use Adobe Reader to complete

(Type in fields)

Sprint6 – Priority Table

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:	06/12/22

Submission Details

Final ***Changelist*** number:	151402
Verified build:	[✓ Yes] No
Required Configurations:	x86 - DEBUG (only)
YouTubeLink:	https://youtu.be/iQ0vaxwTv0g

Discussion (What did you learn):

For the priority table, I learned about how to calculate the running time. When a sound is entered into the table, that entry time is recorded as the start time. When the priority table is printed, we calculate the difference between the requested print time and the entry start time.

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 10:00 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*

use Adobe Reader to complete

(Type in fields)

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 Sprint6

a. Use project from previous sprint or milestone as a starting point
   i. Do all your development in Sprint6
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:

- Given 1 simple mono wave samples
  - Sampled at 48Khz, 32-bit
- Create 1 separate simple playlists (scripts) – one sound wave, one sound ID
  - 301 – Coma
- For DEMO reasons we are allowing a maximum of 6 sound calls at a time
  - Otherwise this demo would be 2x longer

- You need the ability to print to the output window the current status of each sound call
    - Snd Handle, priority, time playing
        - Need handles – unique identifier
    - For example: (3 handles – in the active table)
        - ------- Active Table ------------------
        - 0xAAAA0001: 10 1500 ms
        - 0xAAAA0004: 50 1500 ms
        - 0xAAAA0005: 75 200 ms
        - Use Debug::out()
            - This shows the thread name as well…

- Sound call for this demo is more of a placeholder
    - Keep the volume down to 10% for all of these call
    - Lower number is the higher priority
        - Example: Snd_A 50 priority kills a Snd_X 75 priority
    - If Snd_A new call has the same priority of existing Snd_X's priority,
        - Kill the oldest sound call with the same priorty

- 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 FILE_SLOW class to simulate latency***
        - ***FileSlow::Open()***
        - ***FileSlow::Read()***
        - ***FileSlow::Seek()***
        - ***FileSlow::Tell()***
        - ***FileSlow::Close()***

Demo:

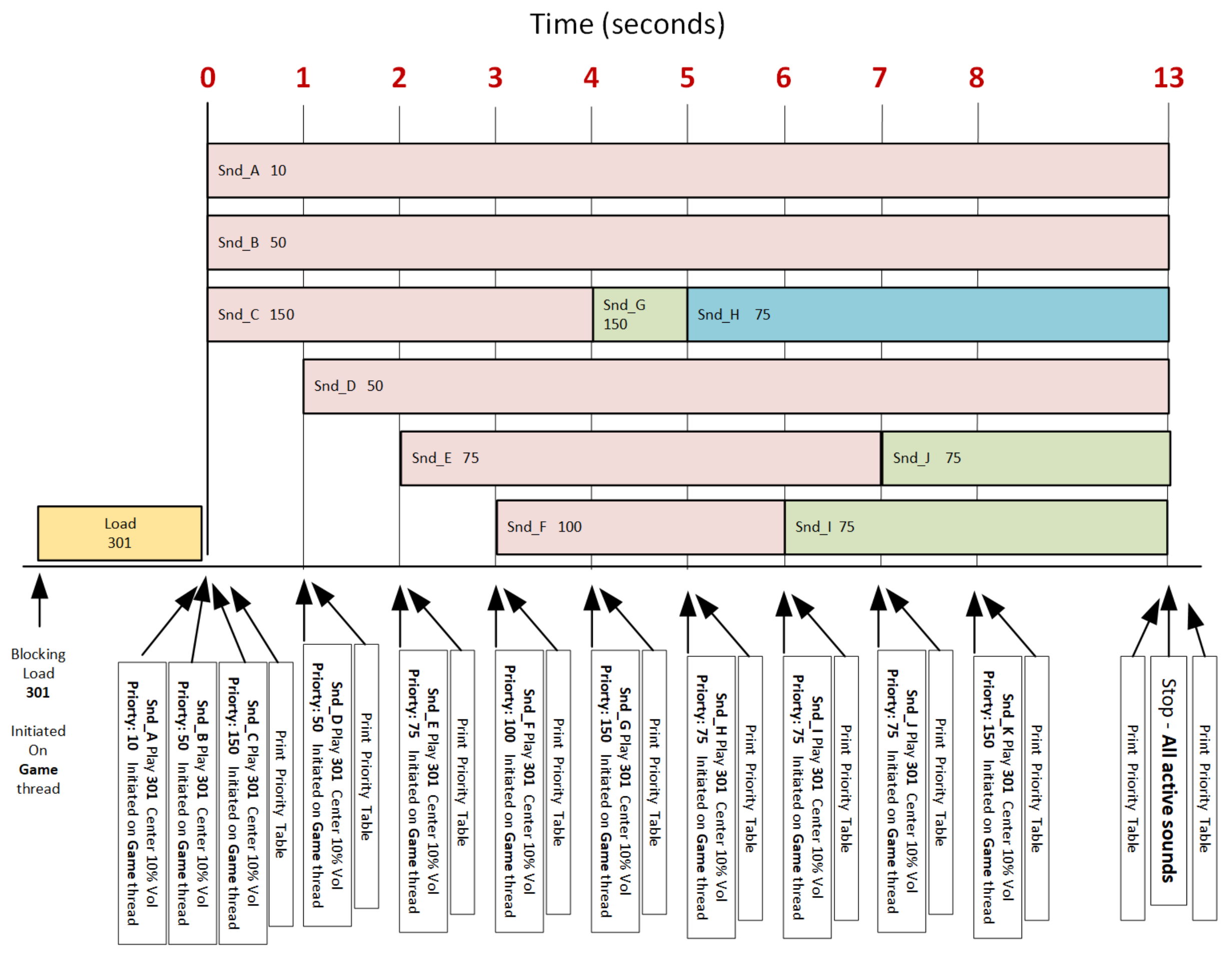
Start Demo –hit the <SPACE> key to trigger it

