PA6 – Animation System *(now 80% easier)*

Student Information

Integrity Policy: All university integrity and class syllabus policies have been followed. I have neither given, nor received, nor have I tolerated others' use of unauthorized aid.

I understand and followed these policies: (•) Yes ( ) No

Name: Rachel Micolichek

Date: 11/28/23

Submission Details

Final ***Changelist*** number: 201380

Verified build: (•) Yes ( ) No

YouTube Link: https://youtu.be/VfEmdJL82Gg

Required Configurations: Any - just need one that works

Discussion (What did you learn):

I learned how tedious it is to do an animation if you don't data drive the values. I hard coded in the T and Q for the bones which resulted in thousands of lines of code and a lot of copy and paste. Next step will be to cut back on the repeated code and make it more manageable.

Verify Builds

- Follow the Piazza procedure on submission
  - Verify your submission compiles and works at the changelist number.
- Verify that only MINIMUM files are submitted
  - No – Generated files
    - *.pdb, *.suo, *.sdf, *.user, *.obj, *.exe, *.log, *.pdb, *.db, *.user
    - Anything that is generated by the compiler should not be included
  - No – Generated directories
    - /Debug, /Release, /Log, /ipch, /.vs
- Typical files project files that are required
  - *.sln, *.cpp, *.h
  - *.vcxproj, *.vcxproj.filters, *.vcxproj.user, CleanMe.bat

Standard Rules

Submit multiple times to Perforce

- Submit your work as you go to perforce several times (at least 5)
  - As soon as you get something working, submit to perforce
  - Have reasonable check-in comments
    - Points will be deducted if minimum is not reached

Write all programs in cross-platform C++

- Optimize for execution speed and robustness
- Working code doesn't mean full credit

Submission Report

- Fill out the submission Report
  - No report, no grade

Code and project needs to compile and run

- Make sure that your program compiles and runs
  - Warning level ALL …
  - NO Warnings or ERRORS
    - Your code should be squeaky clean.
  - Code needs to work "as-is".
    - No modifications to files or deleting files necessary to compile or run.
  - All your code must compile from perforce with no modifications.
    - Otherwise it's a 0, no exceptions

Project needs to run to completion

- If it crashes for any reason…
  - It will not be graded and you get a 0

No Containers

- NO STL allowed {Vector, Lists, Sets, etc...}
  - No automatic containers or arrays
  - You need to do this the old fashion way - ***YOU EARNED IT***

Leave Project Settings

- Do NOT change the project or warning level
  - Any changing of level or suppression of warnings is an integrity issue

Simple C++

- No modern C++
  - No Lambdas, Autos, templates, etc…
  - No Boost
- NO Streams
  - Used fopen, fread, fwrite…
- No code in MACROS
  - Code needs to be in cpp files to see and debug it easy
- ***Exception:***
  - implicit problem needs templates

Leaking Memory

- If the program leaks memory
  - There is a deduction of 20% of grade
- If a class creates an object using new/malloc
  - It is responsible for its deletion
- Any ***MEMORY*** dynamically allocated that isn't freed up is ***LEAKING***
  - Leaking is ***HORRIBLE***, so you lose points

No Debug code or files disabled

- Make sure the program is returned to the original state
  - If you added debug code, please return to original state
- If you disabled file, you need to re-enable the files
  - All files must be active to get credit.
  - Better to lose points for unit tests than to disable and lose all points

UnitTestConfiguration file (if provided) needs to be set by user

- Grading will be on the UnitTestConfiguration settings
  - Please explicitly set which tests you want graded… no regrading if set incorrectly

Due Date

- Due during on Finals - Week 11
  - Submit all files and directories to Perforce
    - **7 am - 27 November - NO late accepted at all**
- Submit all files and directories to Perforce
  - Create a directory called: Game Engine in your student directory
  - /student/<yourname>/PA6/…

Goals

- Write a standalone Animation Converter
  - Takes GLTF Animation/Models and exports to your custom animation format
    - Can be a separate converter that only does animation and not models
    - Submit your converter to student directory
- Export at ***least 4*** Animations from one skeleton (ChickenBot)
  - *You are allowed to do only _one_ model/skeleton*
    - More models with different skeletons/animations are appreciated (Jedi mode)
  - Idea to make this data driven without hand tweaking or modification
    - Look at the animation models directory
- Each skeleton needs to be at least 8 animated bones (ChickenBot)
  - Each skeleton has 4 or more animations
- Modify existing game engine to load Skeleton Model / Animation from binary format file
  - *You are allowed to HARD CODE data… Cut and Paste into cpp files*
  - Play the animation, pause, forwards, backwards, looping, fast / slow
  - Switch between animations

Assignments

1. **Animation Converter:**
   - Update your ***Anim/Model converter*** to convert Animation data into your custom run-time file format.
   - *Your converter can be loosely created (prototyping) to generate animation data*
     - Ideally its imported into a proto file… but for this assignment can be cut-n-paste into the cpp file of engine

2. **Export at *least 4* Animations, from one skeleton**
   - Ideally make this data driven without hand tweaking or modification
     - Animation needs to have many key frames (20 or more)
       - Export all key frames… do not skip any

