Lab 8: Forward Kinematics

Objective

- To understand the following:
    - How forward kinematics works
    - How it is implemented
    - How to setup kinematic linkages
    - What local space is
    - What world space is

Goals

- Implement a “robot arm” using forward kinematics
- Use imgui to create an inspector in the spirit of Unity’s:

You may work individually or in groups of 2 - 5. Feel free to work in your GDW groups. Only one person needs to submit, make sure everyone’s names are in all of the submitted source files

Updates to Framework

- Keyboard input is now properly registered in ImGui, see the KeyboardInputFunction in main.cpp if you are interested in how I got it working
- WASD will move the camera around, click and drag the mouse to rotate the view
- The arrow keys will move the root object around

Theory

- In this lab you’ll be exploring how forward kinematics works and how it is implemented. You will be looking at the implementation provided in the source code, the theory presented in the lecture slides and building an intuition on how the two are related.
- The idea behind forward kinematics is simple:
    - We want to be able to express a transformation relative to some other gameobject
    - For example, if we wanted to put a hat on a character’s head, it’s much more practical to say “put the hat 2 units above the character’s head” than it is to try and position it based on some arbitrary coordinate system. To do this, we create a coordinate system at the character’s head, the origin (0,0,0) of this coordinate system would be the character’s head. We could then refer to points defined in this coordinate space as “head space” if we wanted to.

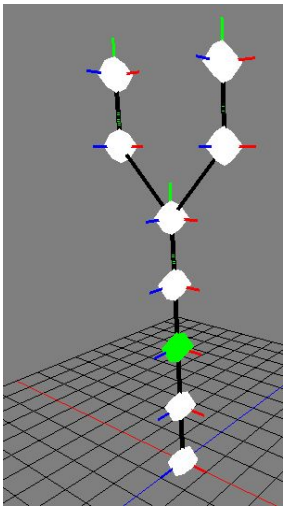
- Remember: the term “<blank> space” just refers to where the origin of the coordinate system is defined
    - Examples:
    - “World space” means points are expressed relative to the world origin
    - “<Joint> space” means points are expressed relative to some specific joint (i.e. “head space”)
    - “Camera space” means points are expressed relative to the camera (this is used extensively in computer graphics, it is sometimes referred to as “view space”).
- If this idea is unclear, please refer to the lecture slides for details

Itinerary

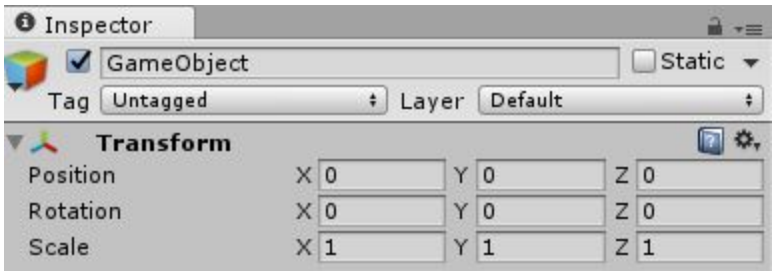
- Open up GameObject.h and take a look at the changes that were made:
    - All transformation attributes (position, rotation and scale) now have the word “local” prefixed on them
    - There is now a member variable for a “parent”
    - There is now a member variable for a list of “children”
- Open up GameObject.cpp and take a look at the update function
    - Investiagte how the new member variables come into play
    - Go through the logic in the update function and make sure you understand it before continuing. This is how forward kinematics is implemented!

- Open up main.cpp and navigate to the InitializeScene function, this is where the GameObjects are initialized and where the kinematic linkage is set up. “Kinematic linkage” is just a fancy way of saying “how are game objects related to each other”. Work out what this function is doing. Go into each member function and make sure you know what is happening before continuing.
- Navigate to the DisplayCallbackFunction and observe how the kinematic linkage is updated. Note that we only need to update and draw the root node. Ask yourself why that is, if you aren’t sure, go into the GameObject::Update and GameObject::Draw functions and see what’s happening. Refer to the lecture slides if needed.

- Now that you’ve got a grasp on what’s happening, it’s time to write some code!
- First, create a “robot arm” with at least 8 joints. It should look something like this:



- Next, use imgui to create an inspector in the spirit of Unity's:



- You should be able to enter values in the inspector and modify the currently selected joint, you only need to worry about the Position, Rotation (euler angles) and Scale attributes.

What's Next

- Forwards kinematics is a very important concept to understand
- Next week we will be looking at making these kinematic linkages in Blender and loading them into our project
- The week after next we will be looking at combining the motion of these bones with a mesh (aka skinning)

Submission

Submit a zip file the following:

1. Any source files you modify

Make sure you put your name and student number in a comment at the top of each file!

Failure to follow these submission guidelines will result in a **zero**!