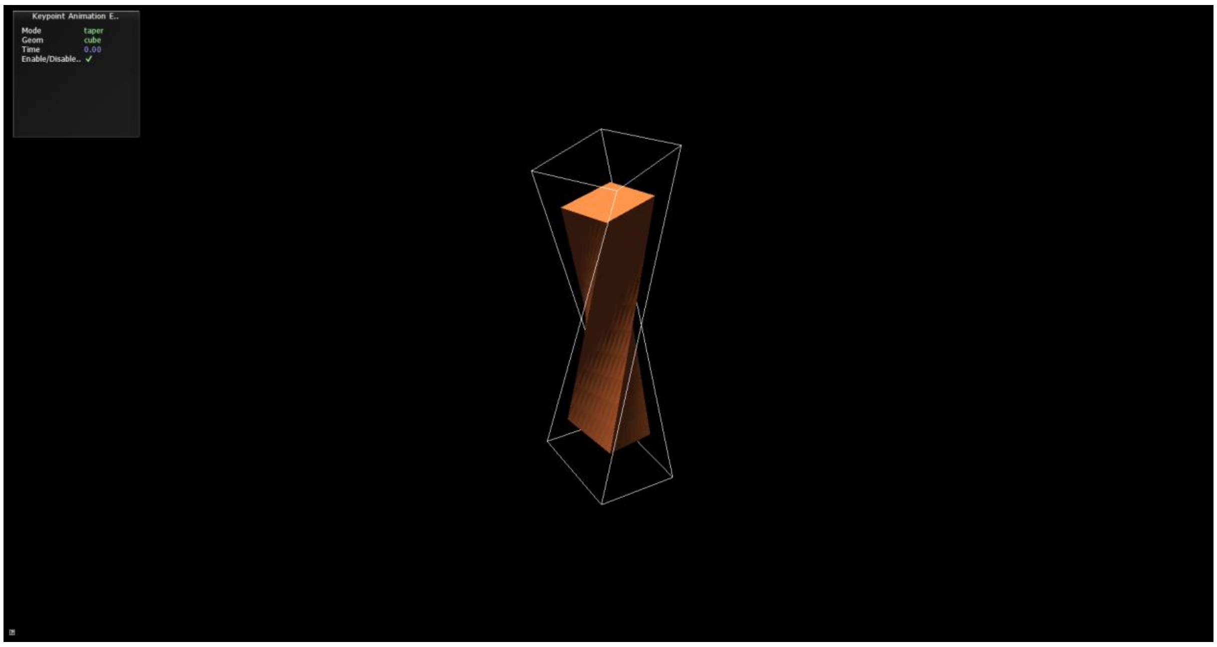
Computeranimation

186.139 - SS 2020

Assignment 3: Deformations

The goal of this assignment is to understand the basics of mesh deformations.



Mode taper
Geom cube
Time 0.00
Enable/Disable.. ✓

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 3: Deformations

This task is about deforming Objects. There are comments in the code in the form of “//todo TASK1” where you can add your code.

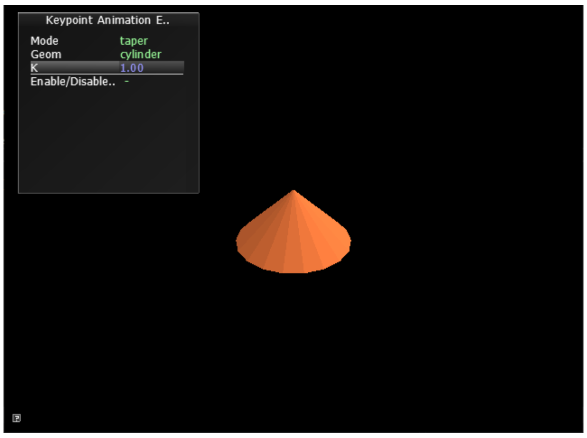
There is some basic functionality already implemented. Use your right mousebutton to rotate the camera, change the currently used geometry and transformation mode through the interface.

Task 1 – Taper Deformation (1 point)

Files: Shaders.h Function name: taper(vec4 pos, float k) inside GetDeformationShader()

Taper an object as described in the book using a glsl shader.

- Chapter 4, page 124/125
- You will have to swap y and z
    - The book uses a left hand coordinate system while opengl uses a right hand one
- The book’s formula does not include the parameter k for this deformation
    - At k = 1.0, the tapering should be as described in the book
    - At k = 0.0, the object is in it’s original, non-deformed state.



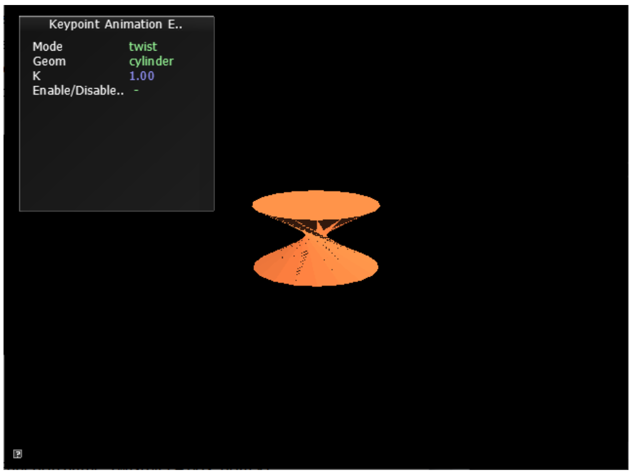
Keypoint Animation E..
Mode taper
Geom cylinder
K 1.00
Enable/Disable.. -

Task 2 – Twist Deformation (1 point)

Files: Shaders.h Function name: twist(vec4 pos, float k)

Twist an object as described in the book using a glsl shader.

