
EDAF80 Introduction to Computer Graphics

# Seminar 1

## Hierarchical Transformation

Michael Doggett

2017 Some slides by Carl Johan Gribel, 2010-13

# Today

- Hierarchical Transformation and Scene Graphs
  - TRS - Translation, Rotation, Scaling
- C++/OpenGL Framework
  - Bonobo, and other libraries
- Visual Studio
  - Debugging
- Assignment 1 - Solar System

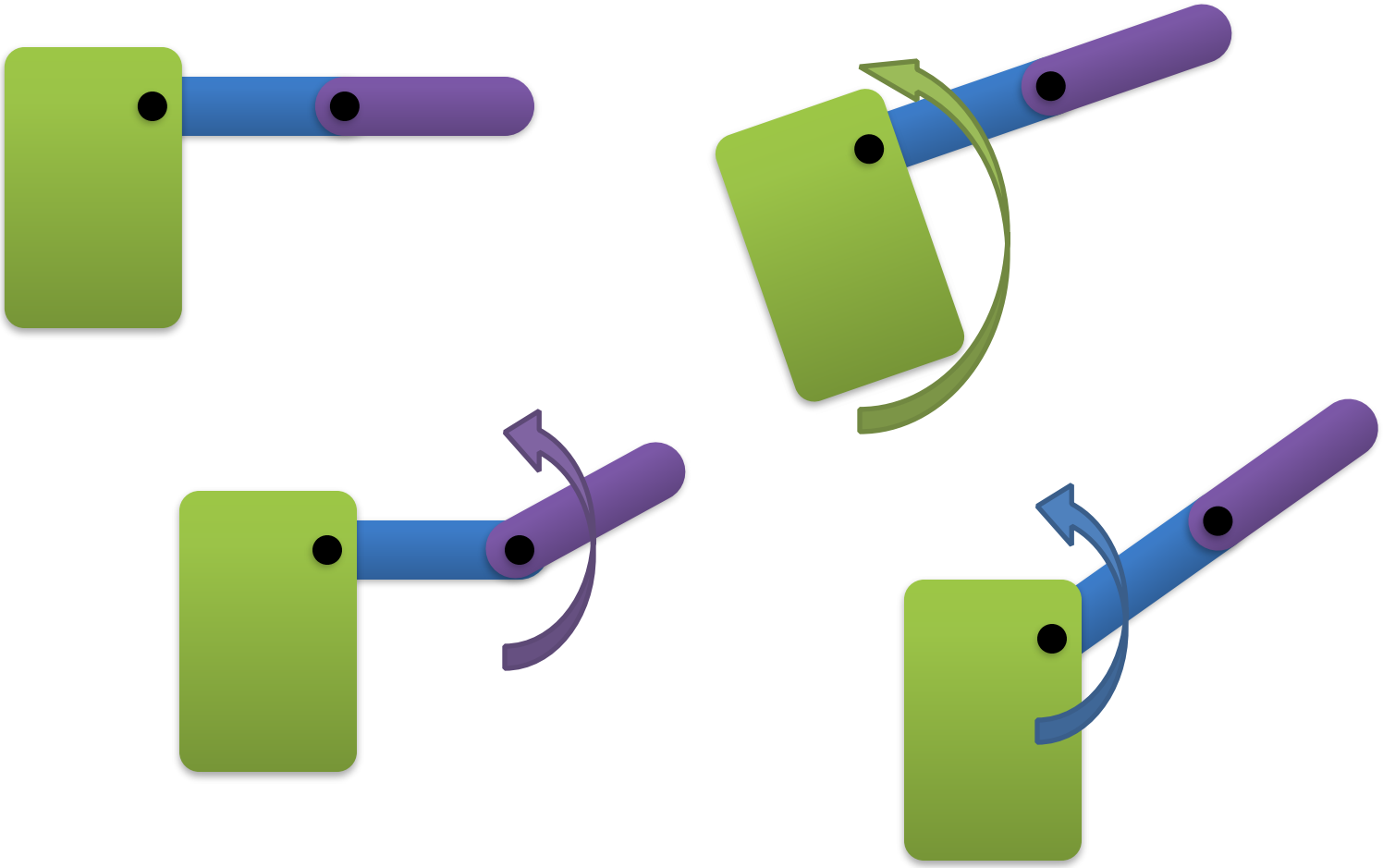
# From lecture: transformation matrices

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

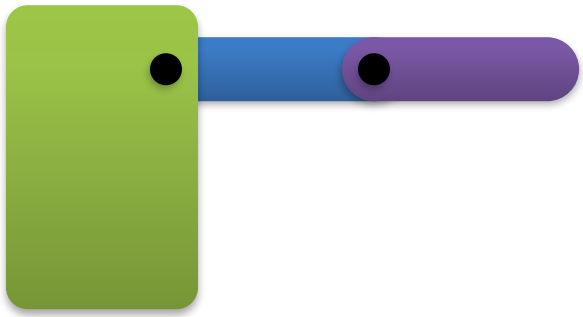
$$\mathbf{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

