use Adobe Reader to complete

(Type in fields)

Sprint3 – Handles

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:	04/30/22

Submission Details

Final ***Changelist*** number:	145249
Verified build:	[✓ Yes] No
Required Configurations:	x86 - DEBUG (only)
YouTubeLink:	https://youtu.be/Cn-wNcQjCiU

Discussion (What did you learn):

This sprint I learned about handles and really put the mutex into play for the circular queue. Previously, I wasn't using the queue so it took a while to set that up. Getting it to play a sound wasn't too hard, but it took some time to get the call operations working. Once I was able to get the volume to change, pitch and panning were pretty easy to copy.

YouTube Process

- Record the YouTube demo
  - You need to record in stereo with commentary
    - 2 channel with both computer (desktop) and microphone recording
  - Use: ***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
  - Handles – creating a custom handle system
  - Add handles to control resources
    - Create your own custom voice object using handles
  - Use a circular queue to communicate to thread
    - Pass data to and from a thread using a circular queue

Assignments

0. ***Setup directory Sprint 3***
   a. Copy your contents of your Sprint2 directory into Sprint3
      i. Do all your development in Sprint3 for this Sprint
      ii. Sprint3 has the new libraries we need to use
   b. 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

1. ***Research Handles***
   a. Experiment and create a handle system
   b. Create a custom handle system
      i. Review the material from class
      ii. Experiment with the handle system
      iii. Improve and complete the system so you can add handles easily to resources
   c. Add protection around the handle systems (mutexes)
      i. To avoid any race conditions

2. ***Research and experiment with circular queues***
   a. Take examples from the class or make your own
   b. Improve on the design, add safe guards for race conditions
      i. Write code to test over flowing the queue.
      ii. Draining the queue completely

iii. Overload operators to make the environment easier to develop

3. ***Develop a 1st pass of audio engine***
    a. Audio engine needs to be in its own thread
    b. Audio engine needs to have its communication between threads going through circular queues.
    c. Audio system should initialized and stand ready for commands.

4. ***Audio engine commands***
    a. Commands are coming through the circular queue.
    b. Required commands
        i. Load a sound resource – need a manager to hold resources
            1. Create a voice buffer from a file
            2. Note – this will require a manager for the resources
        ii. Create Sound Call
            1. Creates a sound call and returns a handle
                a. This handle - is how you play, stop, change volume, etc.
            2. Eventually this will be a script… (week 8)
                a. A specific sound call is a command that
                    i. Create a Voice
                    ii. Associate a buffer to the voice
            3. Can be hard set.
                a. Sound call A – always plays an oboe
                b. Sound call B – always plays a violin
        iii. Simple commands with handle
            1. Start a sound call with a handle
            2. Stop a sound call with a handle
            3. Set Volume with a handle
            4. Get Volume with a handle
            5. Add panning and pausing
    c. Eventually more functionality…

5. ***Create an Audio Demo***
    a. Create an Audio engine on its own thread
    b. Sound calls are communicated through handles
    c. Call several sound calls and change their behaviors on the Game thread
        i. To make the demo easier… you can create additional threads if you want
    d. Some behaviors….
        i. For a specific sound call: Starting, stopping, changing volume
    e. Use 3-5 sound calls demoing the functionality.
    f. Expectation of the demo is around 5 minutes (max 10min) long for this Sprint

i. If over 10:00 min… re-record

6. ***Discuss on YouTube Demo***
    a. Build and explain the code
    b. Run the demo
    c. Make sure you record in stereo
7. ***Deliverables***
    a. Stand-alone C++ demo
        a. Create a demo to show off the ***ALL*** of the above features
        b. Use audio samples that allow you to demonstrate the above features easily
    b. Visual Studio 2019 Enterprise Edition
        a. C++ warning level ALL
        b. Minimum code, no temporaries or generated items
        c. Needs to be able to compile and run "as-is" without checking out from perforce or changing the attributes of the files
    c. For some people – the demo is hardest part of this exercise
    d. YouTube recording
    e. Stand-alone C++ demo
8. ***Grading***
    a. All of the above requirements
    b. ***Highlights*** *(don't forget these)*
        i. Handle system
            1. Make sure you add mutexes for race condition protection
            2. Discuss – Why is the mutex needed?
        ii. Audio system on separate thread
            1. ALL communication through circular queues
            2. Check overflow detection
                a. Demo the overflow happening
                b. Keep your queue small to show overflow
        iii. Snd Call - Class
            1. Loads sounds from waves
                a. No need to load the same wave from the file multiple times…
                b. Have a manager register the wave and reuse
                    i. Demo the reuse
            2. Controls sounds from a call
            3. Every snd call has a handle
                a. Play, stop, pause, volume, panning the snd call interface
                b. Using handles for protection

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 in PDF?

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