use Adobe Reader to complete

(Type in fields)

Particles – Project

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:	Jawan Luangnikone Davis
Date:	6/9/2022

Submission Details

Final ***Changelist*** number:	151042
Verified build:	[✓ Yes] No
(everyone) Logs.pdf:	student/jluangni/GameParticles/J
(grads) Analysis.pdf:	if applicable, Perforce location

Discussion (What did you learn):

I tried implementing what I've learned throughout the course into improving the performance of the project. Although I felt like I changed many lines of code, but there wasn't much improvement in the project. I implemented the techniques that were applicable and realized how difficult it is to maintain 16 byte alignment to utilize the SIMD intrinsics for the math. Overall this gave me an experience of working with "legacy code" where I'm given code to improve upon and I must figure out its behavior before making changes.

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, 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

Containers - ALLOWED

- You can have STL containers and arrays …
  - Be careful – might be a performance slowdown

Project Settings Modifications - ALLOWED

- You are permitted to change settings… Keep Level 4, be careful on SIMD level
- Nothing SSE 4.1 and below allowed → no AVX

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

Adding files to this project - ALLOWED

- Just be careful to add properly and submit the correct files
  - NO extra files

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 Dates

- See Piazza for due date and time
    - The particle system is due before by 7:00am CST – 17 March or earlier
        - Due on exam day (Thursday)
- Submit program perforce in your student directory assignment supplied.
- Fill out your this ***Submission Report*** and commit to perforce
    - ***ONLY*** use Adobe Reader to fill out form, all others will be rejected.
    - Fill out the form and discussion for full credit.

Goals

- Optimized a real world project
    - Refactor and improve an existing project
- Do not change any behavior
    - Number of particles, timing, randomness

Assignments

Please ***VERIFY*** the correct builds for each project

- ***Optimize for execution speed.***

Description

- Given a particle system, that dynamically updates several particles. Every particle and controlling object is dynamically allocated, often in very large bloated data structures. Each particle is controlled per frame by its own unique math transformations that are not optimized. The memory allocations in the system are slow and fragments memory as the number of particles increase. Some of the systems uses STL in a very inefficient manor. Many C++ classes are inefficient and naïve in nature.

Students refactor the project:

- Maximize the performance of particles to be processed
    - Try to keep the frame rate (update) constant
    - Try to keep the memory within a fixed specified size limit
- Real working system
    - Program needs to launch and close cleanly
    - *Error free at Warning level 4 or higher*
- No memory Leaking allowed
    - If you leak 20% deduction

- Graphics has to match the original.. no changing behavior
  - You are optimizing the program, not changing it.
  - Different behavior results in 50% deduction
- It's a contest - See how you can improve to original system.

Allowed to change compiler settings:

- Do not exceed SSE4.1
- Only use MMX, SSE
  - no AVX, VEX instructions

Submission date

- Students pay attention to the dates
  - The particle system is due before by 7:00am CST – 17 March or earlier
    - Due on exam day (Thursday)

Particle system

- BOTH Debug and Release modes need to build and run!
  - Failure – 0 grade
- Warning Level 4 needs to be SET, you cannot go below Warning Level 4
  - Keep warning level at 4
- Testing in Debug mode to check memory leaking
  - `CPU_WITH_GRAPHICS 1`
  - Leaks – 20% of grade
- Testing in Debug and Release to verify the graphics look and behavior
  - `CPU_WITH_GRAPHICS 1`
  - Failure to retain the original behavior – 50% of grade
- Testing in Release for timing
  - `CPU_WITH_GRAPHICS 0`

Required:

- **Graduate Students - Analysis Paper**
  - Graduate students must perform an optimization analysis for the particle system before they work on the system.
  - Write a 2-3 pdf paper detailing the current performance and a plan to improve the performance plan white paper.
    - Name the paper:
      - Analysis.pdf
      - Submit in the same directory as the solution file (GameParticles.sln)
    - Highlight your PLAN with expected improvement

- **Log paper (Everyone)**
  - Everyone needs to submit your log to perforce:
    - Submit as a PDF file.
  - Defining each major optimization and it relative performance
    - Include Dates and Changelist number
    - Expected length 2-3 pages in pdf format.

- Name the file:
  - Logs.pdf
  - Submit in the same directory as the solution file (GameParticles.sln)

- **Use the Submission report to help with logs….**
  - See next page

Sample log:

- Changelist: 90863 (Saturday Nov 5) date stored in changelist automatically
  - Added const to Vect4D
    - Saved 0.25 ms in release
  - Reworked Matrix to use references instead of pointer
    - Saved 0.05 ms in release
  - Added += operator in Vect4D
  - Reworked code to use this everywhere
    - Save 0.1 ms in release
- Changelist: 90872 (Monday Nov 12)
  - Added SIMD to Vect4D
    - Saved 2.5 ms
  - Cursed Keenan's Name out loud
    - His comments in code were wrong
    - Chased a red herring for 4 hours
  - Sutter come-on give a little more help.

Grading

- Project MUST build and Run
  - You cannot change the OpenGLDevice code
  - Exception:
    - Allowed to call the float number types if needed
    - Camera Matrix
      - static void SetCameraMatrixDouble(const double *);
      - static void SetCameraMatrixFloat(const float *);
    - Transform Matrix
      - static void SetTransformMatrixDouble(const double *);
      - static void SetTransformMatrixFloat(const float *);
- Ignoring warnings... if it builds (in Release mode) that's good enough
  - You can change compiler settings... SSE4.1 or lower is allowed
- Grading Ratio
  - Run in Debug for memory leaks check
  - Run in Release - original and each students particle system for timing
  - I will grade on performance ratio:
    - For example
      - if Original = 66ms, student's = 33ms

        - Ratio: 66/33 = 2.0x faster
- The original behavior needs to be preserved
    - Number of particles and lifetime needs to be set to the original for grading.

Grading TABLE:

Performance Ratio:

> 4.00 :	10.0 pts
> 3.75:	9.5 pts
> 3.50:	9.0 pts
> 3.25:	8.5 pts
> 3.00:	8.0 pts
> 2.75:	7.5 pts
> 2.50:	7.0 pts
> 2.25:	6.5 pts
> 2.00:	6.0 pts
> 1.75:	5.5 pts
> 1.50:	5.0 pts
0 - 1.50:	0.0 pts ←------- NOTE you have to 50% improvement for any points

Validation

Simple checklist to make sure that everything is submitted correctly

- Is the project compiling and running without any errors or warnings?
- Does the project run ***ALL*** the unit tests execute without crashing?
- Is the submission report filled in and submitted to perforce?
    - Logs paper
    - Analysis paper (grad students)
- Follow the verification process for perforce
    - Is all the code there and compiles "as-is"?
    - No extra files
- Is the project leaking memory?

Hints

Most assignments will have hints in a section like this.

- Do many little check-ins
    - Iteration is easy and it helps.
    - Perforce is good at it.
- Use piazza forums
    - Share ideas… but not that many
        - You want to win the prize!