- This is triggered in the update() method of the game
    - Read the keyboard input
    - Then load and go with your Demo 3
- The demo should play from there.
    - No user intervention needed – just need the timer triggers working.

Load:

- Setup your playlists
    - Load all the mono wave data for 301 initiate on the game side
    - It's OK to have the playlist table on the Audio Thread side
    - But it cannot load the wave data, that has to be initiated on the game side
- Load all the timer events for this demo at once – let the timer do the work

Start Demo


Time (seconds)
0 1 2 3 4 5 6 7 8 13
Snd_A 10
Snd_B 50
Snd_C 150
Snd_G 150
Snd_H 75
Snd_D 50
Snd_E 75
Snd_J 75
Load 301
Snd_F 100
Snd_I 75
Blocking Load 301
Initiated On Game thread
Snd_A Play 301 Center 10% Vol Priorty: 10 Initiated on Game thread
Snd_B Play 301 Center 10% Vol Priorty: 50 Initiated on Game thread
Snd_C Play 301 Center 10% Vol Priorty: 150 Initiated on Game thread
Print Priority Table
Snd_D Play 301 Center 10% Vol Priorty: 50 Initiated on Game thread
Print Priority Table
Snd_E Play 301 Center 10% Vol Priorty: 75 Initiated on Game thread
Print Priority Table
Snd_F Play 301 Center 10% Vol Priorty: 100 Initiated on Game thread
Print Priority Table
Snd_G Play 301 Center 10% Vol Priorty: 150 Initiated on Game thread
Print Priority Table
Snd_H Play 301 Center 10% Vol Priorty: 75 Initiated on Game thread
Print Priority Table
Snd_I Play 301 Center 10% Vol Priorty: 75 Initiated on Game thread
Print Priority Table
Snd_J Play 301 Center 10% Vol Priorty: 75 Initiated on Game thread
Print Priority Table
Snd_K Play 301 Center 10% Vol Priorty: 150 Initiated on Game thread
Print Priority Table
Print Priority Table
Stop - All active sounds
Print Priority Table

Part A: Load sounds at specific times and priorities (print sound table status)

- In Demo (Start with a key press)
  - Blocking Loading Snd 301
    - Initiate on Game Thread
    - Add Debug::out() to show the call on the correct thread
  - When loaded start time demo

- Timer: 0 seconds
  - Snd_A = Play 301 with priority:10 vol: 10%
  - Snd_B = Play 301 with priority:50 vol: 10%
  - Snd_C = Play 301 with priority:150 vol: 10%
  - → Print the status of the active sound call table (see example)
    - Remember Use Debug::out()

- Timer: 1 seconds
  - Snd_D = Play 301 with priority:50 vol: 10%
  - → Print the status of the active sound call table (see example)

- Timer: 2 seconds
    - Snd_E = Play 301 with priority:75 vol: 10%
    - → Print the status of the active sound call table (see example)
- Timer: 3 seconds
    - Snd_F = Play 301 with priority:100 vol: 10%
    - → Print the status of the active sound call table (see example)
- Timer: 4 seconds
    - Snd_G = Play 301 with priority:150 vol: 10%
    - → Print the status of the active sound call table (see example)
- Timer: 5 seconds
    - Snd_H = Play 301 with priority:75 vol: 10%
    - → Print the status of the active sound call table (see example)
- Timer: 6 seconds
    - Snd_I = Play 301 with priority:75 vol: 10%
    - → Print the status of the active sound call table (see example)
- Timer: 7 seconds
    - Snd_J = Play 301 with priority:75 vol: 10%
    - → Print the status of the active sound call table (see example)
- Timer: 8 seconds
    - Snd_K = Play 301 with priority:150 vol: 10%
    - → Print the status of the active sound call table (see example)
- Timer: 13 seconds
    - → Print the status of the active sound call table (see example)
    - Stop all pending sounds
    - → Print AGAIN the status of the active sound call table (see example)

Questions:

Place in a separate PDF call Sprint6_Questions, in the same directory as the Sprint6 PDF

1) **Please explain and diagram the way you update the time in the priority table?**
    a. *Talk about the commands, threads and how the table is protected*
2) **How does a priority table entry gets delete/removed?**
    a. *Talk about each scenario. (Snd naturally ends, Stop, priority Kill)*

use Adobe Reader to complete
(Type in fields)

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