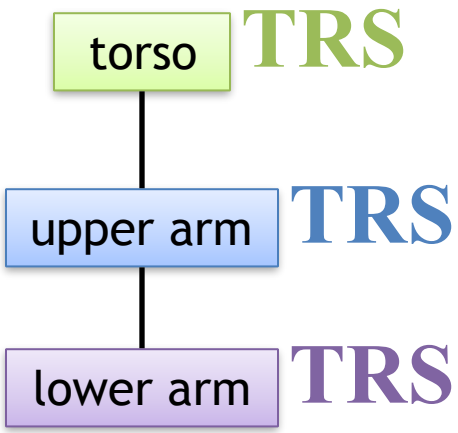
$$\mathbf{S} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

# Hierarchical transform: rigid body

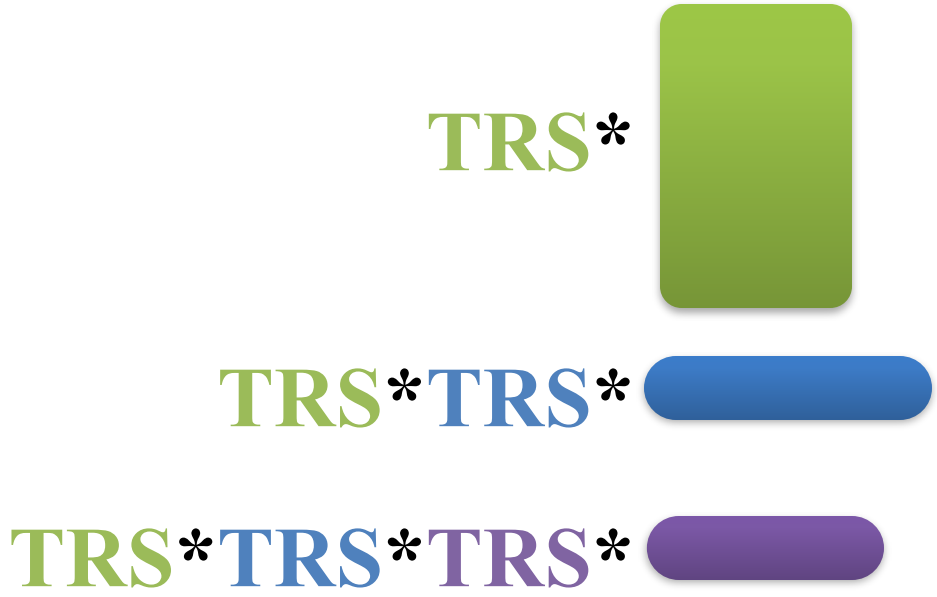


# Hierarchical transform: rigid body



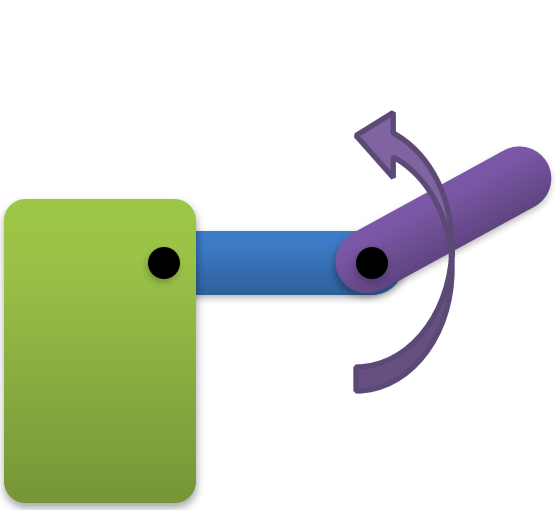


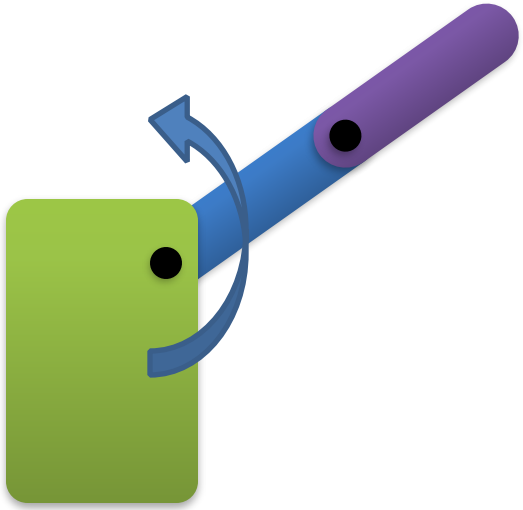

**geometry & scene graph**

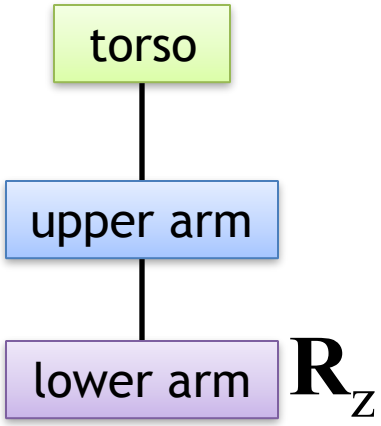


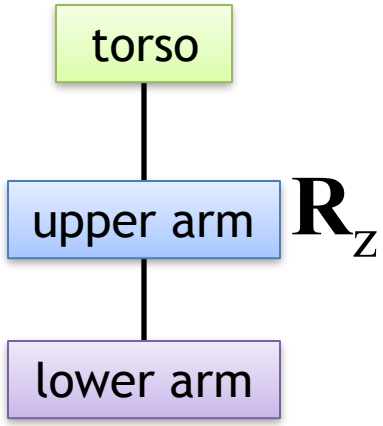

