Computeranimation

186.139 - SS 2020

Assignment 4: Particles

The goal of this assignment is to understand the basics of Particle Animation.

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 4: Particles

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, middle mousebutton to move the camera and mousewheel to zoom.

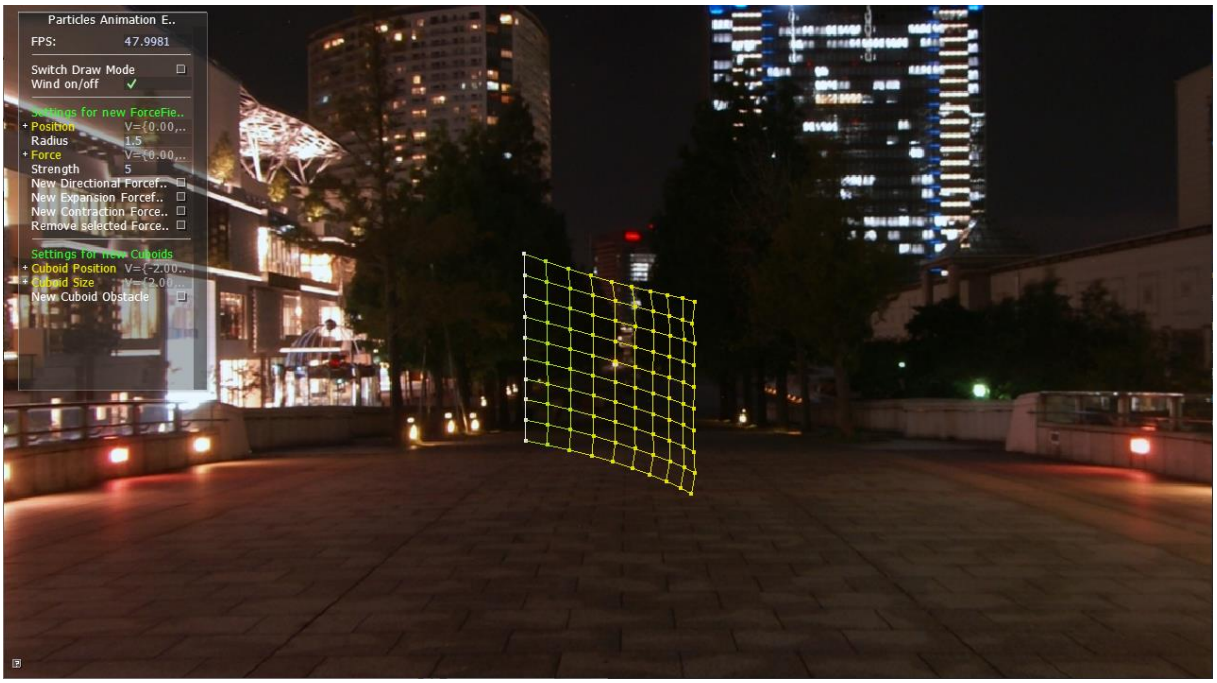
This task makes heavy use of transform feedback buffers, it should be possible to do most of the exercise without understanding the details, but reading up on them might help. (https://www.khronos.org/opengl/wiki/Transform_Feedback)

This exercise is mostly covered by Chapter 7 of the book.

Task 1 – Cloth Animation (2 points)

Files: Shaders/Cloth/update.vert Function name: void main(void)

Add an option to make the Flag swing in the wind. The implementation is entirely up to you. Try to make it look convincing (1 point) and simple (1 point) (1-2 lines of code). The original code for the cloth animation is taken from the cinder cloth animation sample.



