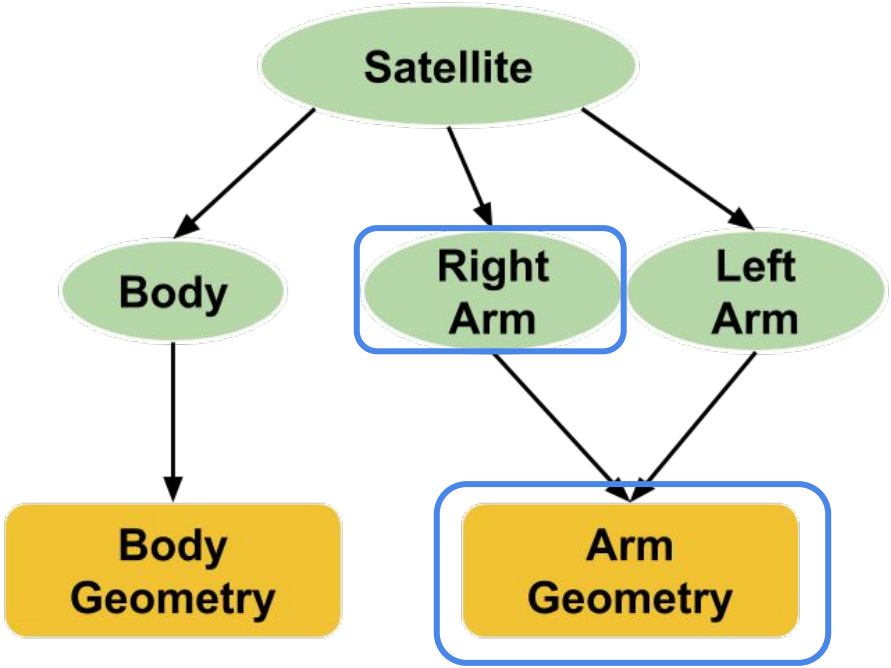
```
Rarm->draw()
```



Scene Graph
Drawing

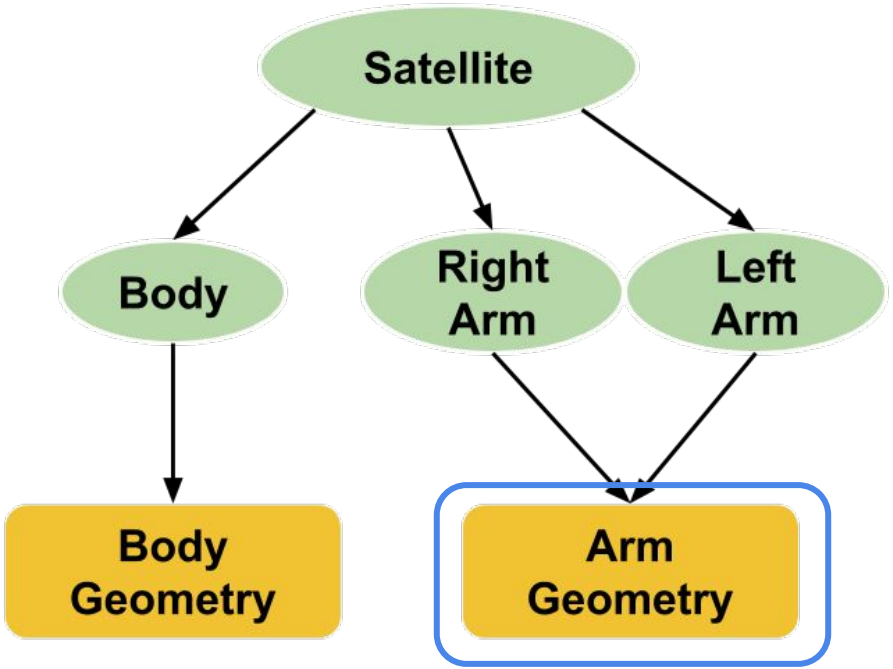
```
Rarm->draw()
```

⇒ `armGeo->draw()`



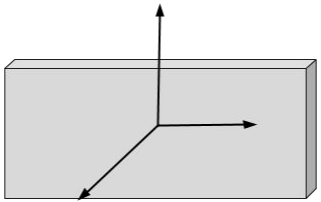
Scene Graph
Drawing

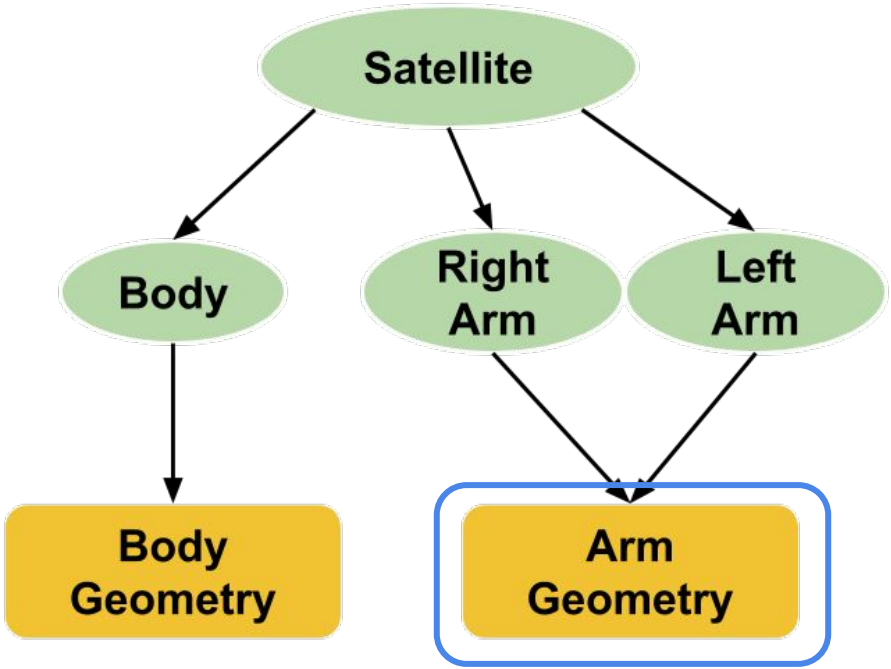
```
armGeo->draw()
```



Scene Graph
Drawing

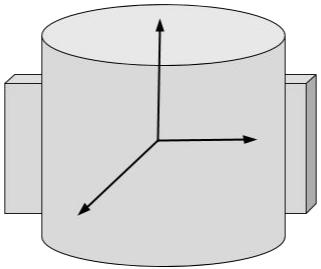
```
armGeo->draw()
=> glDrawElements(...)
```