**transformations applied during rendering**

# Hierarchical transform: rigid body

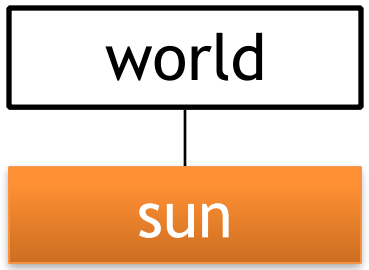


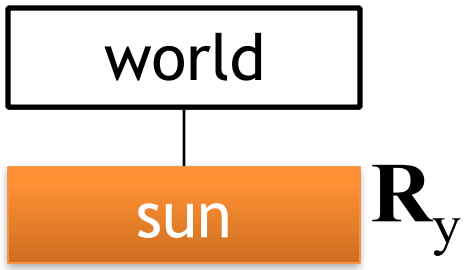


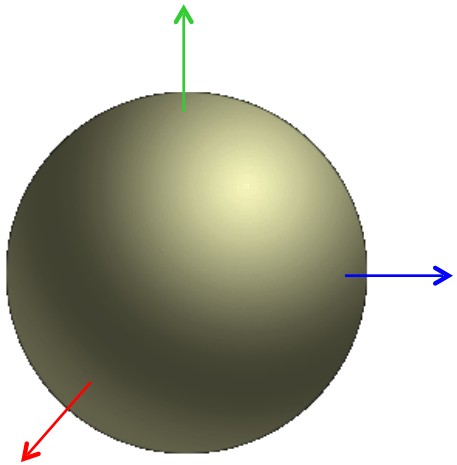


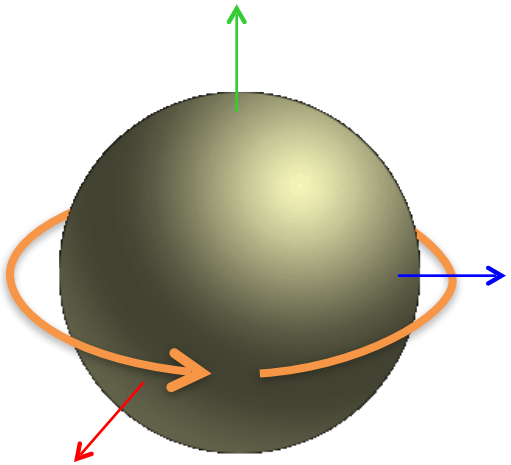



# Hierarchical transformation



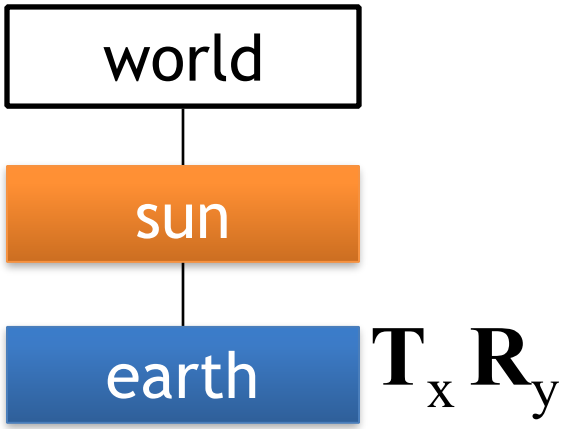



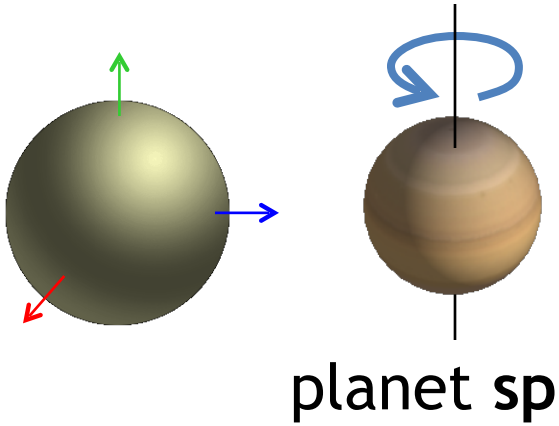





sun **spin**

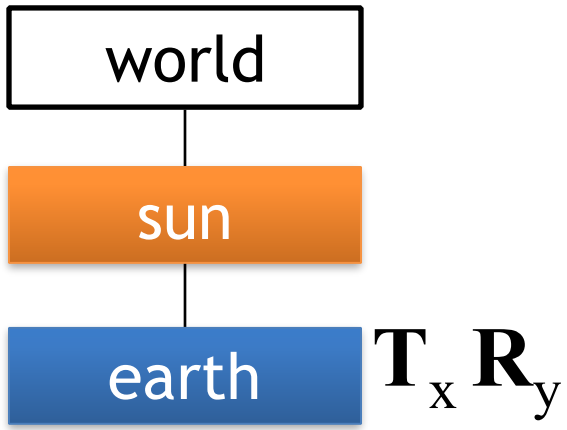
# Hierarchical transformation: Spin vs Orbit

