CSE 167

Discussion 05 ft. Yining
11/06/2017

Announcements

- Project 3 is due 11/13(Mon) 2PM
- Midterm I grades are up
  - Q8 is now an extra credit!!!

Contents

- Texture
  - Buffer and shader
- Camera rotation
- Scene graph
  - Class hierarchy
  - Class implementation
  - Example
- Culling

Texture: buffer and shader

```
GLuint textureID;
glGenTextures(1, &textureID);

glBindTexture(GL_TEXTURE_2D, textureID);
unsigned char * image = loadPPM("myTexture.ppm", width, height);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, image);

glBindTexture(GL_TEXTURE_2D, 0);
```

Texture: buffer and shader

```
GLuint textureID;
glGenTextures(1, &textureID);

glBindTexture(GL_TEXTURE_2D, textureID);
unsigned char * image = loadPPM("myTexture.ppm", width, height);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, image);

glBindTexture(GL_TEXTURE_2D, 0);
```

- Generates the texture
  - Similar to “glGenBuffers”
- Let the GPU know that we are going to send a texture
  - “textureID” is the identifier of such texture

Texture: buffer and shader

```
GLuint textureID;
glGenTextures(1, &textureID);

glBindTexture(GL_TEXTURE_2D, textureID);
unsigned char * image = loadPPM("myTexture.ppm", width, height);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, image);

glBindTexture(GL_TEXTURE_2D, 0);
```

- Similar to VBOs, we need to bind the texture so OpenGL knows which texture we are modifying
  - (Reminder) OpenGL is a state machine!
- Highly recommended to unbind (bind texture 0) once you're done modifying the texture to avoid unexpected results

Texture: buffer and shader

```
GLuint textureID;
glGenTextures(1, &textureID);

glBindTexture(GL_TEXTURE_2D, textureID);
unsigned char * image = loadPPM("myTexture.ppm", width, height);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, image);

glBindTexture(GL_TEXTURE_2D, 0);
```

- Load texture and send it to the GPU
  - GL_TEXTURE_2D: type of texture
  - 0: mimap level
  - GL_RGB: Internal representation of the texture in the GPU
  - width/height: variable for the texture width/height
  - 0: should always be 0
  - GL_RGB: Representation of the texture that we are sending
  - GL_UNSIGNED_BYTE: Type of individual values in our image array
  - image: memory location where the pixel information is stored

Texture: buffer and shader

1 - Create

2 - Bind

3 - Load & send

```
GLuint textureID;
glGenTextures(1, &textureID);

glBindTexture(GL_TEXTURE_2D, textureID);
unsigned char * image = loadPPM("myTexture.ppm", width, height);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, image);

glBindTexture(GL_TEXTURE_2D, 0);
```

4 - Unbind

Texture: buffer and shader

- A texture unit is of type sampler* in the fragment shader
  - Usually a sampler2D since we normally use 2D images as textures
  - This should be a uniform variable
    - uniform sampler2D *someVariableName*;
- The value of sampler2D is an unsigned int
  - *glUniform1i(...)*; or *glUniform1ui(...)*;
  - Value determined by the call glActiveTexture(GL_TEXTURE*n*)
    - This line has to be somewhere in your code to bind textures in the first place
    - *n* is some integer number, usually between 0-7
      - We recommend using GL_TEXTURE0 or GL_TEXTURE1
      - Just cause some machines don't support multiple active textures