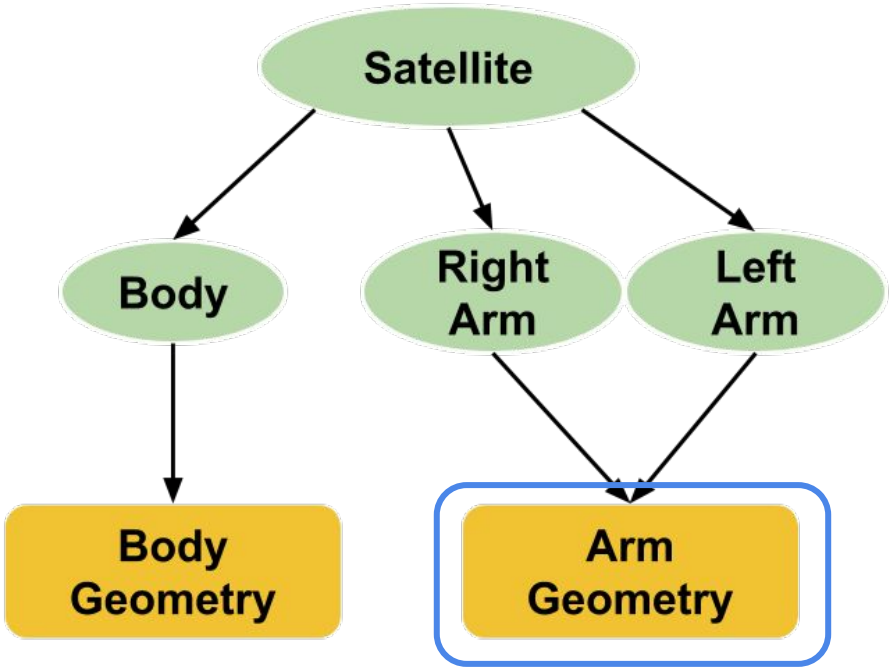




Scene Graph
Drawing

```
armGeo->draw()
=> glDrawElements(...)
```



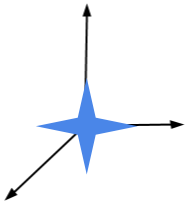


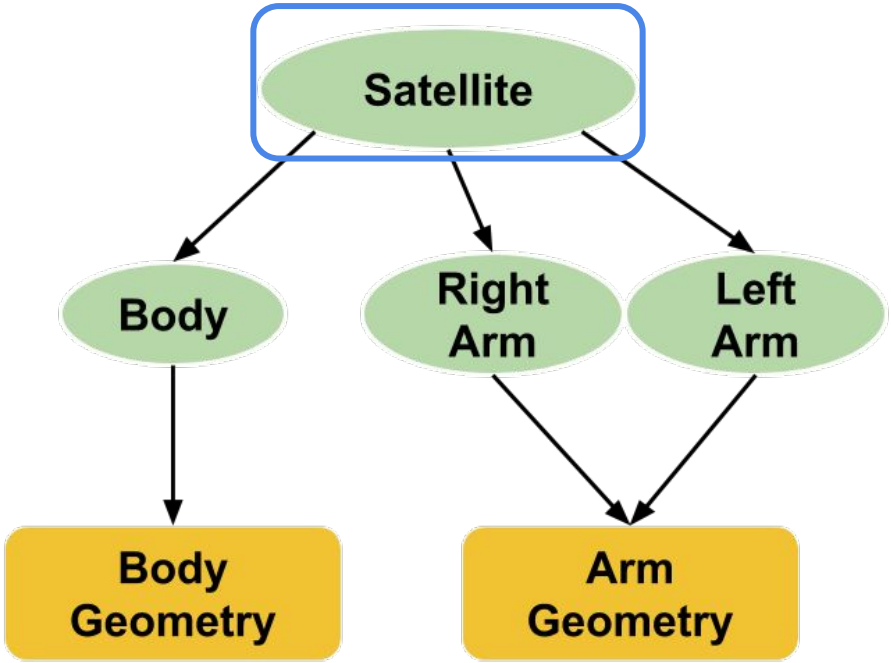
Scene Graph
Drawing

- Job of Transform's draw call is to make sure that all its children get drawn **in the correct position**
  - Loop through all kids
  - Call draw on all kids
- Need to make sure pass along your transform so the child knows where to go
  - Pass down an updated matrix in the draw function

