Final Project

Published | Edit | ⋮

GPR-450 Advanced Animation Programming
Instructor: Daniel S. Buckstein
Final Project

Summary:
The primary takeaways of this course are, in no particular order or rank, ***portfolio*** and ***engineering***. The final project is for you to demonstrate all you have learned in this course from a game programmer's lens and produce a portfolio-worthy project.

Submission:
Submit a link to your online repository with the completed assignment's branch name and commit ID/index. If you have not created an online repository to keep track of your work, you should do so as part of this assignment; it will be checked. ***Work in pairs***.

Instructions:

Design your own project that includes all of the following:

- Keyframe animation, clip/time control and management:
  - Outcomes of lab/project 1: Demonstrate understanding of general keyframe data structures (e.g. sample, keyframe, clip, clip controller, etc.) and algorithms (e.g. clip controller update, transitions, etc.).
  - Must integrate keyframes, clips and clip control.
  - Example concepts: decoupled clips and controllers; clip transitions and sequences; etc.
- Pose-to-pose animation, interpolation and hierarchical/skeletal animation:
  - Outcomes of lab/project 2: Demonstrate understanding of interpolation algorithms and functions, spatial pose data structures and algorithms.
  - Must integrate skeletal pose-to-pose, the setup for forward kinematics and the actual FK algorithm.
  - Example concepts: decoupled hierarchy and transforms; skinning; etc.

- A variety of blend operations:
  - Outcomes of lab/project 3: Demonstrate understanding of blending, layered animation and related data structures and algorithms.
  - Must integrate blend nodes and trees for poses and clips.
  - Example concepts: blend nodes for basically any spatial pose math function; decoupled tree data structure; operator-agnostic functions; etc.
- Character control:
  - Outcomes of lab/project 4: Demonstrate understanding of character interaction and animation, controlling animations in real-time and procedural animation algorithms.
  - Must integrate some form of real-time character control and procedural animation.
  - Example concepts: physics-based vs. interpolation-based integration; conditional blending and clip transitions; FK/IK blend; etc.

Please run your project ideas by the instructor well in advance. Include UML diagrams to help design your project and refine your idea. Examples of types of projects to consider:

- A tool useful for the animator or designer (e.g. an editor facilitating the art-to-screen pipeline for animators)
- A demonstration of an advanced topic of interest (e.g. a new, creative and interesting application of any of the course topics)
- A demonstration of a novel system that improves workflow (e.g. significant improvements on any of the systems covered in the course)

All components of the project must be functional and impressive; this is something you would want to show off at an interview... hence, this project is tied with a technical interview! Find a way to demonstrate the requirements as part of the technical interview (which involves a presentation). Implementing the bare minimum will earn you up to 80% on the project; go above and beyond for the remaining 20%.

Points 10

Submitting a text entry box or a website url

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