Project 2: Hierarchical Animation Resources

Published | Edit | ⋮

GPR-450 Advanced Animation Programming
Instructor: Daniel S. Buckstein
Project 2: Hierarchical Animation Resources

Summary:
In recent classes we have explored keyframe animation control, forward kinematics and introductory hierarchical pose-to-pose animation. Most of the current difficulty is in manually setting up animation data. This project builds off lab 2 to further explore hierarchical animation as we delve into hierarchical animation file formats. This way we can load the data, represent it and control it using systems we have already implemented and used.

Submission:
Start your work immediately by ensuring your coursework repository is set up, and public. Create a new main branch for this assignment. ***Please work in pairs (see team sign-up) and submit the following once as a team***:

1. Names of contributors
   e.g. **Dan Buckstein**
2. A link to your public repository online
   e.g. **https://github.com/dbucksteinccorg/graphics2-coursework.git**
   (note: this not a real link)
3. The name of the branch that will hold the completed assignment
   e.g. **lab0-main**
4. A link to your video (see below) that can be viewed in a web browser.
   e.g. <insert link to video on YouTube, Google Drive, etc.>

Finally, please submit a ***10-minute max*** demo video of your project. Use the screen and audio capture software of your choice, e.g. Google Meet, to capture a demo of your project as if it were in-class. This should include at least the following, in enough detail to give a thorough idea of what you have created (hint: this is something you could potentially send to an employer so definitely show off your professionalism and don't minimize it):

- Show the final result of the project and any features implemented, with a voice over explaining what the user is doing.
- Show and explain any relevant contributions implemented in code and explain their purpose in the context of the assignment and course.
- Show and explain any systems source code implemented, i.e. in framework or application, and explain the purpose of the systems; this includes changes to existing source.
- **DO NOT AIM FOR PERFECTION, JUST GET THE POINT ACROSS**. Please mind the assignment rubric to make sure you have demonstrated enough to cover each category.
- **Please submit a link to a video visible in a web browser, e.g. YouTube or Google Drive.**

Objectives:
This project is an opportunity to build a pipeline for skeletal animation. We move away from hard-coded skeletal animation data and into reusable resources that help bridge the gap between artists and programmers. This is also a direct application of keyframe and clip control as seen in the last project.

Instructions & Requirements:
DO NOT begin programming until you have read through the complete instructions, bonus opportunities and standards, start to finish. Take notes and identify questions during this time. The only exception to this is whatever we do in class.
Complete the following tasks:

