Physics Simulation
CGP3012M

# Workshop #4: Beyond Rigidbodies

**Objectives**

1. Cloth simulation
2. Particle systems

**Guide time:**

4 hours (2 weeks)

**Tools / Libraries:**

1. PhysX, libraries and examples, installed & compiled.
2. Microsoft Visual Studio, 2019
3. The Visual Studio solution files

**Introduction**

The aim of these workshops is to introduce you to the practical aspects of real-time physics programming using the PhysX middleware. Each session is meant to serve as a tutorial for a topic in physics simulation with explanations and additional practical tasks which should help you to understand the theory covered during lectures.

This workshop will give you some guidance to help you to complete this week's tasks. These tasks are not assessed but are designed in a way which should lead you to the solutions required in the practical assessment for this module. To make the most out of these sessions, discuss your answers with your colleagues and/or ask demonstrators/favourite lecturer for feedback.

## Cloth simulation

Firstly, download and unpack the source code for Tutorial 4 which can be found on Blackboard. Additionally, you may wish to copy the code you have been working on in previous workshops into this project.

## Cloth Creation

For the sake of simplicity, we will use a rectangular piece of cloth in our simulations. However, more complex shapes are also possible. A cloth mesh is defined by a set of vertices, and can be built in a similar way to triangle meshes.

As a result, the creation of dynamic cloths requires a cooking procedure (as it is similar to a triangle mesh). A simple implementation is provides in the **`Cloth`** class, and allows you to specify two parameters:

- The cloth size (scale)
- The horizontal and vertical resolution (fidelity of cloth mesh)

Higher values for the horizontal and vertical resolution of the cloth result in a finer fidelity, and therefore a more accurate model of cloth physics. However, this does require more computational resources (because of the high resolution). In the example, the basic cloth object is already created in the scene, with one edge fixed to the world coordinates.

**Cloth Task 1:** Tweak the values specified in the **`Cloth`** constructor. What happens to the behaviour and appearance of the cloth when you double the number of vertices? What about if you half the number of vertices?

Does the number of vertices (and level of fidelity) influence the stiffness, simulation speed, or any other property of the cloth?

**Cloth Task 2:** It is also possible to change a default location of the cloth by using two methods:

- `PxCloth::setGlobalPose()`
- `PxCloth::setTargetPose()`

Now, increment the pose of the cloth continuously (moving it in a direction constantly). What is the difference between the two methods?

If you're struggling with this, try increase the increment distance, or check the SDK documentation to see the difference between the two.

## External Forces

A common task when simulating cloths is to apply external forces which will cause additional cloth movement. An example of an external force in this case might be the wind, which blows the cloth in a particular direction.

The simplest way of achieving this type of effect is by applying an external force with the `PxCloth::setExternalAcceleration()` function. This force is applied to each vertex in the cloth, along its normal direction only.

The parameter of this method accepts a 3D vector specifying both the direction and strength of the wind acceleration. Start with a `PxVec3(5.0f, 0.0f, 0.0f)` and tweak the values to see noticeable differences.

**Parameters**

The cloth class has a number of different parameters which influence the cloth's behaviour. Explore the following parameters, and see how different parameters affect the physics simulation. It is also advised to check the SDK documentation for further details about each of these functions/parameters:

- **Bending:** `PxCloth::setStretchConfig(PxClothFabricPhaseType::eBENDING, PxClothStretchConfig(value));`
- **Shearing:** `PxCloth::setStretchConfig(PxClothFabricPhaseType::eSHEARING, PxClothStretchConfig(value));`
- **Horizontal stiffness:** `PxCloth::setStretchConfig(PxClothFabricPhaseType::eHORIZONTAL, PxClothStretchConfig(value));`
- **Vertical stiffness:** `PxCloth::setStretchConfig(PxClothFabricPhaseType::eVERTICAL, PxClothStretchConfig(value));`
- **Friction:** `PxCloth::setFrictionCoefficient(value);`
- **Damping:** `PxCloth::setDampingCoefficient(value);`
- **Drag coefficient:** `PxCloth::setDragCoefficient(value);`

## Optional Task: Particles

To celebrate the last task in our series of workshops, we will explore a state-of-the-art particle simulator called Flex (**https://developer.nvidia.com/flex**). Flex is Nvidia's middleware for simulating different particle systems including fluids, cloths, deformable bodies, and so forth.

Flex is planned to be integrated into PhysX SDK, but for the time-being, it is a separate product from PhysX as Unity PlugIn. As a result, Flex comes with its own series of different demos, all of which are highly customisable with a number of simulation parameters that can be changed in real-time. To learn more about the different parameters, refer to the official Flex documentation: **http://docs.nvidia.com/gameworks/content/gameworkslibrary/physx/flex/**.

If you feel adventurous, you can try to build your own examples using Flex. Flex computations are very intensive (due to the amount of particles being simulated), and require a high-end GPU.

**Particles Task 1:** Explore different demos, and experiment with different simulation parameters. In particular, check out demos simulating fluids, cloth, etc. which are being covered during the lectures.