Scene Graph
Drawing

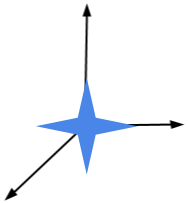
```
Satellite->draw(X)
```

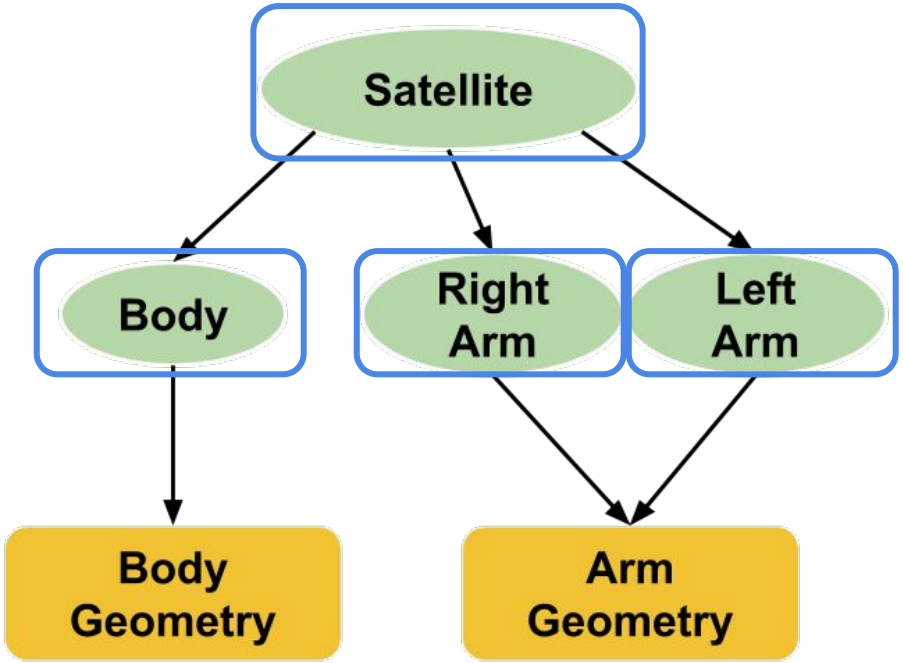




Scene Graph
Drawing

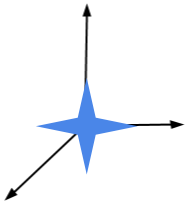
```
Satellite->draw(X)
⇒ Body->draw(X*I)
⇒ Rarm->draw(X*I)
⇒ Larm->draw(X*I)
```