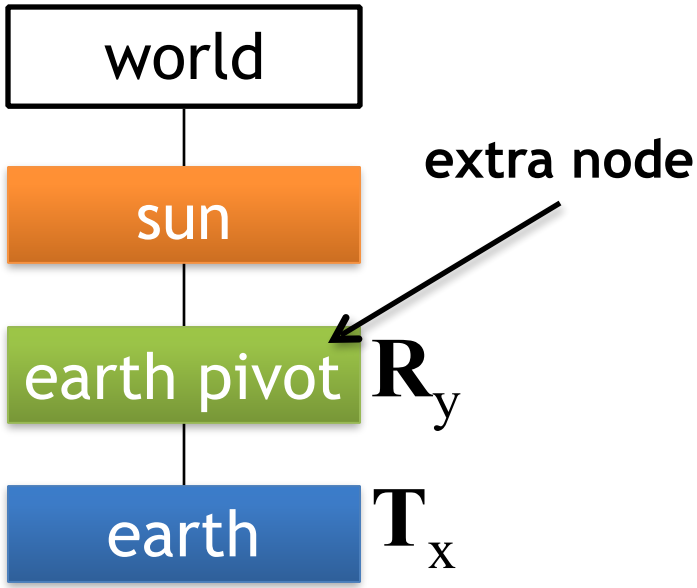


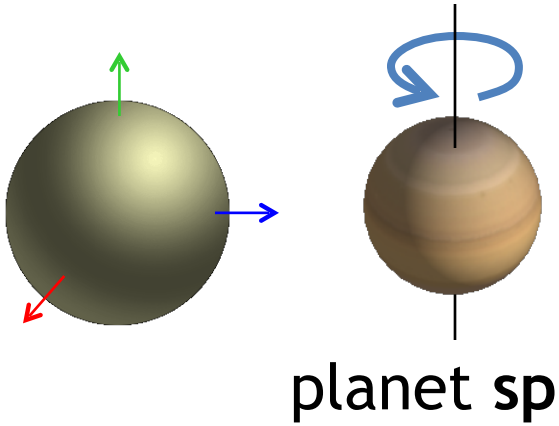



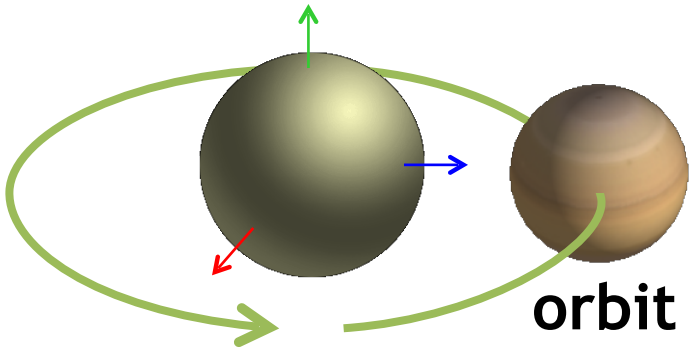
planet **spin**

# Hierarchical transformation: Spin vs Orbit

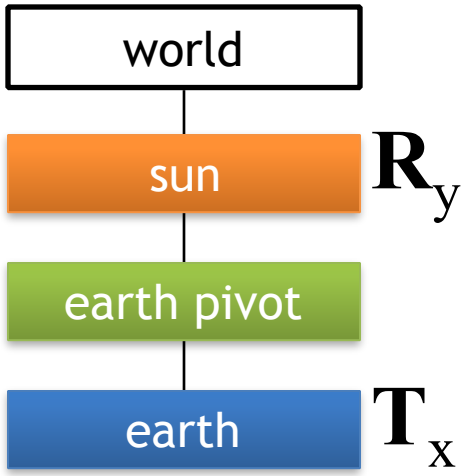