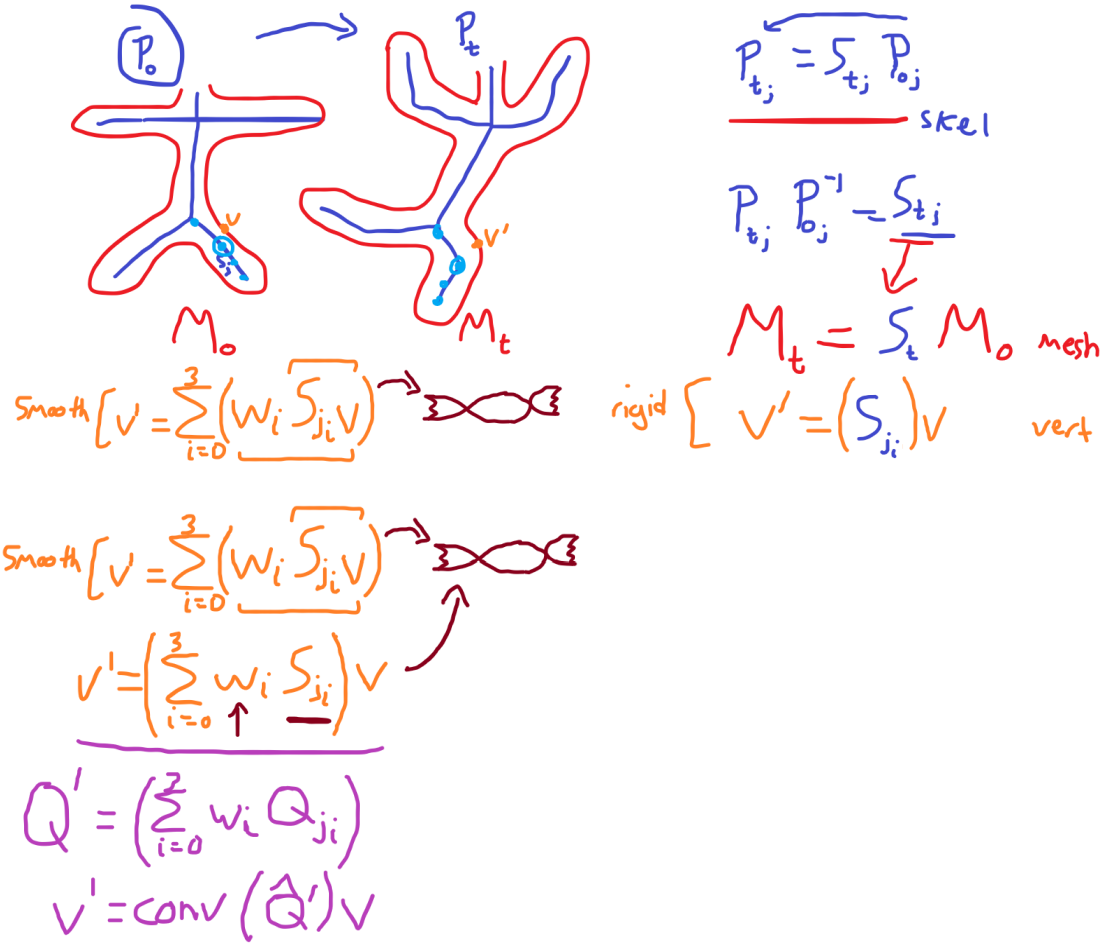
1. ***Pose-to-pose interpolation***: Extend the 'interpolation' phase of lab 2 to include the 'step', 'nearest', 'linear' and 'smoothstep' interpolation functions (we will discuss their implementations for spatial poses in class).
2. ***Animation resource tool***: Implement an importer or exporter for some ASCII hierarchical animation format. Here are some project suggestions:
    - Implement a BVH/HTR skeletal animation resource loader in the framework of your choice.
        - See function prototypes in animal3D as an example. The loader should have a minimum of 3 parameters: a new hierarchy (output), a new set of keyframe/pose data (output) and the file path (input). Display the resulting animated skeleton.
    - Implement a skeletal animation resource exporter from Maya (e.g. BVH/HTR).
        - Export a file in the desired format using Python. Fundamentally the opposite of the above: provide a target file path and write the hierarchy and pose data. In addition to the exporter, display the skeleton using your own implementation of forward kinematics.
    - Proprietary hierarchical animation format import/export (e.g. procedural data).
        - Propose an exporter/importer for another animation resource type... or invent your own! Use forward kinematics in some way.

Regardless of what you create, your system must support hierarchical pose-to-pose animation with a variety of interpolation functions (step, nearest, linear, smoothstep). Use what you made for lab/project 1 to help with this, and make improvements as needed. You must also implement a testing environment for this project:

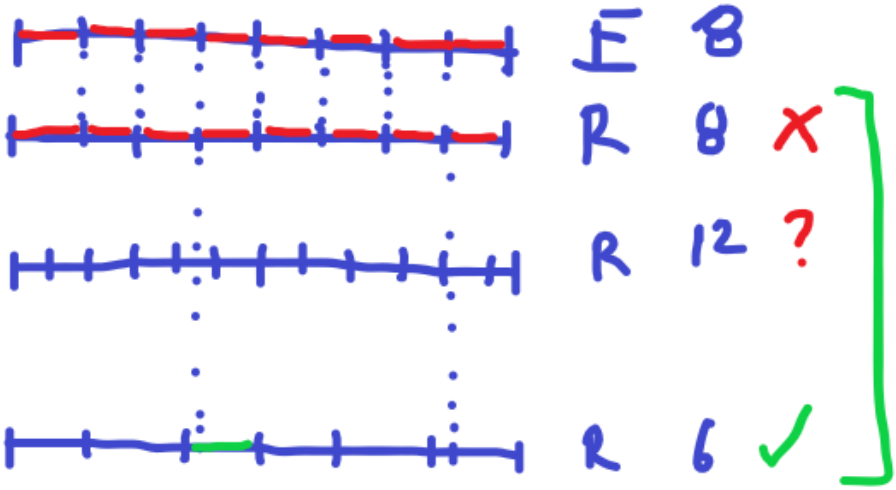
- Add a simple UI that allows the user to load/save data for the target resource.
- You must use your own forward kinematics implementation to display the skeleton at the current time (and optionally other states, e.g. flipping through keyframes).
  - Follow the 4-step formula: interpolation of "key/delta poses" which represent the pose keyframes; concatenation with "base pose" to set joint local poses; convert poses to joint-space (local) matrices; forward kinematics to calculate object-space (relative to root) matrices.

