Project 5: Intro to Animation Programming

Publish Edit ⋮

GPR-300 Intermediate Graphics & Animation Programming
Instructor: Daniel S. Buckstein
Project 5: Intro to Animation Programming

Summary:
In the previous project we explored interpolation and applications on both the CPU-side simulation and GPU-side display. In this project, we apply these principles again for the purposes of animating objects. We will explore GPU-based morph target animation, and skeletal animation and display.

Objectives:
Upon successful completion of this assignment, you will have accomplished the following:

- Revisit interpolation and keyframe control, this time for animating scene objects using morph targets and spooky scary skeletons.
- Explore morph target animation using a vertex shader.
- Explore forward kinematics and pose-to-pose animation for a skeleton.

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). Begin your submission immediately, copy the following into the text box, decide on your repository branch name for this assignment and fill in the information below.*** **The submission box locks at the specified deadline; be proactive and don't miss it. Late submissions, even by a minute, will not be accepted.**

Copy, edit and submit the following text once as a team (you do not need the headings, please just provide your info as shown in the examples here):

1. ***Names of contributors***: Write the names of the contributors of this assignment.
   e.g. **Dan Buckstein**
2. ***A link to your public repository online***: Grab the clone link from your working repository, which should end with "*.git*".
   e.g. **https://github.com/dbucksteinccorg/graphics2-coursework.git** (note: not a real link)
3. ***The name of the branch that will hold the completed assignment***: Create a new branch for this project, submit only the name of this branch.
   e.g. **project0-main**
4. ***A link to the read-me and user instructions for this project***: Ensure your repository includes a read-me that summarizes how to use your finished product.
   e.g. **https://github.com/dbucksteinccorg/graphics2-coursework/blob/project0-main/project0-readme.pdf** (note: not a real link)
5. ***A link to your video (see below) that can be viewed in a web browser***: Ensure your video is public or shared with your instructor.
   e.g. YouTube link: **https://www.youtube.com/watch?v=OqOyxQVs8IY** (note: "Attack on Game Development" by Will Gordon & Connor Breen)

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.**
- **Instead of waiting until last-minute for the video to upload, create a folder or links document on Drive that is accessible to your instructor. Submit the link to this folder as part of your official submission, and copy your video file/link there when it is ready.**

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.
Using the provided framework, implement the following:

1. **Setup framework**: Complete the following steps to ready the framework for this project:
   A. ***Project branch***: Check out the project starter branch "*graphics2/proj5*" to begin the project.
   B. ***Implement renderer utilities***: Here are the steps for setting up this demo:
      I. *Explore pre-built example*: Load the pre-built example demo for this project: "*File > Load demo > ...Proj5*". The demo shows the completed scene with procedural and loaded models, and the ability to toggle a variety of pipeline stages and features.
      II. *Explore renderer library*: The renderer library source is now available. Browse through the source for the "*animal3D-A3DG-OpenGL*" library.
         - Take a look at the header and interface for the drawable object, namely "*a3_VertexDrawable-OpenGL*" to better understand how models are drawn using OpenGL and the animal3D a3_VertexDrawable interface.
      III. *Initial build and run*: Either build the project in Visual Studio then launch using "*File > DEBUG... > Load without building*", or use "*Quick build*" to hot-build and run the demo directly through the player window.
      IV. *Additional setup*: Complete the following steps to make sure the framework is all set:
         - Navigate to the intro demo mode filter: "*Source Files/common/A3_DEMO/a3_DemoMode4_Animate*"; open the update source for this mode: "*a3_DemoMode4_Animate-idle-update.c*". Here you must update the different steps in pose-to-pose animation to properly visualize the skeleton.
2. **Implement shaders**: Using the pipeline implemented in part 1:
   A. ***Encode/decode shaders***: Upon successful completion of the render pipeline, using the encoded shaders should allow you to traverse the complete set of passes to see how the bloom effect is achieved. For your actual implementation, remove the "*e/*" from each shader file path.
      - With encoded shaders disabled (required), the first two modes (NM and POM) are identical, and the final mode shows empty space.
      - Since the last project, a bunch of the decoded shaders have been fully or partially implemented.
   B. ***Implement shader programs***: Implement the following effects by visiting and completing the following GLSL files (navigate to "*Resource Files/A3_DEMO/glsl/4x*"):
      I. *Morph target animation*: In olden days, animation on a 3D model or character was achieved using *morph targets*, which were fixed states of the mesh that could be blended to produce the illusion of animation. Here we explore a simplified GPU-based approach.

- Vertex shader: This program uses "*vs/04-animate/passTangentBasis_morph_transform_vs4x.glsl*" which is responsible for transforming the vertex based on a blend of morph targets. A morph target is simply a collection of attributes representing a "waypoint" (think abstractly) for a 3D model; i.e. a known state or keyframe for each attribute used in the shading algorithm. All attributes must be morphed appropriately to achieve the desired effect. This effect can occur in a vertex shader because it is the vertices that we deform to give the model its shape.

II. *Skeletal animation*: For this part we implement the fundamental algorithm for animating a skeleton. Skeletal animation is an improvement on morph targets because the skeleton represents a rig for skinning a mesh (which we will not cover here).

- Code setup: In "*a3_DemoMode4...update.c*", complete the tasks described in part 1, namely:
    - *Pose-to-pose*: The individual poses that can be animated represent changes from the *base pose*. Here, you must perform some interpolation between these poses to give the illusion of animation (in the pre-built demo, lerp is used). You should consider implementing a function to help perform the blending.
    - *Concatenation with base pose*: With the "delta" poses blended, we have the desired *change* from the base pose, which means to get the actual pose of the skeleton we must *concatenate* the result of the above step with the fixed base pose. The base pose describes the default pose of the skeleton as the fixed transformations between joints, and is sometimes referred to as "T-pose" or "A-pose".
    - *Conversion*: After determining the final pose for each joint, they must be converted to matrices.
    - *Forward kinematics*: The converted matrices represent each joint's transformation from their parent joint; *forward kinematics* is the algorithm used to calculate a new matrix for each joint in a common space (i.e. relative to the object as a whole). The matrix is calculated as the product of the parent's solved object-space matrix and the joint's local matrix, with a special case for the root joint.
- Vertex shader: This program uses the vertex shader "*vs/04-animate/passColor_hierarchy_transform_instanced_vs4x.glsl*" which is responsible for passing along a color based on each skeletal bone's *depth* in the hierarchy. Initially, bones are colored based on their *index* in the hierarchy.

3. **Testing & demonstration**: You must thoroughly test your render pipeline and shaders. Your demonstration must show evidence of the following:

   A. **Walkthrough and justification of architecture**: Demonstrate that the above requirements have been met in code.

B. **Framework**: Demonstrate completion of the pipeline setup using animal3D data structures and functions.
C. **Shaders**: Demonstrate the functional rendering pipeline and completion of the shaders implemented throughout the project. Furthermore, you must demonstrate that the shaders are your own and not the encoded files.
D. **Takeaways**: Discuss personal and professional takeaways from this project.

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

- ***Pose editor (+2)***: Implement an editor interface that allows the user to change skeletal pose values using keyboard and mouse input.
- ***More morphing (+2)***: Implement a model and shader program that would allow 16 morph targets using a simple color effect (e.g. uniform color); the user should be able to influence which targets is displayed using input.

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.
- ***Immediate mode is forbidden (zero on assignment)***: In this course we are studying modern graphics engineering principles; immediate mode refers to an ancient and deprecated set of OpenGL functions. The tutorials followed use the correct techniques; do not use tutorials on the internet that will lead you astray.

Points	10
Submitting	a text entry box

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