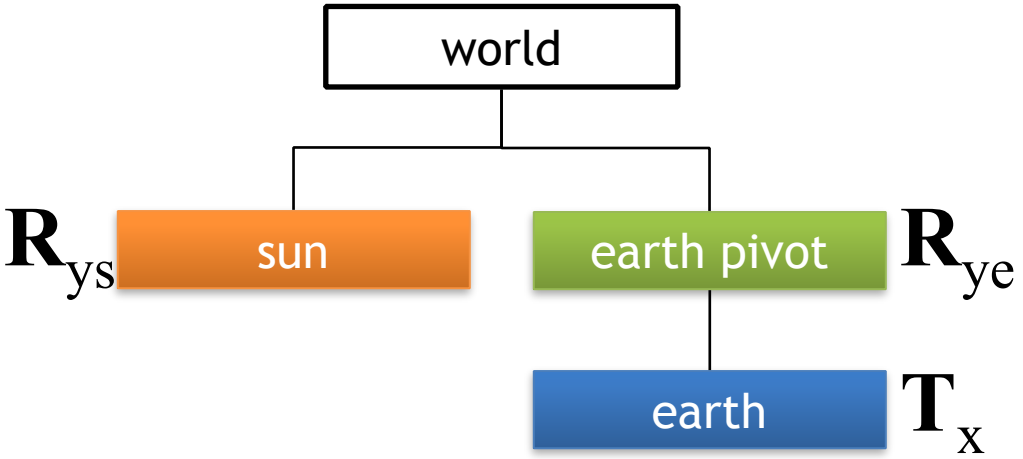



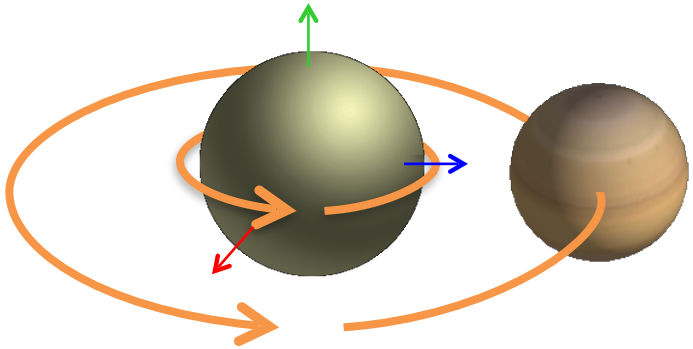



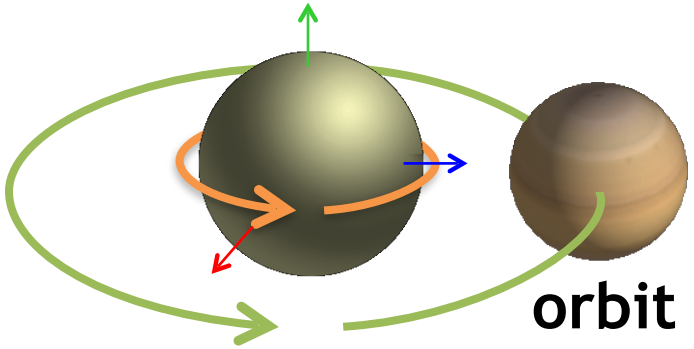
# Hierarchical transformation: Spin vs Orbit