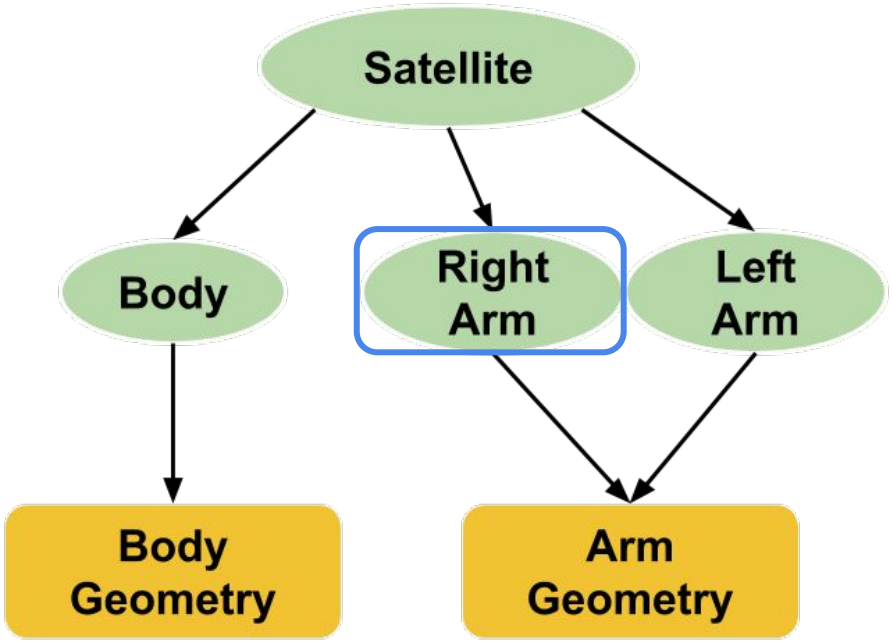




Scene Graph
Drawing

```
Rarm->draw(X*I)
```

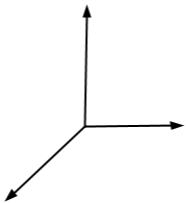


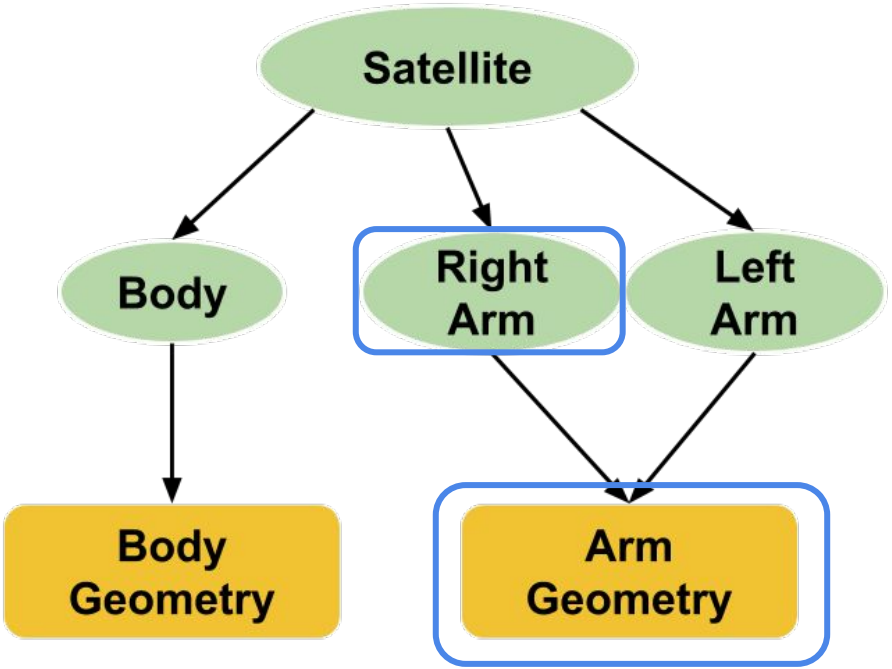


Scene Graph
Drawing

```
Rarm->draw(X*I)
```

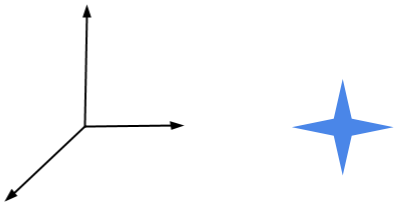
⇒ `armGeo->draw(X*I*T2)`

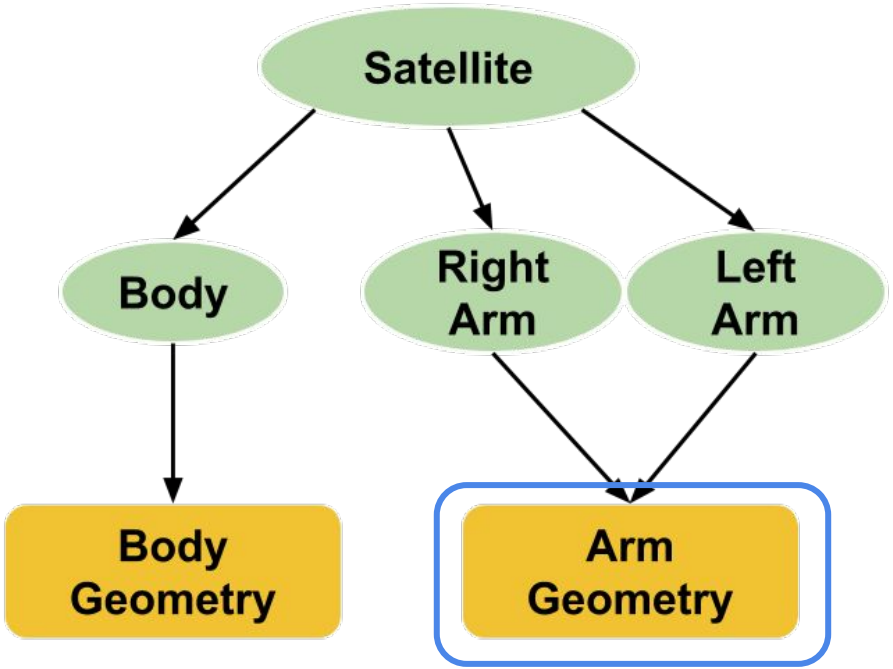




Scene Graph
Drawing

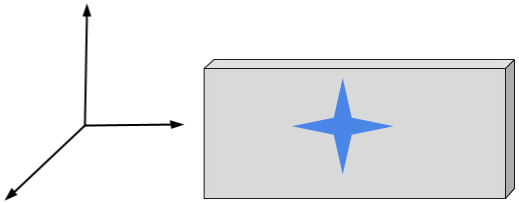
`armGeo->draw(X*I*T2)`

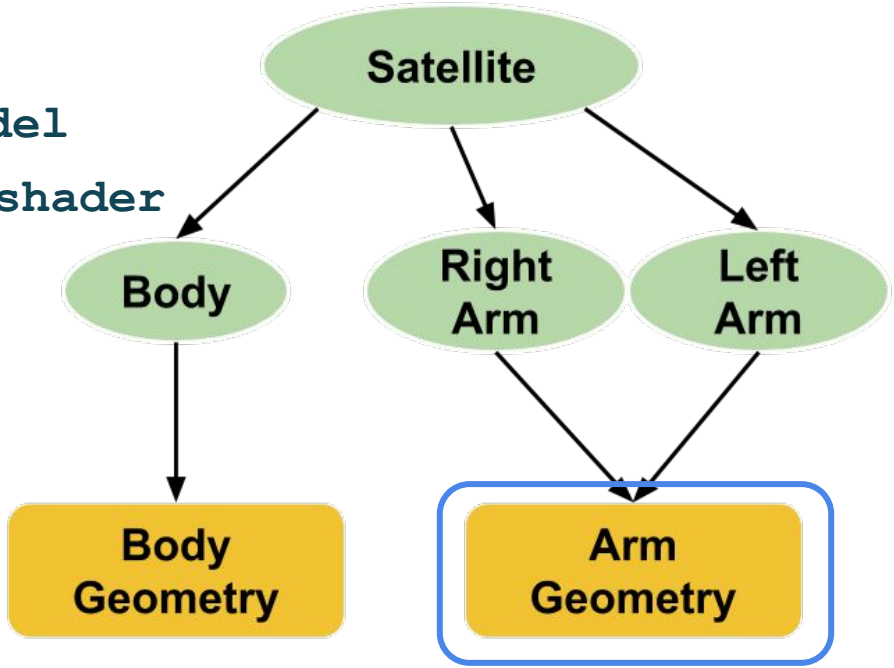




Scene Graph
Drawing

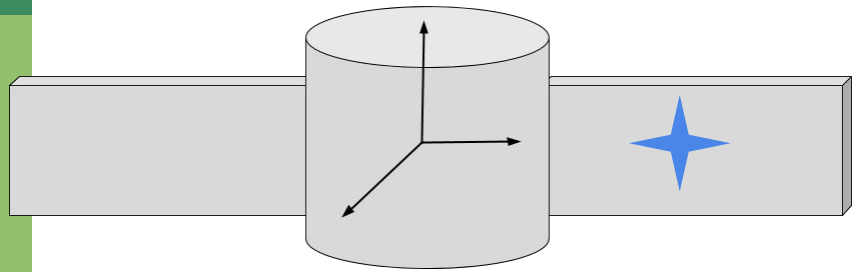
```
armGeo->draw(X*I*T2)
=> model = X*I*T2*initModel
=> send model matrix to shader
=> glDrawElements(...)
```