Task 2 – Force Fields (3 points)

Files: Shaders/Particles/updateParticles.vert

Function names:
getDirectionalForceFieldInfluence(vec3 pos)
getExpansionForceFieldInfluence(vec3 pos)
getContractionForceFieldInfluence(vec3 pos)

1) Add a gravity force.
   Implement the functions to apply the forcefields to the particles.
2) Directional: Depending on a vector, each particle should have a force applied to it.
3) Expansion: Each particle should be pushed away from the center of the forcefield
4) Contraction: Each particle should be pulled towards the center of the forcefield

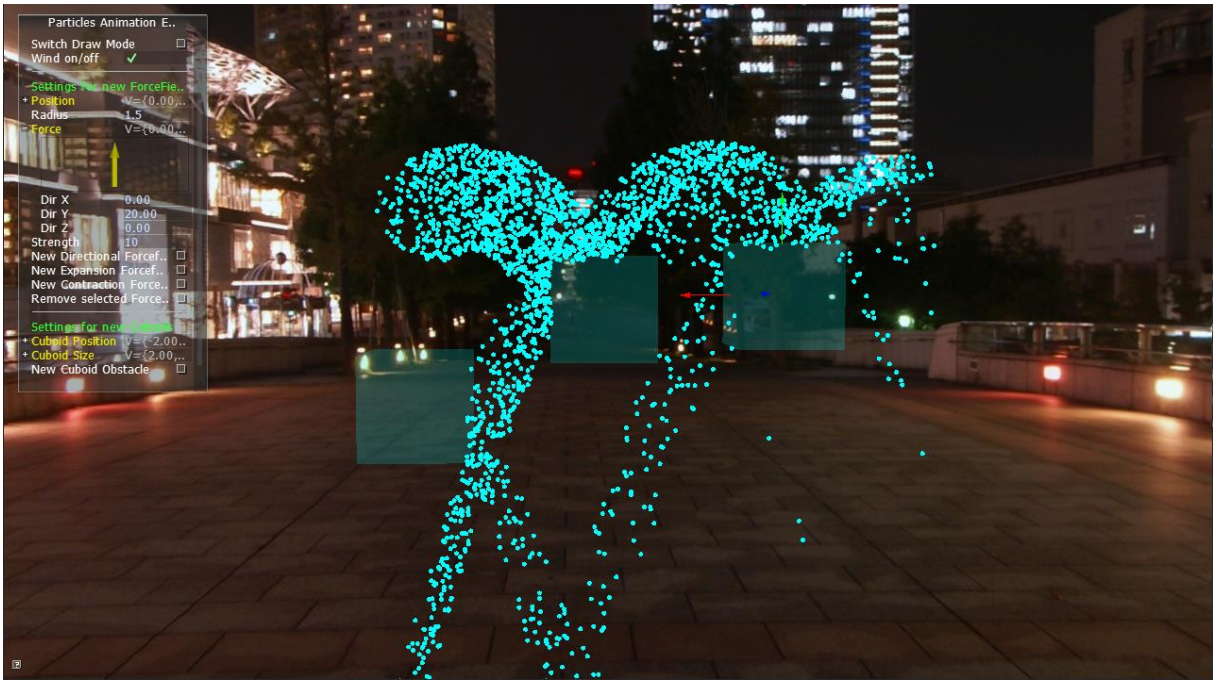
- The framework allows for up to 10 force fields of each kind.
- The position, radius and force of each forcefield is stored in structs:
  - directionalForceField
  - expansionForceField
  - contractionForceField
- The force fields are stored in arrays:
  - directionalForceFields
  - expansionForceFields
  - contractionForceFields.
- The number of active force fields is stored in the ints:
  - numDirectionalForceFields
  - numExpansionForceFields
  - numContractionForceFields

Task 3 – Collisions (3 points)

Files: Shaders/Particles/updateParticles.vert

Implement basic collision detection. If a collision is detected, simply reflect the velocity vector. You can ignore the radius of the particle.

Instead of simply reflecting the velocity vector, you can also correctly calculate the new position. You can ignore that the radius of the particles may intersect with the collision volume.



Task 4 – Showcasing + Theory (2 points)

Please explain in a sentence or two what the principle behind transform feedback buffers is in opengl (1 point).

Submit a forcefields-and-cubes-constellation that demonstrates the functionality of your program well. Include at least 4 force fields and 2 cubes (the particles should hit different faces of the cube). Show your constellation in a separate video (just the finished example, you do not need to “build” the scene in the video) (1 point). You can also increase the lifetime of your particles and change other parameters, include different colors, create another spawn point etc. (You do not need to do this for 1 point but you can if you like ;)).