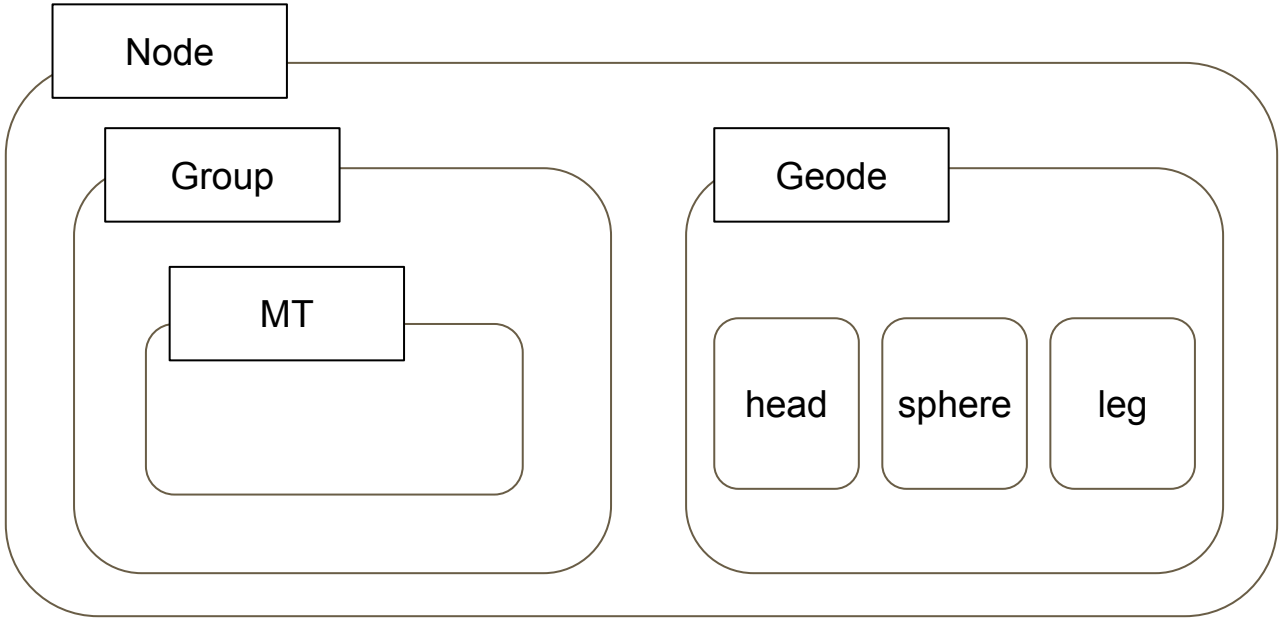
Texture: buffer and shader

- Skybox is just an extension to all this
- Instead of sending a single 2D texture, now you would be sending 2D texture cube map
- Create a separate set of shaders for the skybox

Camera rotation

- If the trackball mapping is not working, use keyboard instead to do zoom-in/out, translation, and rotation
- How to actually rotate it?
  - Chalkboard time!