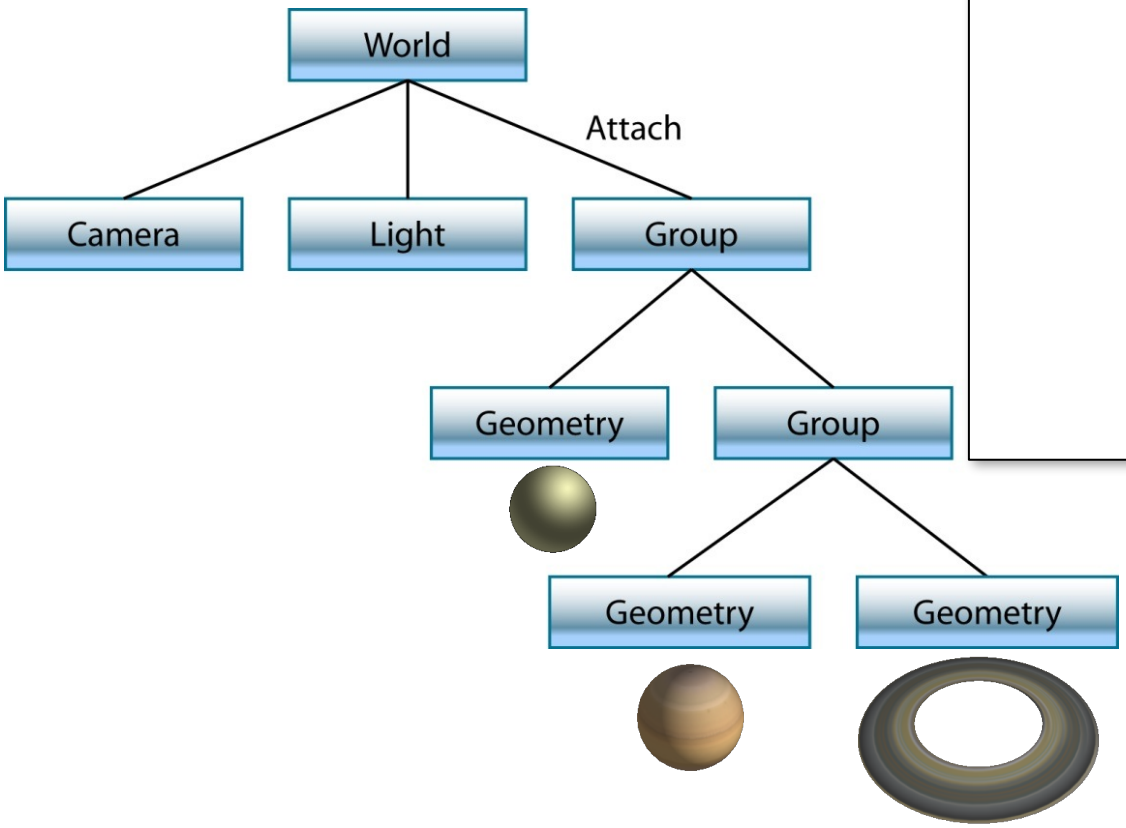



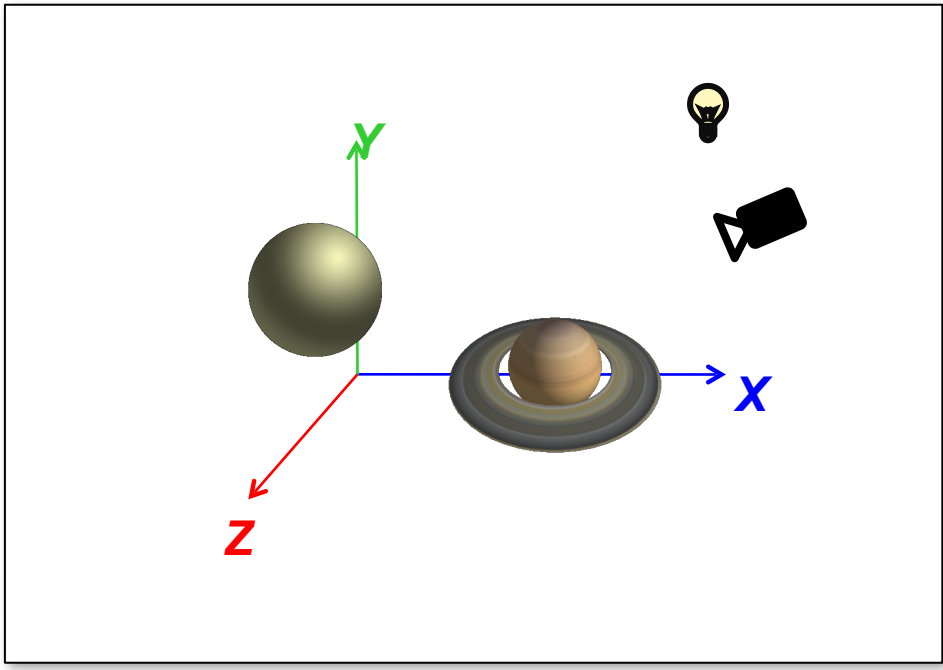
sun **spin**, planet **orbit** (coupled)




sun **spin**, planet **orbit** (decoupled)

# Scene Graphs



# C++/OpenGL Framework - Libraries

- User Interface (**GLFW**, **imgui**)
  - Window, GL context, mouse, key, log window, GUI
  - Using OpenGL 4.1
- Resource loading
  - Model/Geometry loading (**assimp**)
  - Image/Texture loading (**lodepng**)
- Vector algebra library (**GLM**)
  - Based on OpenGL Shading Language (GLSL) Specification
- Don’t need to look at this code, just use them as tools

# C++/OpenGL Framework - Functions

- Sphere structure (sphere_t)
- Node class
    - child pointers to build a simple scene graph
    - render() member function to Draw the sphere
- OpenGL texture setup function (loadTexture2D() )
    - Shader setup - loading, compiling, linking (createProgram() )
- while loop to render scene graph
    - Add per frame node operations here (for example : sun.rotate_y(0.01f);)
    - Pushes root_node onto a stack, then process all nodes

# Making a node

- `sphere` and `shader` are setup and can be reused

```
Node sun = Node();
GLuint sun_texture = loadTexture2D("sunmap.png");
sun.set_geometry(sphere);
sun.set_program(shader, [](GLuint /*program*/){} );
sun.add_texture("diffuse_texture", sun_texture, GL_TEXTURE_2D);
```

# Adding nodes

- Add new nodes to the sun, etc.

```
Node world = Node();

world.add_child(&sun);
sun.add_child(/* Add your node here */);
```

# Moving nodes

- Use translation, rotation and scaling functions
- Use `nowTime` for current time
  - How is this different to constant values?