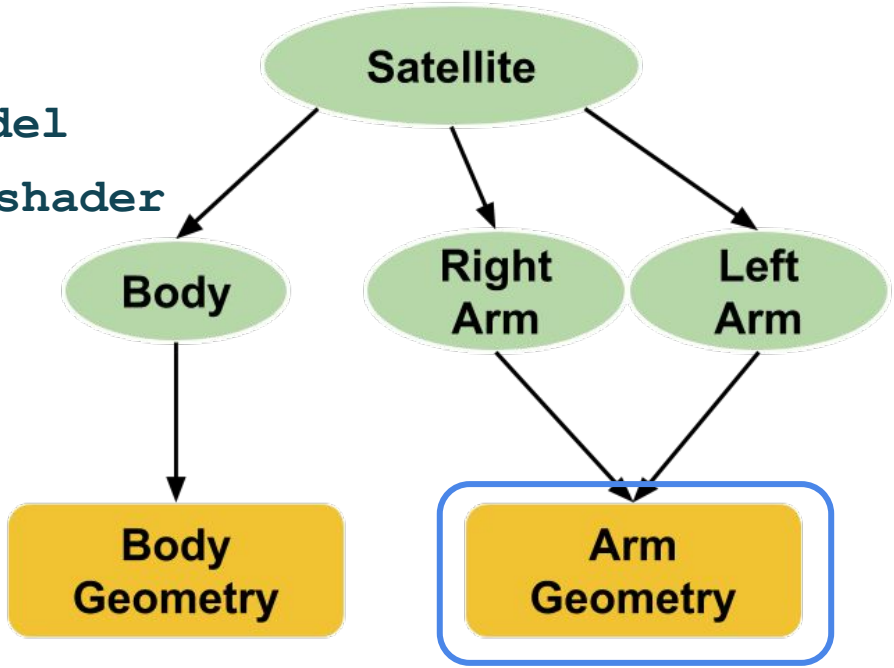




Scene Graph
Drawing

```
armGeo->draw(X*I*T2)
=> model = X*I*T2*initModel
=> send model matrix to shader
=> glDrawElements(...)
```





Scene Graph
Drawing

- Transform draw call:
  - Loop through children
  - Call draw on all kids, passing:
    - ShaderProgram
      - So can pass the model matrix to the shader
    - Matrix
      - So we know where to draw the object

Scene Graph
Drawing

- Geometry draw call:
  - Calculate toWorld matrix
    - Based on the passed in matrix and the geometry's initial model matrix
  - Send that toWorld matrix to the shader
  - glDrawElements(...)

Creating your Robots

Creating Robot

- Given Robot parts:
  - Head, body, limb, eye …
- Requirements:
  - Need to use at least 3 different body parts
  - 4 in total
  - 3 need to move independently from each other and be connected to the 4th part
  - Get Creative!

Creating Robot

- Create your Robot using
  - Geometry nodes to load the obj files
  - Transform nodes to place the parts and create your Robot

Rotating Camera

Rotating Camera

- Need to be able to move your camera
  - Modify trackball to rotate the camera instead
- In PA2:
  - Found the angle and axis of rotation with trackball mapping
  - Applied rotation matrix to the bunny

Rotating Camera

- For PA3:
  - Find the angle and axis of rotation with trackball mapping
  - Apply rotation to the Camera's direction Vector
  - Update the Camera Matrix (View matrix)
  - Send the matrix to the shader

```
// View matrix, defined by eye, center and up.
glm::mat4 Window::view = glm::lookAt(Window::eye, Window::center, Window::up);
```

Animating Robots

Animating Robot

- Need 3 limbs to move independent of each other
- Need to make your Robot walk
  - Walking in place is OK
- How?
  - Need to apply/update transformation nodes
  - Want cyclic motion for walking…
- Where?
  - With the rest of our update calls

Animating Robot

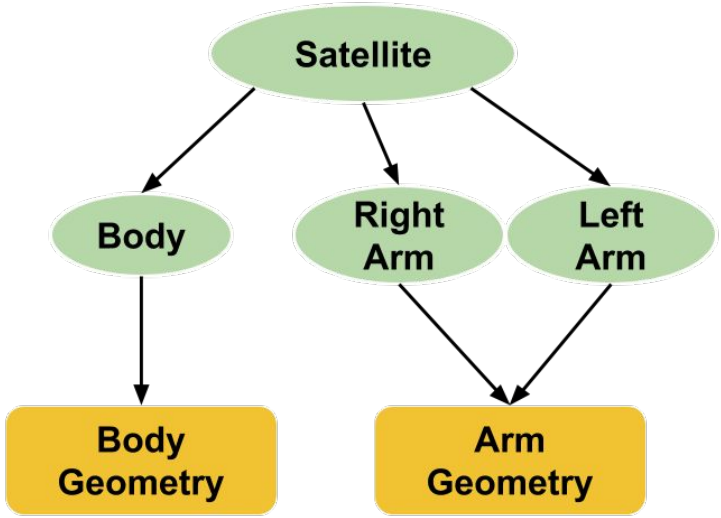
- Where?
  - initialize_objects()
    - Build Robot/Robot Army
  - display_callback()
    - Draw on root node (root->draw(...))
  - idle_callback()
    - Update calls (root->update(...))

Creating Robot Army

Robot Army

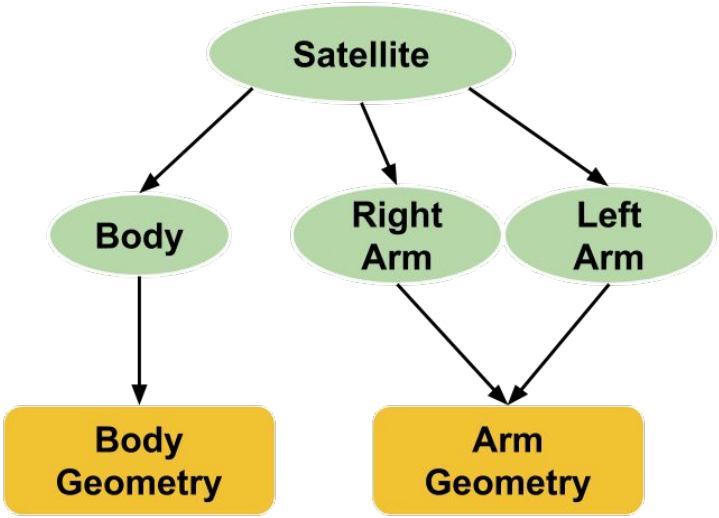
- Need at least 100 robots
  - Identical clones
  - Place in a 2D grid (10x10)
  - Make sure all of them are animated
- Camera Movement
  - Rotate the camera with trackball
  - Zoom in and out (with scrolling)