Bonus:
You are encouraged to complete one or more of the following bonus opportunities (rewards listed):

- ***Mesh skinning (+2)***: Complete the mesh skinning vertex shader using the pre-compiled animal3D skinning demo. The decoded starter code is in '*resource/glsl/4x/vs/00-common/passTangentBasis_skin_transform_vs4x.glsl*'; copy this file into the encoded shaders '*e/*' directory to edit in real-time while running the pre-built demo. We discussed the foundations of skinning in class during the second class of week 5. Use the diagram below and the skeletal application slides and the quaternions slides to help you.
  - ***The initial attempt at skinning must be demonstrated live (in office hours or another pre-determined appointment) before the beginning of the first class of week 6. Also include in your submission video.***
  - ***For the full bonus, complete dual quaternion (DQ) skinning and demonstrate a comparison between DQ and linear (matrix) skinning in your submission video.***



- ***Pre-sampling/keyframe reduction (+2)***: Implement additional or reduced keyframes depending on your creation:
  - If you are implementing an exporter, pre-sample all pose channels so that you are not just exporting a few keyframes, but rather many samples for the target framework/engine to use.
  - If you are implementing an importer, reduce the number of poses stored and use interpolation techniques to fill in the blanks. This is a form of reparameterization.
  - Here is a diagram visualizing a very low engine (E) framerate (8) and varying resource (R) framerates:



- ***Quaternions (+2)***: All pose rotations should be quaternion-based. Implement your own interpolation, concatenation and conversion functions. Add a 'real' component to the rotation member of the spatial pose structure (default = 1, negative denotes the other three channels are Euler angles), or an entirely separate 4D vector for quaternion orientation (default = 0,0,0,1).
  - Quaternions
  - Rotation and transformations
- ***Double-team (+3)***: Collaborate with another team to implement a functional asset pipeline: one team builds an exporter (e.g. from Maya) and the other builds an importer (e.g. into animal3D).
  - Each team should demonstrate their own contribution in detail and briefly show how it ties to the other (with evidence that the same asset is going from source to destination without tampering). Each team will be assigned their own respective grade.
  - In addition to the individual demonstration, produce a short (5 min) video with all collaborating members discussing the collaborative process, takeaways about the asset pipeline, etc. ***(+1 each)***.
  - Alternatively, build both an importer and exporter in the same engine/interface with a starter data set to prove that it can be exported and re-imported.

Coding Standards:
You are required to mind the following standards (penalties listed):

- ***Reminder: You may be referencing others' ideas and borrowing their code. Credit sources and provide a links wherever code is borrowed, and credit your instructor for the starter framework, even if it is adapted, modified or reworked (failure to cite sources and claiming source materials as one's own results in an instant final grade of F in the course)***. Recall that borrowed material, even when cited, is not your own and will not be counted for grades; therefore you must ensure that your assignment includes some of your own contributions and substantial modification from what is provided. ***This principle applies to all evaluations***.
- ***Reminder: You must use version control consistently (zero on organization)***. Commit after a small change set (e.g. completing a section in the book) and push to your repository once in a while. Use branches to separate features (e.g. a chapter in the book), merging back to the parent branch (dev) when you stabilize something.
- ***Visual programming interfaces (e.g. Blueprint) are forbidden (zero on assignment)***. The programming languages allowed are: C/C++, C# (Unity) and/or Python (Maya).
  - If you are using Unity, all front-end code must be implemented in C# (i.e. without the use of additional editors). You may implement and use your own C/C++ back-end plugin. The editor may be used strictly for UI (not the required algorithms).
  - If you are using Unreal, all code must be implemented directly in C/C++ (i.e. without Blueprint). Blueprints may be used strictly for UI (not the required algorithms).

- If you are using Maya, all code must be implemented in Python. You may implement and use your own C/C++ back-end plugin. Editor tools may be used for UI.
- You have been provided with a C-based framework called *animal3D* from your instructor.
- You may find another C/C++ based framework to use. Ask before using.

- ***The 'auto' keyword and other language equivalents are forbidden (-1 per instance)***. Determine and use the proper variable type of all objects. Be explicit and understand what your data represents. Example:
  - auto someNumber = 1.0f;
    - This is a float, so the correct line should be: float someFloat = 1.0f;
  - auto someListThing = vector<int>();
    - You already know from the constructor that it is a vector of integers; name it as such: vector<int> someVecOfInts = vector<int>();
  - Pro tip: If you don't like the vector syntax, use your own typedefs. Here's one to make the previous example more convenient:
    - typedef vector<int> vecInt;
    - vecInt someVecOfInts = vecInt();
- ***The 'for-each' loop syntax is forbidden (-1 per instance)***. Replace 'for each' loops with traditional 'for' loops: the loops provided have this syntax:
  - In C++: for (<object> : <set>)...
  - In C#: foreach (<object> in <set>)...
- ***Compiler warnings are forbidden (-1 per instance)***. Your starter project has warnings treated as errors so you must fix them in order to complete a build. ***Do not disable this***. Fix any silly errors or warnings for a nice, clean build. They are generally pretty clear but if you are confused please ask for help. Also be sure to test your work and product before submitting to ensure no warnings/errors made it through. This also applies to C# projects.
- ***Every section/block of code must be commented (-1 per ambiguous section/block)***. Clearly state the intent, the 'why' behind each section/block. This is to demonstrate that you can relate what you are doing to the subject matter.
- ***Add author information to the top of each code file (-1 for each omission)***. If you have a license, include the boiler plate template (fill it in with your own info) and add a separate block with: 1) the name and purpose of the file; and 2) a list of contributors and what they did.

