**Assignment 1**

In this assignment, you will use the Unity3D graphics engine to understand transformations, and apply this to implementing forward and inverse kinematics (Jacobian transpose and pseudo inverse methods) on the Kyle humanoid robot character. This assignment must be done with C# or JavaScript. The students can start from the following tutorial.
https://unity3d.com/learn/tutorials/s/roll-ball-tutorial

**Requirements.**

- Show that you can move the Kyle humanoid robot character by changing the joint angles (forward kinematics, FK) or the end effector positions (inverse kinematics, IK).
- You should try the Jacobian transpose and pseudo inverse method for the IK.
- Please attach a README file that explains about your source code and what you have actually done.
- Place your files in a single directory, and call the informatics electronic submission script for this course as follows :

  submit cav 1 your_directory_name

Submit: Deadline 4:00PM, 22nd February, 2018. It must be submitted electronically.

Tutor : Sebastian Starke: sebastian.starke@ed.ac.uk

Follow the steps below for doing the assignment.

#1 Download & Install Unity:
Windows & Mac: https://unity3d.com/de/get-unity/download
Linux: https://forum.unity.com/threads/unity-on-linux-release-notes-and-known-issues.350256 (Chose any version starting from Unity3d.2017. …)

#2 Create a new project and download the Kyle humanoid robot model from the asset store:
https://assetstore.unity.com/packages/3d/characters/robots/space-robot-kyle-4696
(Or simply search for it)

#3 Familarise yourself with the <Transform> class of Unity. Look at the online documentation and explain the difference between Transform.position / Transform.localPosition etc., and describe how one can be derived from another. Hint: You can easily validate via testing inside the Engine)

#4 Unity is component-based, where each component is a single script which is called automatically inside the game-update-loop. To test this, simply add a new component script to any game object inside the inspector field, and add some root transformation to Kyle with respect to the delta time. Press the “Play” button to see what happens. (You can also have a look at any of the various online tutorials)

#5 How do Quaternions work? How can the function Quaternion.Euler(…) be represented using an Quaternion.AngleAxis notation?

#6 Set up a script which serialises an array of transformations (game objects or transform components) for a serial kinematic chain (i.e. the arm or leg of the robot). Modify the transforms via script.

#7 What's the difference between forward and inverse kinematics (FK ↔ IK)? Give some examples where one might be preferred over the other. Explain the term "Degree of Freedom" (DOF).

#8 Implement the angle axis respresentation to compute forward kinematics while using 3 unconstrained axes (full 3 DOF joint) for each segment / bone.

#9 Try to make your chain end effector move a circular line within the Z or X plane by using joint values.

#10 Download the Matrix class from www.starke-consult.de/UoE/Matrix.cs, and import it somewhere into your project. Implement inverse kinematics using the Jacobian Tranpose method. Note: Position-only IK is sufficient, so you just need to solve the system for the end effector position and can ignore the rotation.

#11 Extend your implementation to use the Jacobian Pseudoinverse method instead of the Transpose. Why can't the pure inverse be used for that task?

#12 Extend your implementation to use the Jacobian Damped Least Squares method, and explain the new damping parameter.

#13 Compare those three method – what can you observe?
BONUS #1: Make your inverse kinematics usable for full-pose (position and orientation) goals.

BONUS #2: Extend your inverse kinematics solver to be usable for multi end effector systems. General proof-of-concept results are sufficient here, but if you can get it working super well, the better ;)

**Readings:**
*Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods – Samuel R. Buss, 2004*
https://groups.csail.mit.edu/drl/journal_club/papers/033005/buss-2004.pdf