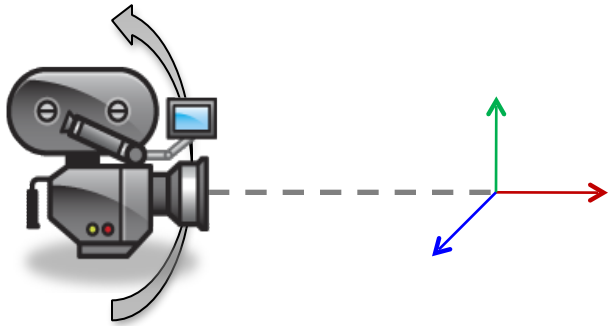
```
sun.set_translation(glm::vec3(std::sin(nowTime),
0.0f, 0.0f));

earth.translate( vec3 );

earth.scale(0.2);

sun.set_rotation_y(0.7);
```

# Interaction



*”First Person-style”*


- FPS camera : “WASD” move camera forward/left/back/right
- Mouse
  - Over non-UI elements: left button -> control camera
  - Over UI elements: left button -> control gui (log window). double click to minimize

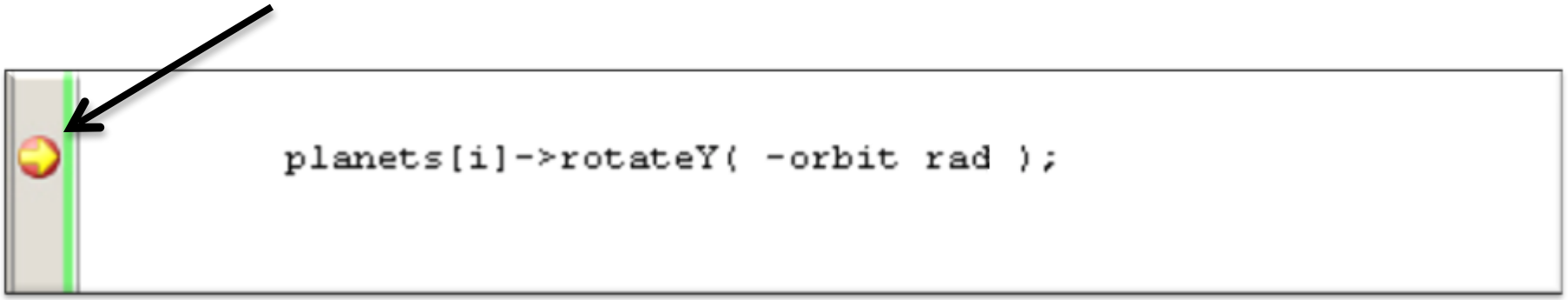
# Visual Studio Debugging

- breakpoints
- DataTips
- `printf`

# Breakpoints

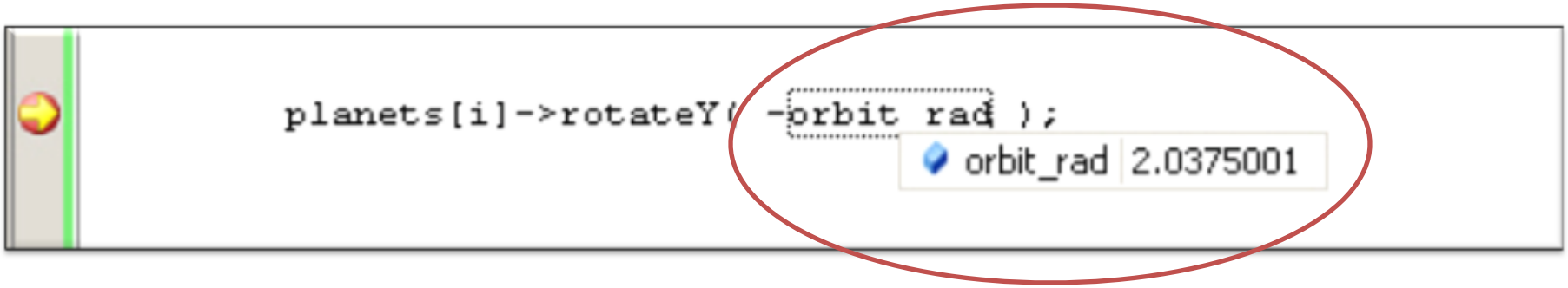
- Toggle breakpoint on current line with **F9**
  Pauses execution: enters **break mode**

- Right-click on the breakpoint symbol to add conditions, hit-counters, filters ...



# DataTips

- Available during **break mode**
- Display / edit variables by hovering over them:




- Can even expand hierarchically into class members, pointer-targets etc
- Right-click and select ”Watch” to pin variable to a Watch-window

# printf

- Brute force debugging:
  print whatever you need to monitor to the standard output (console window)

```
printf("I want to monitor this value: %f\n", var);
```

- ☺ Can format the output
- ☺ Can monitor output continuously as the program executes
- ☹ Messy code

# Assignment 1

- **Model the solar system!**

  Sun, planets, moons, comets...spaceships? It's up to you.