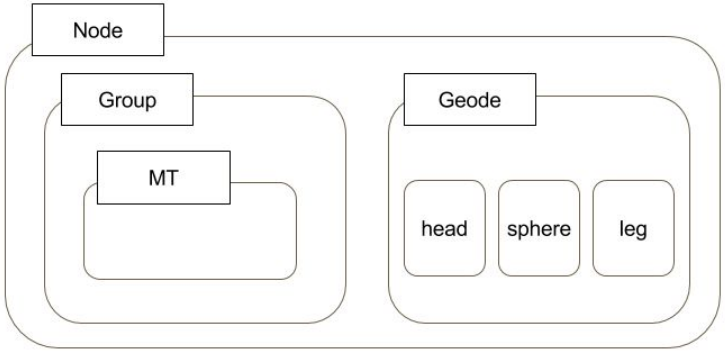
Scene graph: class hierarchy



Scene graph: class implementation

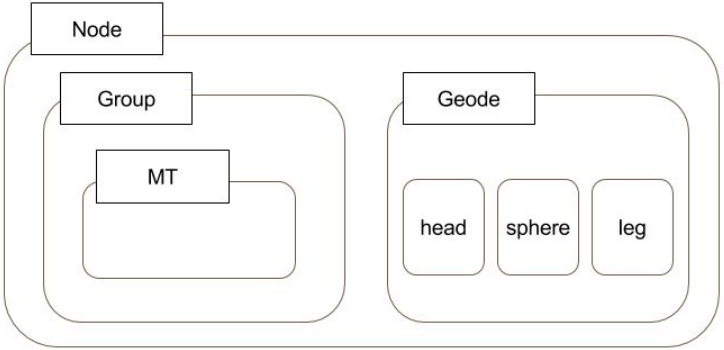
- Use virtual functions
-

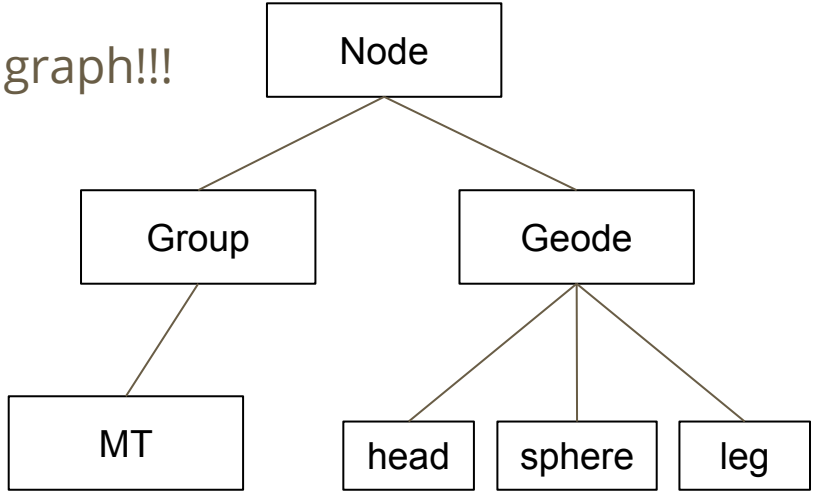
```
class Node
{
        …
        virtual void draw(int program, glm::mat4 M) = 0;
}

class Group: public Node
{
        …
        virtual void draw(int program, glm::mat4 M);
}

class MatrixTransform: public Group
{
        …
        void draw(int program, glm::mat4 M);

}
```



Scene graph: example

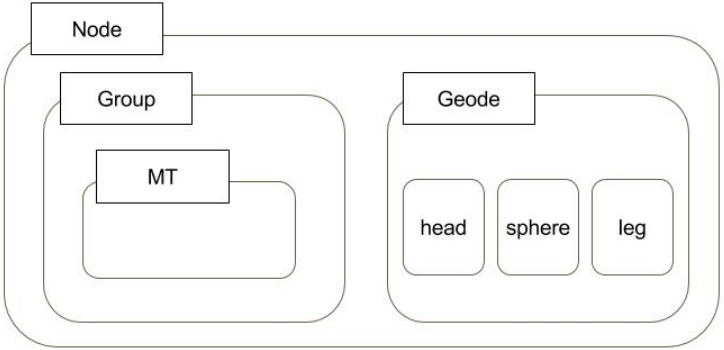
The class hierarchy is not equal to the scene graph!!!



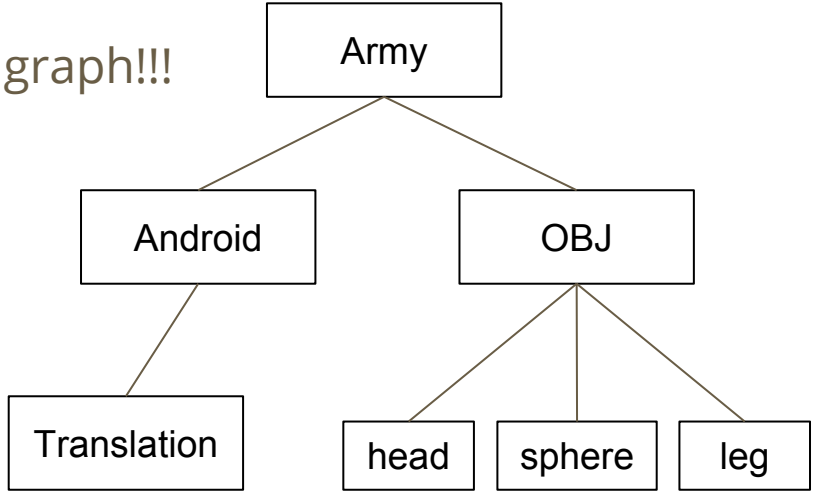


Scene graph: example

The class hierarchy is not equal to the scene graph!!!

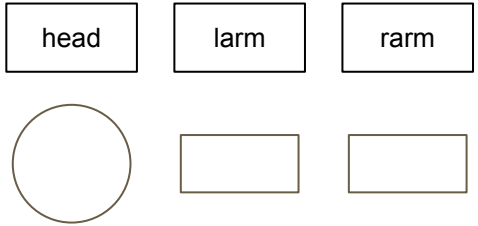


≠



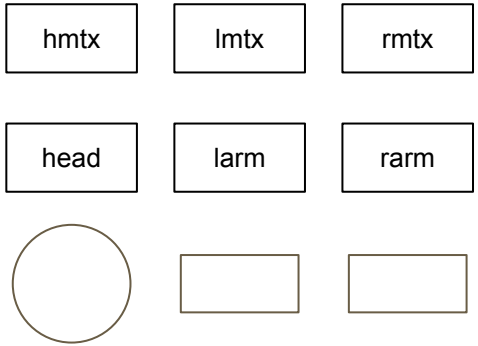
Scene graph: example(bottom-up)

```
Geode* head = new OBJ('head');

Geode* larm = new OBJ('arm');

Geode* rarm = new OBJ('arm');
```