Robot Army


Satellite
Body
Right Arm
Left Arm
Body Geometry
Arm Geometry

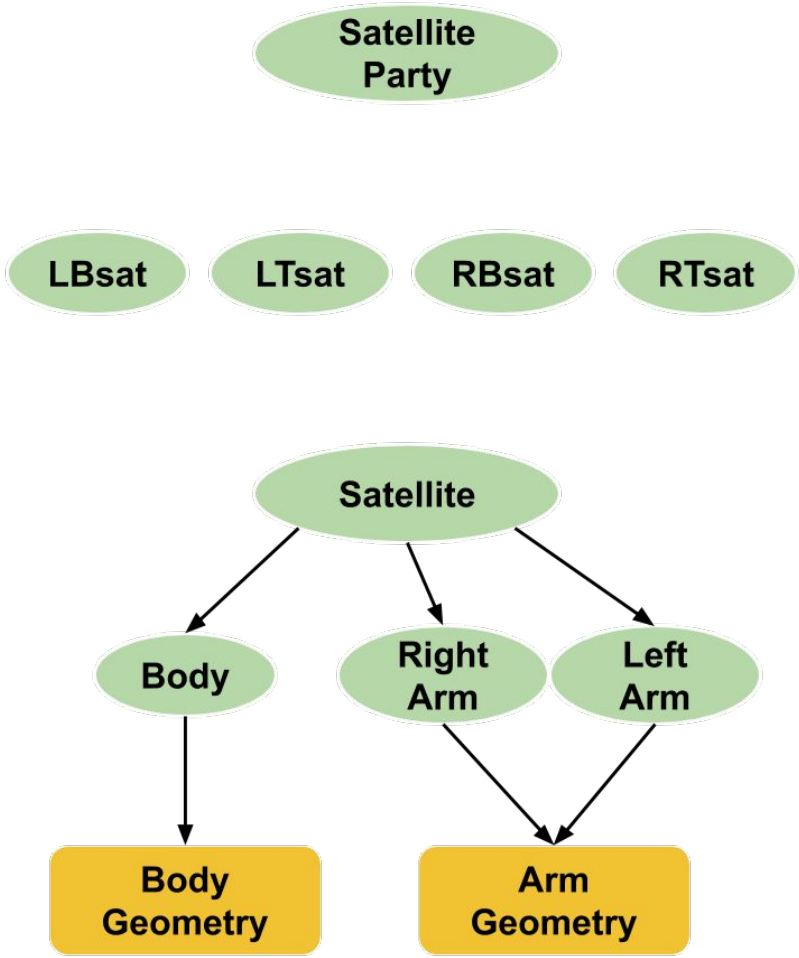
Robot Army


```
SatelliteParty = new Transform(I)
```

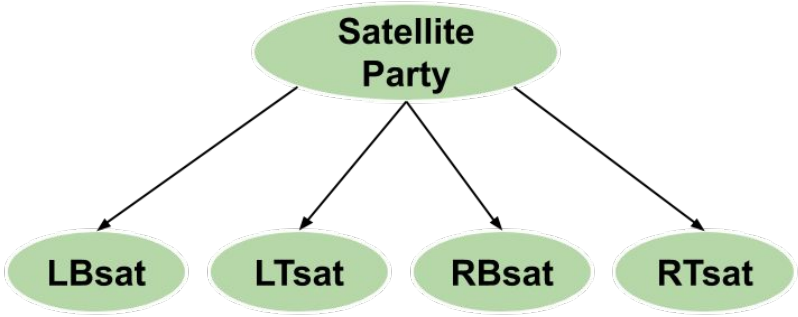


Robot Army

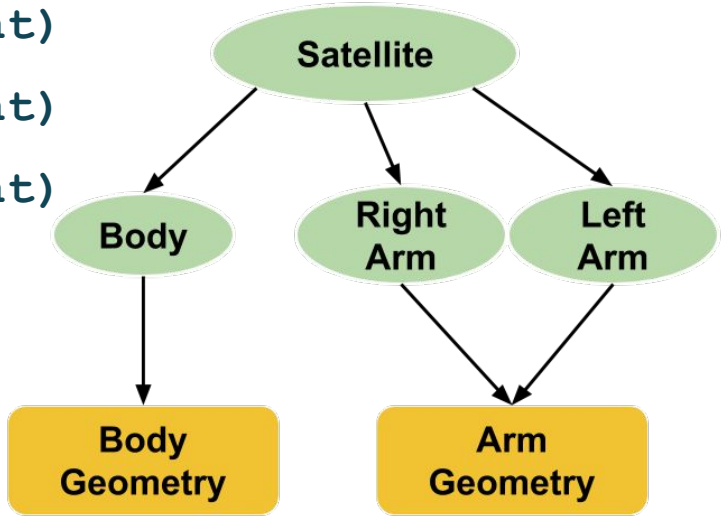
```
LBsat = new Transform(T'1)
LTsat = new Transform(T'2)
RBsat = new Transform(T'3)
RTsat = new Transform(T'4)
```



