Sprint5 – ASync Load

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: 06/05/22

Submission Details

Final ***Changelist*** number: 150339

Verified build: (•) Yes ( ) No

Required Configurations: x86 - DEBUG (only)

YouTubeLink: https://youtu.be/HINvjYDeMfs

Discussion (What did you learn):

I struggled with the asynchronous loading. While I understood what it meant, actually setting it up was a whole different story. I ended up utilizing your code and walking through it and the lecture several times until I could understand how it was moving.

YouTube Process

- Record the YouTube demo
  - You need to record in stereo with commentary
    - 2 channel with both computer (desktop) and microphone recording
  - Suggestion: ***OBS*** screen capture
- Record the desktop (enough to show your directory and the visual studio and output)
  - Show your directory in recording
    - Launch the visual studio (double click solution)
  - Show off relevant parts of the code with commentary
  - Launch and demo the Sprint
    - Play the demo and add your commentary in real-time
  - Watch your video
    - Verify that video clear and can you hear the commentary with audio in stereo?
- Note: Weekly Sprints cannot be longer that 2:00 mins
  - If you go over… do it again
- Publish your YouTube recording
  - Make sure it is accessible without any login or permission to play
  - It can be private but not restrictive to play by anyone with the link
    - If unplayable as-is… Grade 0
- Submit your code to perforce to the appropriate Sprint directory
  - Verify it

Pdf form (this document)

- *Submit this PDF to perforce*
  - *Fill in form*
    - *Name, changlelist, etc…*
  - *Submit back to perforce*
    - *Check it out*
    - *Submit it back to perforce to the same location*

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

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

No Adding files to this project

- This project will work “as-is” do not add files…
- Grading system will overwrite project settings and will ignore any student’s added files and will returned program to the original state

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
- Submit program perforce in your student directory assignment supplied.
- Fill out your this ***Submission Report*** and ***Sprint*** 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

- Learn
  - Async loading
    - You need to use FileSlow library
  - Creating a file thread with a queue
  - Callback attached to the async loading
    - Signaling the game thread when done loading

Assignments

0. Create a directory Sprint5

a. Do all your development in Sprint5 directory
b. You need to use FileSlow methods for Sprint 5, 6 and milestone2
c. Make sure you submit this project many times to perforce as you develop
   i. You need to submit the project and the video for this Sprint

Setup:

- 3 samples
  - Given 2 simple mono wave samples
    - Sampled at 48Khz, 32-bit
  - Given 1 stereo wave sample
    - Sampled at 48Khz, 32-bit
- Create 3 separate simple playlists (scripts) – 1 sound wave, one sound ID
  - 501 – Electro
  - 502 – Alert
  - 503 – Beethoven

- We are demoing Asynchronous loading and game user file load callback
  - Create 1 unique game user file load callback
    - Using Debug::out() to the output window wave file name loaded
    - Example:
      - Beethoven.wav → Loaded
    - Callback is triggered when the file load is finished
  - Goal of this demo
    - Load 2 sound waves 501, 502 initiated on the game thread
    - Start playing sounds 501 and 502
    - After starting 501
      - Load 1 sound wave asynchronously
      - Once loaded – a callback will be triggered (communicating its loaded)
- Call the SPECIAL loading file loading functions
  - Since many have Solid State Drive… we need to simulate
    - Delay and latency of network or slow hard drive
  - ***Use the FileSlow class to simulate latency***
    - ***FileSlow::Open()***
    - ***FileSlow::Read()***
    - ***FileSlow::Seek()***
    - ***FileSlow::Tell()***
    - ***FileSlow::Close()***

Demo:

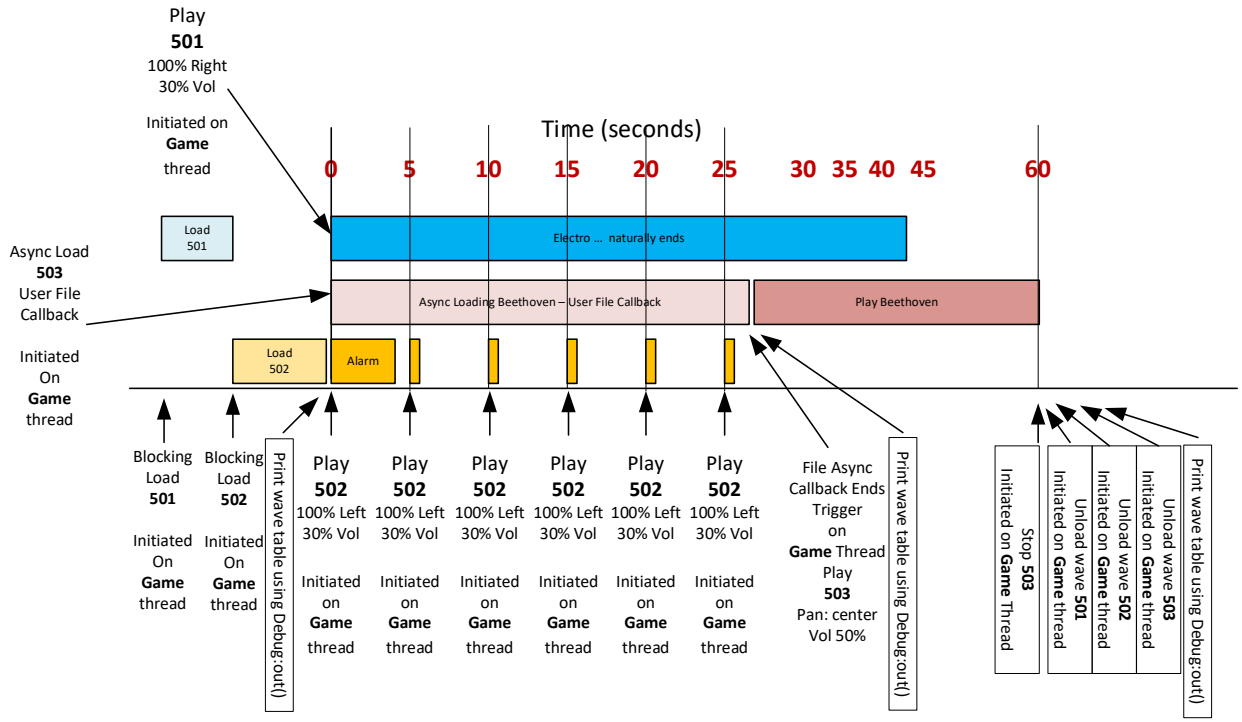
Start Demo – hit the <SPACE> key to trigger it