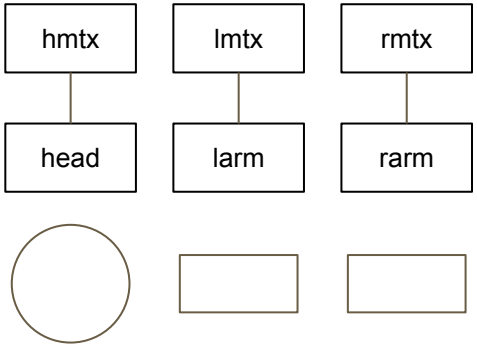
Scene graph: example(bottom-up)

```
MT* hmtx = new MT(glm::mat4(1.0f));

MT* lmtx = new MT(glm::translate(glm::mat4(1.0f), vec3(-2.0f, 0.0f, 0.0f));

MT* rmtx = new MT(glm::translate(glm::mat4(1.0f), vec3(+2.0f, 0.0f, 0.0f));
```



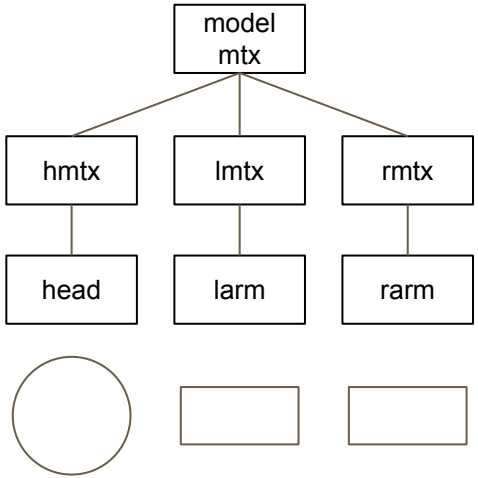
Scene graph: example(bottom-up)

```
hmtx->addChild(head);

lmtx->addChild(larm);

rmtx->addChild(rarm);
```



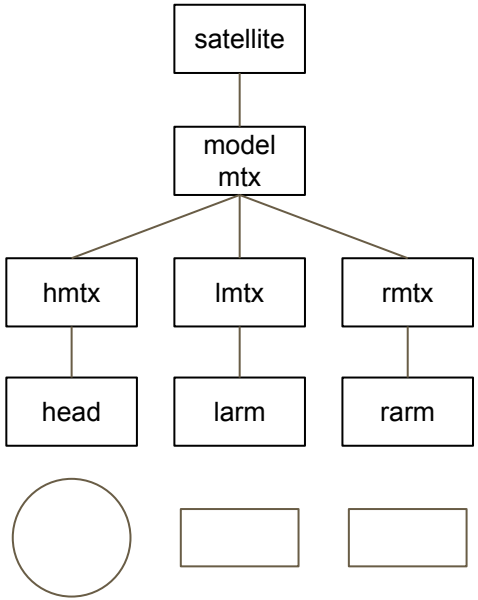
Scene graph: example(bottom-up)

```
MT* modelmtx = new MT(glm::mat4(1.0f));

modelmtx->addChild(hmtx);

modelmtx->addChild(lmtx);

modelmtx->addChild(rmtx);
```



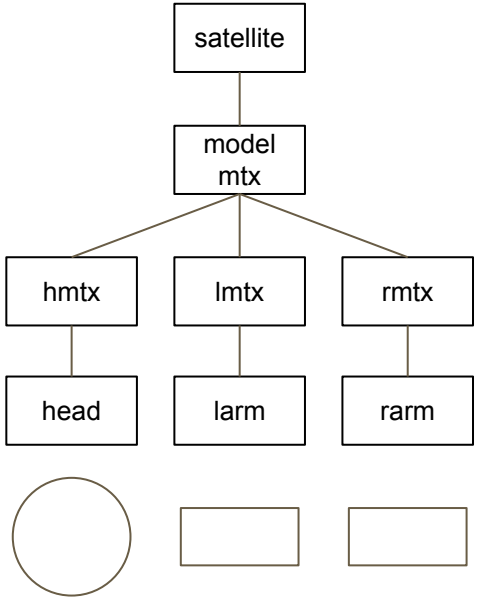
Scene graph: example(bottom-up)

```
Group* satellite = new Group();

android->addChild(modelmtx);
```