Robot Army

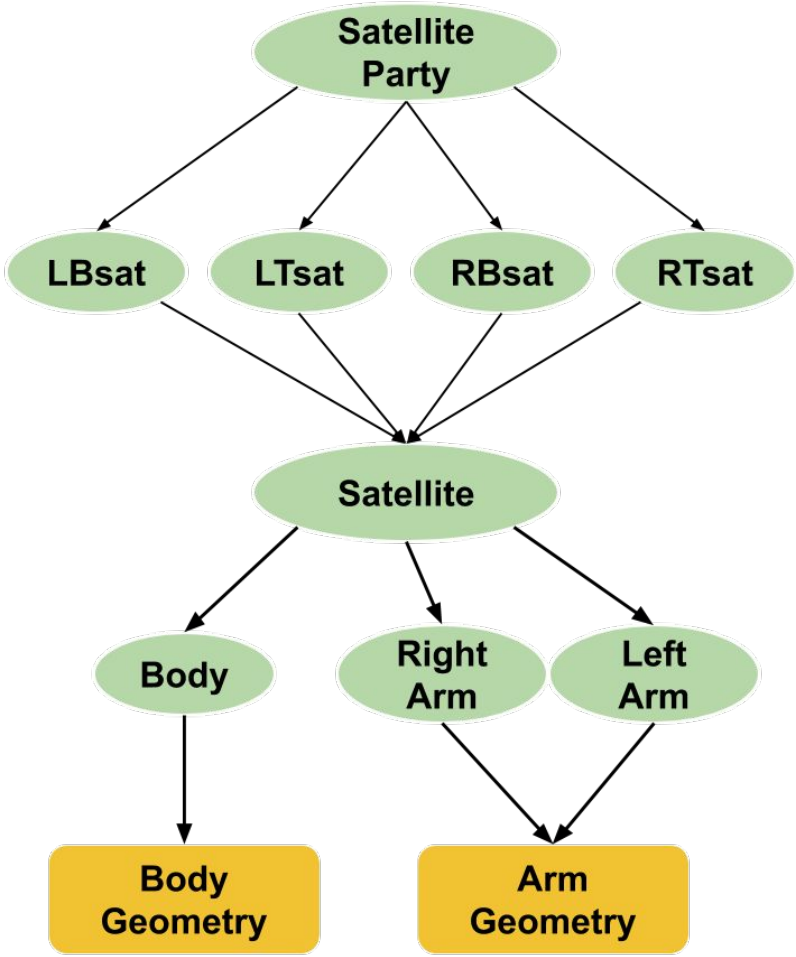



```
SatelliteParty.addChild(LBsat)
SatelliteParty.addChild(LTsat)
SatelliteParty.addChild(RBsat)
SatelliteParty.addChild(RTsat)
```



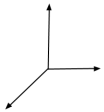
Robot Army

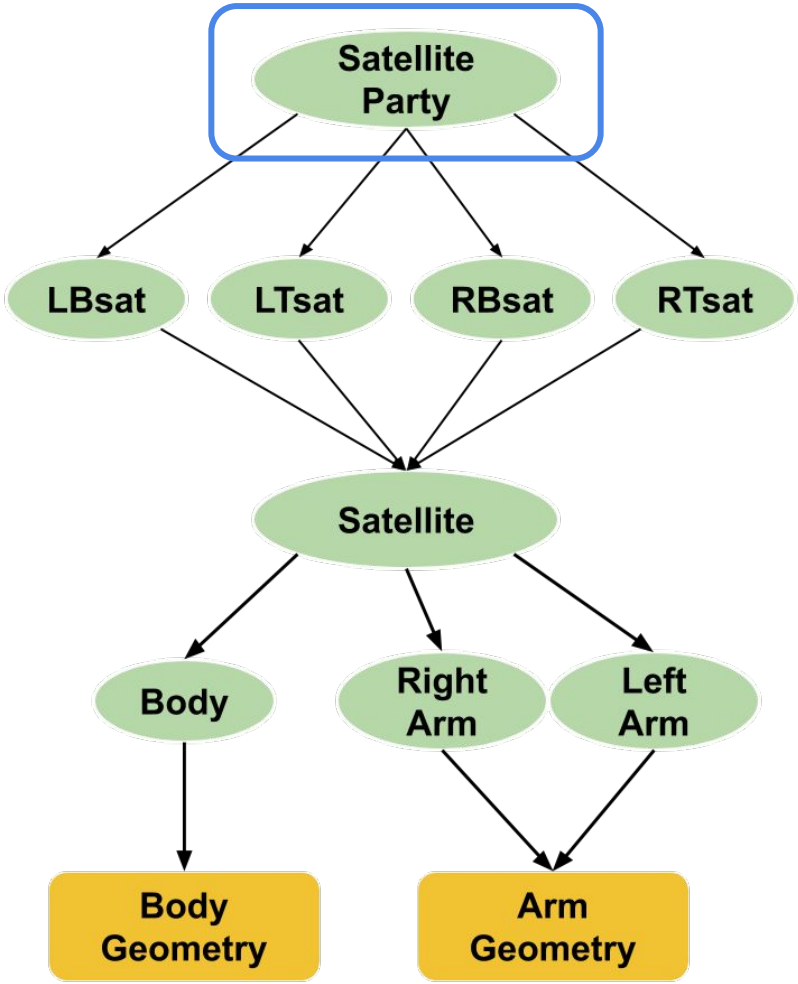
```
LBsat.addChild(Satellite)
LTsat.addChild(Satellite)
RBsat.addChild(Satellite)
RTsat.addChild(Satellite)
```