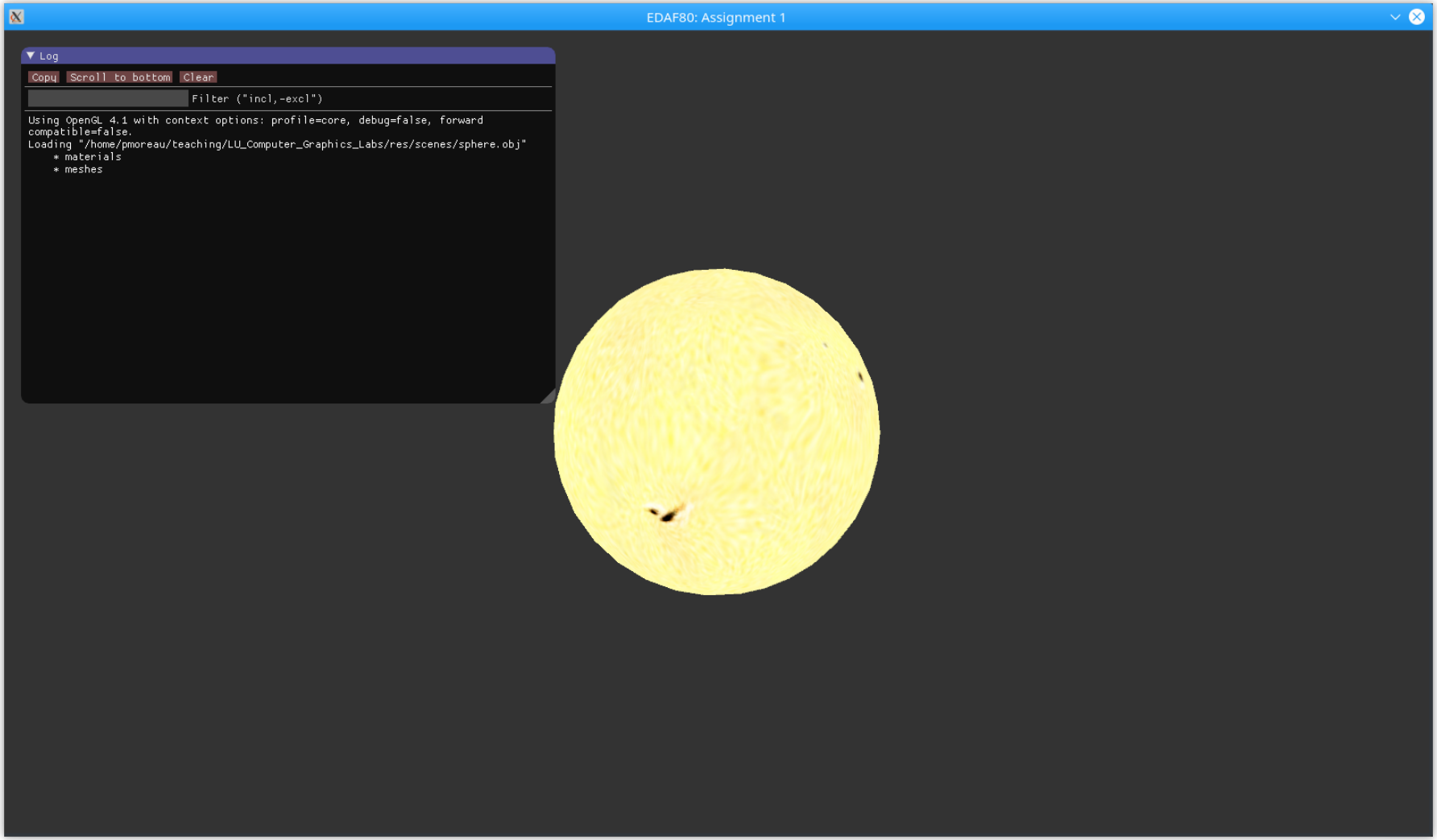
- Resources included in Framework are at your disposal
  - Models, textures, shaders
    Code for how to add shaders is included (more in assignment 3-4)

- See assignment description (PDF) for details
  - Available on webpage

# Assignment 1

- Get the code from Github
- Checkout the README.rst file in the download for more instructions
    - you can read it on Github https://github.com/LUGGPublic/CG_Labs
- Look for comments for where to add your new code in :
    - assignment1.cpp
        - Add planets, animate planets
        - Implement the traversal of the scene graph
- Student Lab Computers
    - **Windows will only allow EXEs to run in a directory with a directory called "Program" in the hierarchy**

# Assignment 1



# Textures

http://planetpixelemporium.com/planets.html

You will need to convert jpegs to png

# Next

- Download the code and get started
- Post questions to the forum
- Watch the forum for updates