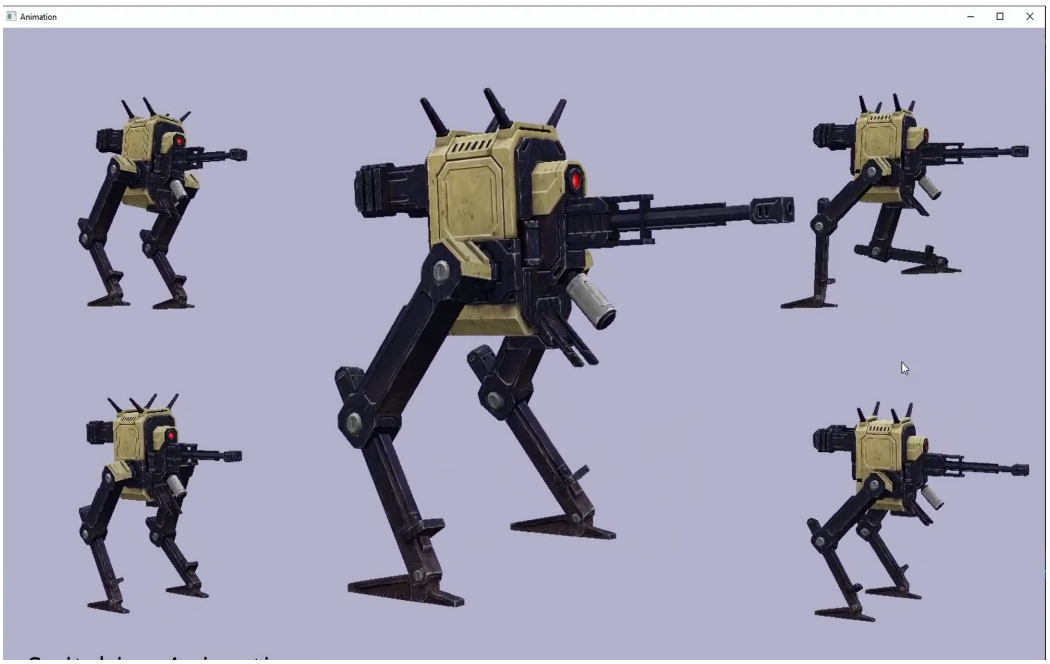
- Convert the data to your custom format
    - Try do as much off line processing to allow the game to display easily
- Each skeleton needs at least 8 animated bones
    - Skeletons need to be different
        - Suggestion: ChickenBot
- Ideas to help you convert
    - Export a hierarchy format
        - Understanding how each bone is related to each other
    - Export general information
        - Frames per Seconds
        - Number of frames
        - Number of bones
        - Etc.
    - For each Keyframe
        - Export the attributes of the bones
            - Translation
            - Rotation
            - Scaling

3. Game engine needs to be modified to read your custom binary file

- *Data can be hard coded…*
    - Remember:
        - you need every frame of data
        - for every animation (at least 4 of them)
- Engine should be able to load your animation files data
    - Add controls to play the animation
        - Play
        - Pause
        - Forward
        - Backwards
        - Fast or slow
        - Looping
        - Switching animations
            - Does not need to be blended with other animation
            - A hard switch is OK

4. DEMO - Video

- *Simple and sweet*
  - *5 instances all together on the screen*
    - *1 instance - Screen showing model in the Center*
      - *Cycle through 4 different animations (button key to switch animations)*
        - *i. Looping*
          - *1. Playing forward/reverse*
        - *ii. Faster and slower*
    - *4 instances of the ChickenBot in the corners*
      - *4 instances of mode on one screen*
      - *Each instance playing a different animation cycling*
  - *Great Demo from last year (only the animation portion of video)*
    - *https://www.youtube.com/watch?v=2a4k2np1ZeI&t=50s*
    - *Subtitling and music not needed*



5. **Write up document, video, and code**
    - **Code submitted**
        - Converter
        - Engine
        - With any converted data submitted to perforce
    - ***Record the demo***
        - Window with your audio commentary explaining the system
            - Quick show and tell the relevant code in your engine/converter
            - Show case your system by recording the demo
                - please post on YouTube with a link in your paper
        - Use any recording mechanism of your choice
            - As long as there is a YouTube link

Validation

Simple checklist to make sure that everything is checked in correctly

- Make sure program build and run without crashing
    - Converter
    - Game Engine
    - Animation data
- Documentation
    - Movie recording (5 – instances all together)
        - 4 instances (location in the corners playing different clips)
        - 1 instance in the center
            - demo slow/fast, reverse and switch to different animations
- Submit the data to perforce
    - Any source animation glb files that you used
    - Exported animation data of the model

Hints

- Do this assignment by iterating and slowly growing your project
    - You won't be able to finish this assignment in one day - Start now

Troubleshooting

- Focus Input format
    - Get a input format working first
    - Make sure you can load and display the format
    - Modify and change the two bone model example

- Focus on converter
    - Create data into an internal temporary format
        - Suggestion STL format
        - Print to a file
    - Next convert the data to your input format
- Extra step that REALLY helps
    - Create a standalone program that reads the input binary format
    - Read and print that to a file
    - This way you can visualize your data.
        - Way cool
        - Pragmatic programmer way

BABY STEPS!

- Time is your enemy, baby steps are key
    - Incremental development!