Robot Army

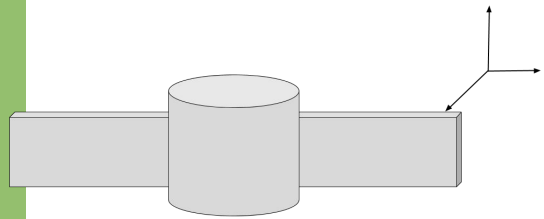
```
SatelliteParty->draw(I)
```

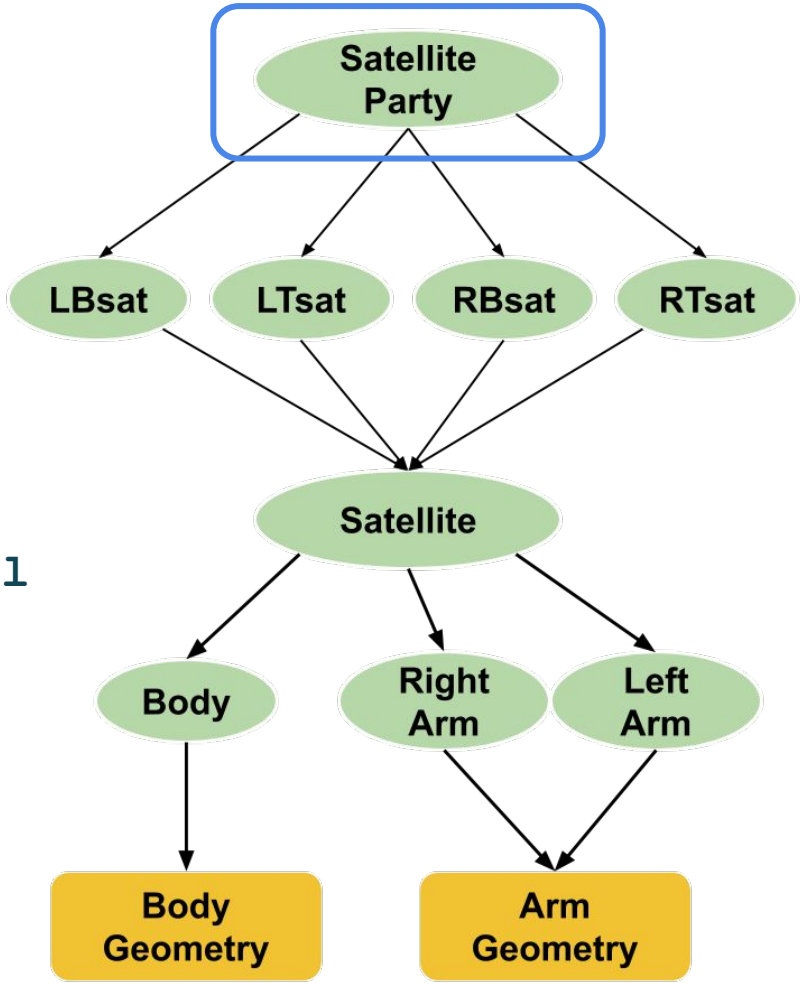




Robot Army

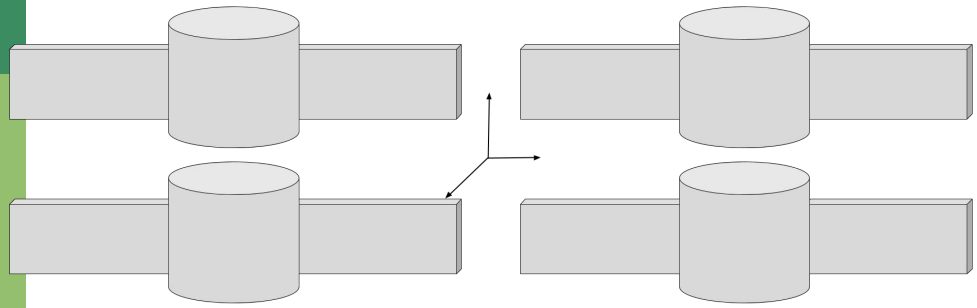
```
SatelliteParty->draw(I)

...

model = I*T'1*I*T2*initModel
```

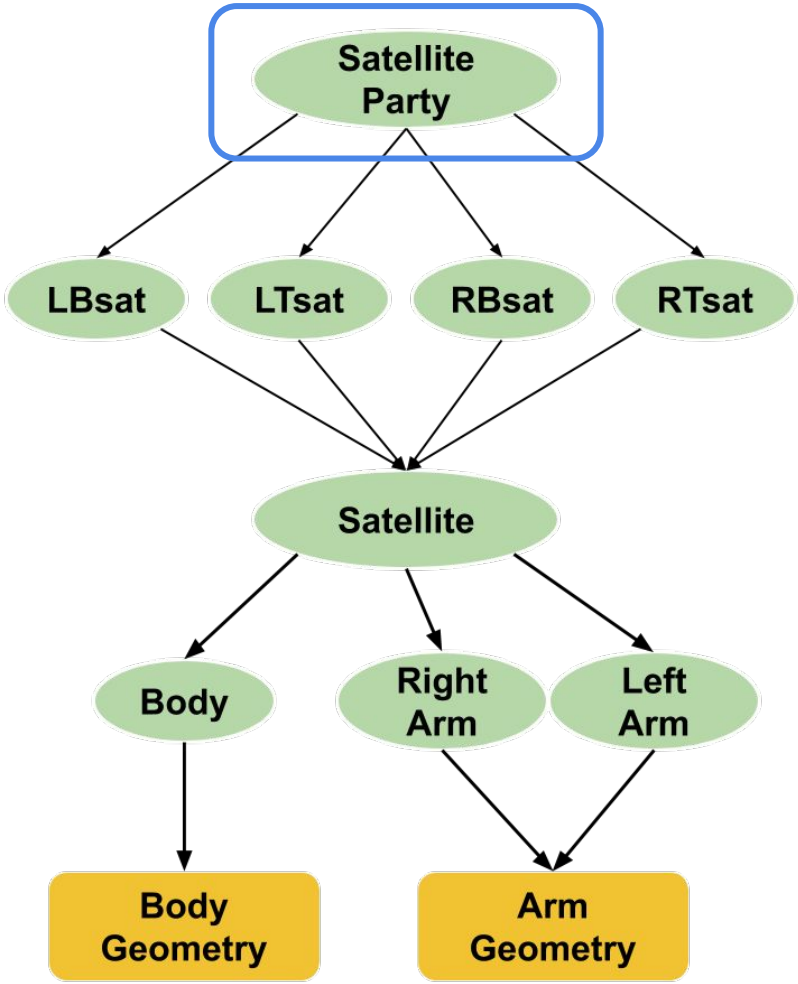




Robot Army

```
SatelliteParty->draw(I)
```





Robot Army

- Note:
  - Only load the objects 1x
    - ie. if using the limb twice, we just load it one time
  - Only one instance of robot for the entire army
    - Draw many robots because we add it as a child to many Transformation nodes

Questions?