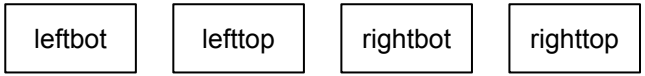
Scene graph: example(bottom-up)

If we call draw method on satellite right now, it would look like:



Scene graph: example(bottom-up)

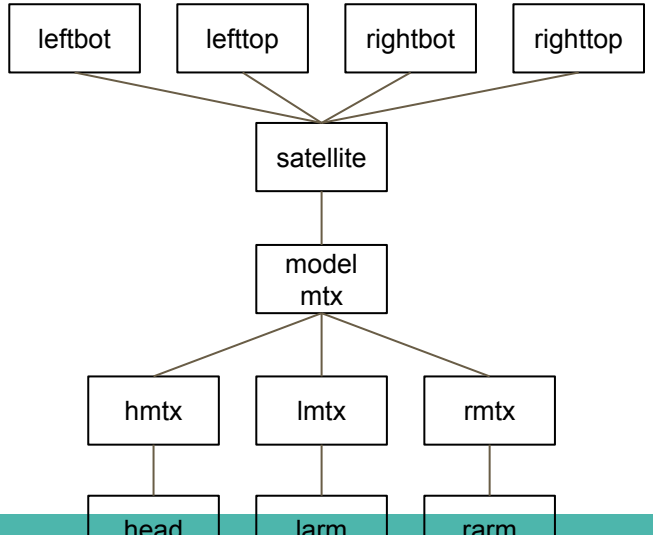
```
MT* leftbot = new MT(glm::translate(glm::mat4(1.0f), vec3(-1.0f, -1.0f, 0.0f));

MT* lefttop = new MT(glm::translate(glm::mat4(1.0f), vec3(-1.0f, +1.0f, 0.0f));

MT* rightbot = new MT(glm::translate(glm::mat4(1.0f), vec3(+1.0f, -1.0f, 0.0f));

MT* righttop = new MT(glm::translate(glm::mat4(1.0f), vec3(+1.0f, +1.0f, 0.0f));
```



Scene graph: example(bottom-up)

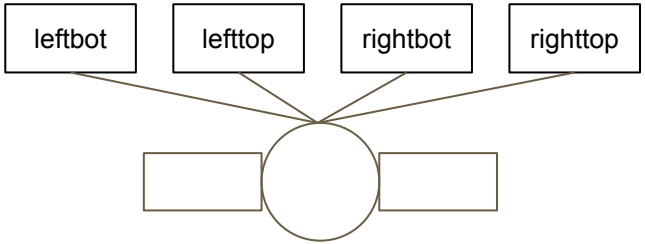
```
leftbot->addChild(satellite);

lefttop->addChild(satellite);

rightbot->addChild(satellite);

righttop->addChild(satellite);
```

Note that there is only a single instance of satellite!!!



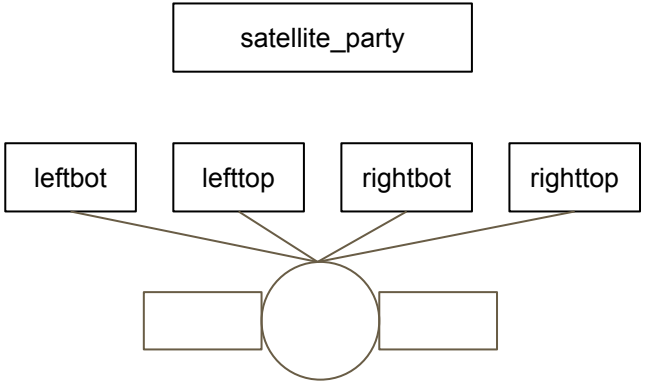
Scene graph: example(bottom-up)

```
leftbot->addChild(satellite);

lefttop->addChild(satellite);

rightbot->addChild(satellite);

righttop->addChild(satellite);
```