- This is triggered in the update() method of the game
  - Read the keyboard input
  - Then load and go with your Demo
- The demo should play from there.
  - No user intervention needed – just need the timer triggers working.

Load:

- Setup your playlists
  - Load the mono wave data needed for 501 and 502 initiate on the game side
  - ***DO NOT LOAD*** 503 - Beethoven
    - It's OK to have the playlists table on the Audio Thread side
  - Audio side cannot load the wave data, that has to be initiated on the game side
- Create game user file loading callback something like GameLoadingCallback()
  - This will be used on the load call in the demo
    - Printing to the output window wave file name loaded when file is finished loaded
- Load all the timer events for this demo at once – let the timer do the work

Start the demo



- In Demo (Start with a key press)
  - Blocking Loading Snd 501
    - Initiate on Game Thread
    - Add Debug::out() to show the call on the correct thread
  - Blocking Loading Snd 502
    - Initiate on Game Thread
    - Add Debug::out() to show the call on the correct thread
  - Print the wave table
    - Use Debug::out() in the wave table printing;
  - When loaded start time demo

- Timer: 0 seconds
  - SndA = Play 501, vol: 30%, pan: 100% Right, Priority default (optional)
    - Add Debug::out() to show the call on the correct thread
  - Start wave loading async data with GameLoadingCallback()
    - Game thread initiates the Beethoven wave data load
      - The callback is created on game side
      - Will be triggered when that wave data (Beethoven is loaded)
    - Add Debug::out() to show the call on the correct thread
  - SndB = Play 502, vol: 30%, pan: 100% left, Priority default (optional)
    - Add Debug::out() to show the call on the correct thread

- Timer: 5 seconds
  - SndB = Play 502, vol: 30%, pan: 100% left, Priority default (optional)
    - Add Debug::out() to show the call on the correct thread

- Timer: 10 seconds
  - SndB = Play 502, vol: 30%, pan: 100% left, Priority default (optional)
    - Add Debug::out() to show the call on the correct thread

- Timer: 15 seconds
  - SndB = Play 502, vol: 30%, pan: 100% left, Priority default (optional)
    - Add Debug::out() to show the call on the correct thread

- Timer: 20 seconds
  - SndB = Play 502, vol: 30%, pan: 100% left, Priority default (optional)
    - Add Debug::out() to show the call on the correct thread

- Timer: 25 seconds
  - SndB = Play 502, vol: 30%, pan: 100% left, Priority default (optional)
    - Add Debug::out() to show the call on the correct thread

- As soon as the Beethoven is loaded… from the callback
  - Start the Beethoven sound
  - SndC = Play 503, vol: 50%, pan: center, stereo, Priority default (optional)
    - Beethoven should start
    - Debug::out() in the callback
  - Print the wave table

- Timer: 60 seconds
  - Stop SndC
  - Initiated from Game side
    - Unload(501 wave);
    - Unload(502 wave);
    - Unload(503 wave);
  - Print the wave table

Nothing else – the callbacks to the work

Questions:

Place in a separate PDF call Sprint5_Questions, in the same directory as the Sprint5 PDF

1) **Please explain and diagram the way you did Async user File Callback?**
   a. *Talk about the commands, threads and how the wave table is protected*
2) **How does a wave table entry gets delete/removed?**

a. *Talk about each scenario. (unload command)*

Deliverables

- Stand-alone C++ demo
  - Create a demo to show off the ***ALL*** of the above features
  - Use audio samples that allow you to demonstrate the above features easily

- Visual Studio 2019 Enterprise Edition
  - C++ warning level all
  - Minimum code, no temporaries or generated items
  - Needs to be able to compile and run "as-is" without checking out from perforce or changing the attributes of the files
- For some people – the demo is hardest part of this exercise

Validation

Simple checklist to make sure that everything is submitted correctly

- Submitted project to perforce correctly
  - Is the project compiling and running without any errors or warnings?
  - Is the submission report filled in and submitted to perforce?
  - Follow the verification process for perforce
    - Is all the code there and compiles "as-is"?
    - No extra files
  - Is the project leaking memory?
- Submitted the YouTube link to perforce?

Hints

Most assignments will have hints in a section like this.

- Dig into the material read the online blogs…
  - Lots and lots of information
- You can discuss the tools and drivers on Piazza
  - Share
- Use the Piazza FORUMs
  - Read, explore, ask questions