Points	10
Submitting	a text entry box or a website url

Due	For	Available from	Until

Due	For	Available from	Until
-	Everyone	-	-

GraphicsAnimation-Master-Range-x2

Criteria	Ratings			Pts
IMPLEMENTATION: Architecture & Design Practical knowledge of C/C++/API/framework programming, engineering and architecture within the provided framework or engine.	**2 to >1.0 pts** **Full points** Strong evidence of efficient and functional C/C++/API/framework code implemented for this assignment; architecture, design and structure are largely both efficient and functional.	**1 to >0.0 pts** **Half points** Mild evidence of efficient and functional C/C++/API/framework code implemented for this assignment; architecture, design and structure are largely either efficient or functional.	**0 pts** **Zero points** Weak evidence of efficient and functional C/C++/API/framework code implemented for this assignment; architecture, design and structure are largely neither efficient nor functional.	2 pts
IMPLEMENTATION: Content & Material Practical knowledge of content relevant to the discipline and course (e.g. shaders and effects for graphics, animation algorithms and techniques, etc.).	**2 to >1.0 pts** **Full points** Strong evidence of efficient and functional course- and discipline-specific algorithms and techniques implemented for this assignment; discipline-relevant algorithms and techniques are largely both efficient and functional.	**1 to >0.0 pts** **Half points** Mild evidence of efficient and functional course- and discipline-specific algorithms and techniques implemented for this assignment; discipline-relevant algorithms and techniques are largely either efficient or functional.	**0 pts** **Zero points** Weak evidence of efficient and functional course- and discipline-specific algorithms and techniques implemented for this assignment; discipline-relevant algorithms and techniques are largely neither efficient nor functional.	2 pts
DEMONSTRATION: Presentation & Walkthrough Live presentation and walkthrough of code, implementation, contributions, etc.	**2 to >1.0 pts** **Full points** Strong evidence of accuracy and confidence in a live walkthrough of code discussing requirements and high-level contributions; walkthrough is largely both accurate and confident.	**1 to >0.0 pts** **Half points** Mild evidence of accuracy and confidence in a live walkthrough of code discussing requirements and high-level contributions; walkthrough is largely either accurate or confident.	**0 pts** **Zero points** Weak evidence of accuracy and confidence in a live walkthrough of code discussing requirements and high-level contributions; walkthrough is largely neither accurate nor confident.	2 pts
DEMONSTRATION: Product & Output Live showing and explanation of final working implementation, product and/or outputs.	**2 to >1.0 pts** **Full points** Strong evidence of correct and stable final product that runs as expected; end result is largely both correct and stable.	**1 to >0.0 pts** **Half points** Mild evidence of correct and stable final product that runs as expected; end result is largely either correct or stable.	**0 pts** **Zero points** Weak evidence of correct and stable final product that runs as expected; end result is largely neither correct nor stable.	2 pts

Criteria	Ratings			Pts
ORGANIZATION: Documentation & Management Overall developer communication practices, such as thorough documentation and use of version control.	**2 to >1.0 pts** **Full points** Strong evidence of thorough code documentation and commenting, and consistent organization and management with version control; project is largely both documented and organized.	**1 to >0.0 pts** **Half points** Mild evidence of thorough code documentation and commenting, and consistent organization and management with version control; project is largely either documented or organized.	**0 pts** **Zero points** Weak evidence of thorough code documentation and commenting, and consistent organization and management with version control; project is largely neither documented nor organized.	2 pts
BONUSES Bonus points may be awarded for extra credit contributions.	**0 pts** **Points awarded** If score is positive, points were awarded for extra credit contributions (see comments).		**0 pts** **Zero points**	0 pts
PENALTIES Penalty points may be deducted for coding standard violations.	**0 pts** **Points deducted** If score is negative, points were deducted for coding standard violations (see comments).		**0 pts** **Zero points**	0 pts
				Total Points: 10