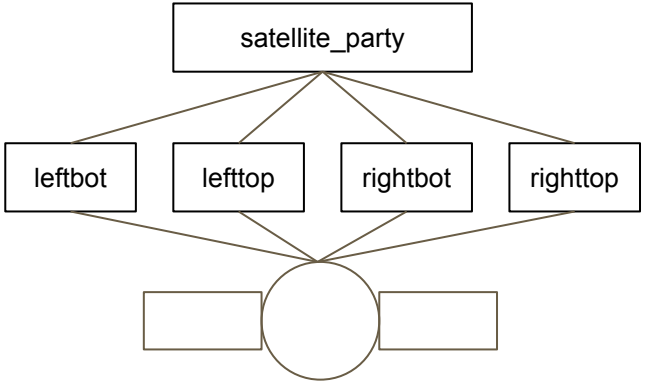
Scene graph: example(bottom-up)

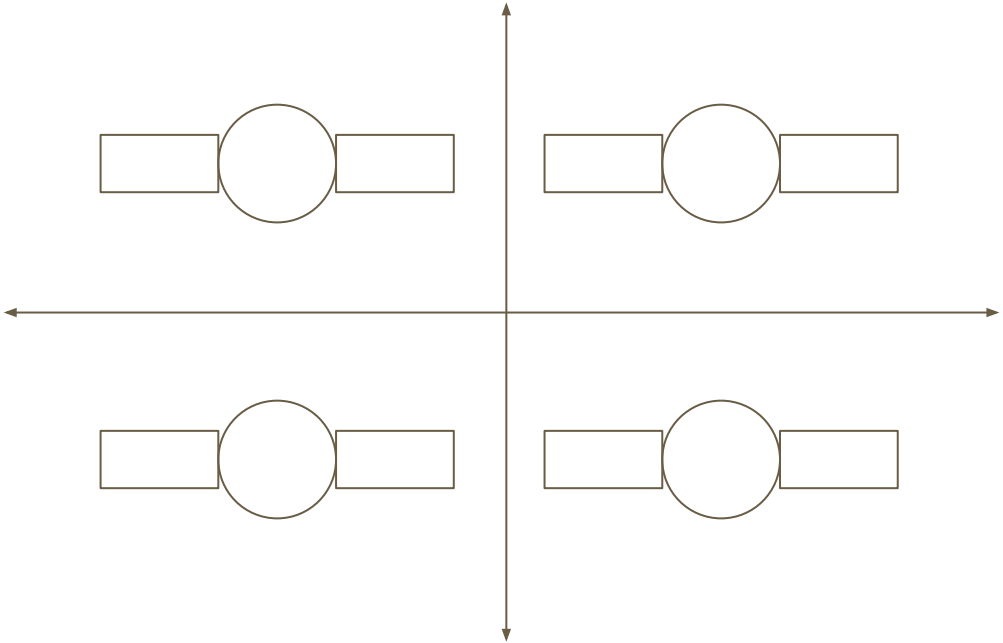
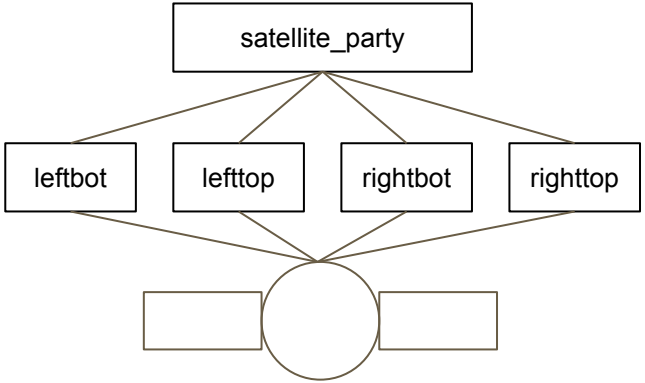
Group* satellite_party = new Group();



Scene graph: example(bottom-up)

```
satellite_party->addChild(leftbot);

satellite_party->addChild(lefttop);

satellite_party->addChild(rightbot);

satellite_party->addChild(righttop);
```



Scene graph: example(bottom-up)

If we call draw method on satellite_party right now, it would look like: