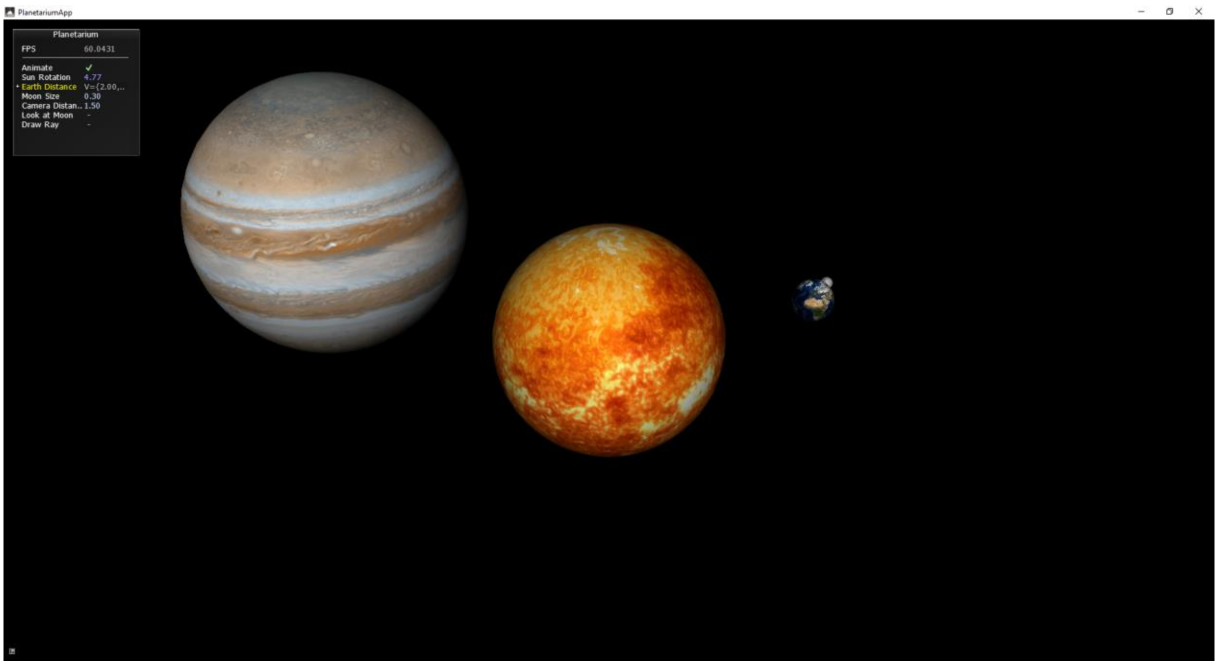
Computeranimation

186.139 - SS 2020

Assignment 1: Planetarium and Hierarchical Modelling

The goal of this assignment is to understand the basics of libcinder and hierarchical modelling.



```
FPS               60.0431
Animate           ✓
Sun Rotation      4.77
Earth Distance    V={2.00,..
Moon Size         0.30
Camera Distan..   1.50
Look at Moon      -
Draw Ray          -
```

Configurations:

Windows:

- Install Visual Studio 2019 or 2017 (the Community Edition is free)
  - https://visualstudio.microsoft.com/de/vs/community/
- Download the Windows SDK 10.0.18362.0 (this should be the most current one)
  - https://developer.microsoft.com/de-de/windows/downloads/windows-10-sdk/
- The prebuilt libraries can be downloaded from TUWEL
  - Download the CommonFiles package from TUWEL for your version of Visual Studio
  - This folder contains an include and lib folder
  - Copy these two folders into each exercise package
- Run the project – Debug – x64
  - 141 for VS 2017, 142 for VS 2019, SDK 10.0.18362.0 (this should be default)

If this does not work you may need to build the libraries for yourself. Contact me via discord! I can share some useful advice ;)

OS X (no Discord tutor support):

Get the libcinder Assignment package (Planetarium) from Tuwel

Get libcinder: https://libcinder.org/download

Linux (no Discord tutor support):

You will have to compile cinder for yourself. Follow one of the guides from here: https://github.com/cinder/Cinder/wiki/Cinder-for-Linux

Get the libcinder Assignment package (Planetarium) from Tuwel

Assignment:

Each of the tasks will reward you with a certain amount of points. To pass each exercise, you will need at least 50% of the total points of each exercise. You may use the provided Assignment package, which contains some code to help you learning cinder, or start your own entirely. If you are unsure how to do something with libcinder, you can check out the samples provided by cinder.

Hand in:

- A short video of your program (maximum of 60 seconds), showing each task.
- The source code.

Exercise 1: Planetarium

You will have to create a small planetarium (only containing a few objects). A finished example of Exercise 1 (in the form of a video) can be found in TUWEL.

There are comments in the code in the form of "//todo TASK1" where you can add your code.

Task 1 - Hierarchical modelling (3 points)

1) Place the sun at the centre
2) Place the jupiter in orbit around the sun
3) Place the earth in orbit around the sun
4) Place the moon in orbit around the earth

- Each astronomical object should rotate around its axis
- Choose whatever scale you think is appropriate
- You are free to implement it whichever way you want to, the only condition is that you use a world-matrix of some sort. One easy and quick way is to use the following commands:
  - gl::rotate()
  - gl::translate()
  - gl::scale()
  - gl::pushModelMatrix()
  - gl::popModelMatrix()
- You could also generate a mat4 for each object, and then set the necessary shader uniform (will be more work most likely)
- You can take a look at this tutorial:
  - https://libcinder.org/docs/guides/opengl/part1.html

Task 2 - Interface (2 points)

To help control the planetarium you should add an interface. The interface should at least contain the following options:

1) Animate (on/off)
   If switched off; the planetarium should freeze until switched on
2) Sun rotation [0,2*PI]
   Control (and show) the current rotation of the sun
   (Already implemented as an example)

3) Earth Distance (vec3)
   This vector should change the relation sun → earth
4) Moon Size (float)
   Change the size of the moon

- Cinder allows you to add interface params in a convenient way
- The Sample ParamsBasic or the provided code should explain the needed functions and classes

Task 3 - Camera (2 points)

For now the camera is looking at the centre. Add the following options:

1) Add an option to the Interface which allows setting the distance from the camera to the Sun
2) Add an option to the Interface which centres the camera on the moon

- You can get the current modelmatrix with: gl::getModelMatrix()
- You can extract the translation with glm::decompose
- see
  - https://glm.g-truc.net/0.9.6/api/a00204.html or
  - http://stackoverflow.com/questions/17918033/glm-decompose-mat4-into-translation-and-rotation/
- or use the function "getPosFromMat(mat4 matrix)" (see the sources)

Task 4 - Line (1 point)

1) Add a line which connects the jupiter and the moon
2) Add an option to the Interface which turns this line on or off

- The sample CameraPersp contains the use of a line.
- Use:
  - auto rayLine = gl::VertBatch(GL_LINES);
  - rayLine.vertex(vec3(0,0,0));
  - rayLine.draw();

Task 5 - Picking (2 points)

Clicking any of the astronomical objects should change their axis rotation (but must not change anything about any other object).

If two objects are in front of each other, only the object in the front (closest to the camera) shall change the axis rotation. (https://libcinder.org/docs/classcinder_1_1_sphere.html, use the intersect function with the min and max value and compare the min values to ensure that you only change the closest object)

- Picking is explained in the sample Picking3D. Instead of a bounding box you should use a ci::Sphere and ci::Sphere::intersect()