- Chapter 4, page 124/125
- The TUWEL example multiplies the inner terms of the sin() and cos() with PI to emphasize the deformation. You can do this as well.



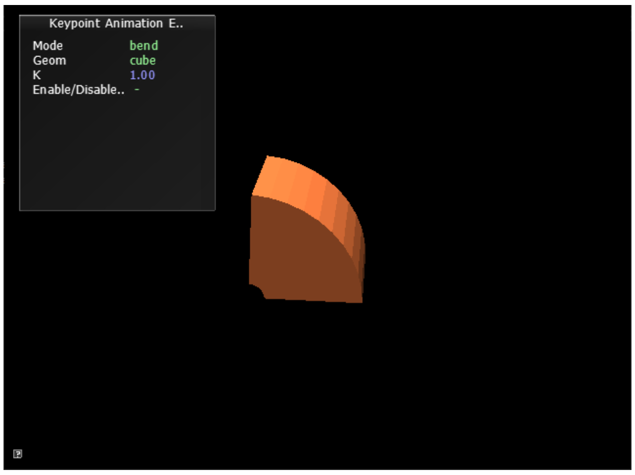
Mode twist
Geom cylinder
K 1.00
Enable/Disable.. -

Task 3 – Bend Deformation (2 points)

Files: Shaders.h Function name: vec3 bend (vec3 p, float k)

Bend an object as described in the book using a glsl shader.

- Chapter 4, page 127
- In Fig. 4.18 y' and z' show z<zmax in the first row. This should be z<zmin.



Mode bend
Geom cube
K 1.00
Enable/Disable.. -

Task 4 – Linear FFD (4 points)

Files: Shaders.h Function name: vec3 transformPoint(vec3 p)

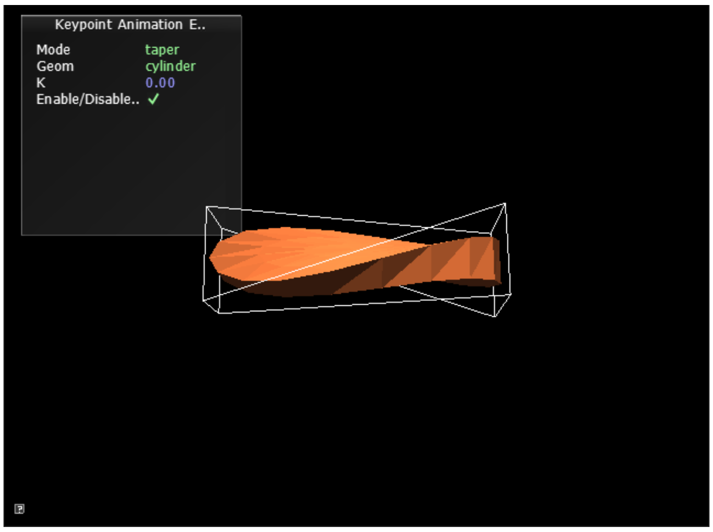
Implement linear free form deformation. You do not have to implement trivariate Bezier interpolation.

- The book explains trivariate Bezier interpolation for FFD. To keep things simpler, linear interpolation will be the goal of task 4.
- The principle of FFD is that you have a set of control points. Each vertex inside the volume formed by the control points is then manipulated by changing the positions of the control points.

- As a first step, each vertex needs to be registered with each control point. The dependency of each vertex to each control point lies between 0 and 1. The sum of all dependencies per vertex is 1.
- For example, a vertex that has the same position as a control point will only be dependent on this control point and none of the others. A point in the exact center of the volume will be equally dependent on all control points. A point on a face of the volume will only be dependent on the 4 control points spanning the face.

```
For each vector v:
    For each control point c:
        Do the next steps for x, y and z coordinate:
            Calculate distance from v to c
            Map the distance to [0-1] (dependency dep)
            //The cube has a side length of 2
            //dep = 1 => current point lies on control point
            //dep = 0 => current point = opposite side
        Multiply the 3 dependencies for x, y and z
```

- dep now holds the dependency of the vertex in regards to the control point. Multiplying the dependency factors with every control point will reconstruct the original vector. Multiplying the dependency factors with modified control points will produce the modified vertex.
- To save some time: You do not need to optimize this, recalculate the dependency each time you process a vertex (no need to store the vertex dependency between frames).
- This transform will only work if the FFD volume is convex and produce errors when it's not (which will happen during the animation.


Keypoint Animation E..
Mode taper
Geom cylinder
K 0.00
Enable/Disable.. ✓

Task 5 – Global Deformation of your choice (2 points)

Implement a global deformation (as in tasks 1-3) of your own choice. You will have to expand the vector "modeStrings" in KeypointAnimApp.cpp, line 32 and the main function in GetDeformationShader in "Shaders.h"