Milestone 1

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: 05/25/2022

Submission Details

Final ***Changelist*** number: 148543

Verified build: (●) Yes ( ) No

Required Configurations: DEBUG

YouTube Link: https://youtu.be/l9qD0WFoLsk

Self Grading Section:

Required Features:

- ☑ Basics Audio Playback
  - ☑ XAudio2 Voices
  - ☑ Attributes (volume, pan, pitch, etc.)
  - ☑ Load / Unloading waves
  - ☑ XAudio2 Callbacks
- ☑ Audio Management
  - ☑ Handles
  - ☑ Buffer (wave data)
  - ☑ Voice (management)
- ☑ Timer Event System
  - ☑ Sound Duration
- ☑ Stitching
  - ☑ Seamless transitions between voices using Voice Callbacks
- ☑ Memory Leaking Verification
  - ☑ No Resources Leaking

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 15:00 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
- Submit your code to perforce to the appropriate MS1 directory
  - Verify it

Pdf form (this document)

- *Submit this PDF to perforce*
  - *Fill in form*
    - *Name, changlelist, etc…*
    - *YouTube Link*
  - *Submit back to perforce*
    - *Check it out and 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

Allowed to Add files to this project

- This project compile as is… with whatever is submitted

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 ***Milestone*** 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

- Demo the Audio Engine through 2 demos
  - Basics
  - Stitching
- Summary of features
  - Audio demo inside a Game project
    - Space ship spinning while all the demos are working
  - At least 2 threads
    - Game Thread
    - Audio Thread
  - Additional threads
    - Use to make the testing easy….
    - Let the threads work for you
  - All communication is multithreaded
    - Game call sound system on Game Thread communicates to Audio or Audio/File thread combo
  - Wave data format
    - All sound files are Wav files, 48 KHz, 32 bit PCM
    - Any wave data can be in Mono or Stereo form
  - Audio system supports
    - 2 speaker stereo configuration ONLY
    - This is needed for demos
  - Every sound on the game thread is protected with ***HANDLES***
    - Protects resources on the game side

Summary of Milestone

- Playlists
  - We introduce a Playlist
    - Game side calls a Playlist
      - it creates an instance of sound (a voice and associates the wave data)
      - from the playlist… you can start, stop, change attributes of the sound
    - Simple or complex
      - A playlist can be simple as a single wave playing or more complicated that associates several wave files playing with one controlling structure
        - This allows several sounds to be played from a single instance
  - There are two types of playlists in the audio engine
    - Simple Playlist
      - Associates a wave file and a sound call name (ID)
      - Creates one Voice and associates Wave data to the voice with internal XAudio2 callbacks (if needed)

        - Complex Playlist (use for Seinfeld)
            - Special playlist to show off Voice Callbacks and stitching wave data together on a voice
                - This playlist that a series of wave files that are associated together through Voice callback on a SINGLE voice.
            - Wave Files:
                - Intro
                - Wave_A
                - Wave_A_to_B
                - Wave_B
                - Wave_B_to_A
                - Wave_C
                - Wave_C_to_A
                - Wave_END
- Basics Information
    - Every sound call that is active
        - You can query the current status of
            - Volume
            - Pan
            - Pitch
            - Time - How long has it been playing
    - You can set the following attributes
        - Volume
        - Pan
        - Pitch
    - A sound is a unique instance of a playlist
        - So you can have several of the same playlist playing at once
        - You can control each instance individually
- Miscellaneous
    - Make sure you have a Timer system working
        - Its need to do the demos on the Game Side
        - For example:
            - Start sound call A at 1500 ms
            - Start sound call B at 2000 ms
            - At 5000 ms, stop Sound call A
    - Make sure you can read a keyboard to trigger a demo
        - For example:
            - start Demo 1
                - hit the "1" key
            - start Demo 2
                - hit the "2" key
- NO individual or special demos required
    - Only these required demos
    - Trying to make it easy for you.

General

0. Do all your working in MS1

a. Do all your development in MS1 for this milestone
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 Threads, XAudio2, Voices and callbacks

a. Look up Voice callbacks
    i. Explore the demos
b. Experiment with Data Buffers
    i. Wave data with formatting structures
c. Passing data to threads to start and control audio
    i. You need to communicate and have Audio thread control functionality
d. You need a Handle System
    i. Make sure your resources are protected by handles
e. Contention between threads…
    i. Should be protected with mutexes

Seinfeld Setup:

- Given simple mono wave samples
  - Sampled at 48Khz, 32-bit
- Create 1 Starting Seinfeld Voice Intro (aka. Script or function call that starts one voice)
  - This triggers – the first wave.
  - Through call backs one at a time….
    - Stitch the 8 sound waves starting from that one voice
- We are demoing XAudio Callbacks on a voice (stitching together audio buffers)
  - ***Option 1*** – starts another voice triggered in the callback
    - Easiest option – for this demo
      - Callback starts a new voice in the callback
      - Allows the next voice to be stitched on the master voice list
    - Difficult to adjust a single volume, pan, pitch once you start this playlist
      - Since there are several voices effectively being stitched together.
  - ***Option 2*** - append audio on single voice
    - This is a harder option, but the benefit is a single voice controls the volume, pitch and pan. Easy to adjust
      - Appends an audio buffer data on the one and only voice
        - The callback appends different data on the SINGLE voice
      - Difficult part:
        - Need persistent state or linking waves for wave data appending
- Wave flow
  - NOTE: You need to do the individual stitching in the callback.

- Triggering one voice at a time in callback (option1)
- Or Adding one buffer at a time in callback (option 2)

- Ordering of waves
  1. Intro
  2. A
  3. AtoB
  4. B
  5. BtoC
  6. C
  7. CtoA
  8. End

2. **Demo cannot LEAK resources or memory**
   a. Make sure you shut down all resources and threads correctly
   b. Add a special Key to kill the program before exit
      i. Key Q – QUIT is a good choice
      ii. Then escape key to close the window
   c. Need to see that there is no Memory Leaks on exit

3. **For Demo timing…**
   a. Use ***std::this_thread::sleep_for()*** to control the time…
   b. If you need to sequence actions in the demo

4. **Do not have any Threads spin directly**
   a. For example Audio Thread…
      i. As long as there is input… grab the input
      ii. Then have the Audio Thread sleep for 1 ms before grabbing input again

5. **Sound not specified**
   a. Default Attributes:
      i. Vol: 70%
      ii. Pan: Center
      iii. Pitch: Original

6. ***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

Demo

Demo 1: Basics

Setup:

- Given 5 simple mono wave samples
  - Sampled at 48Khz, 32-bit
- Create 5 separate simple playlists (scripts) – one sound wave, one sound ID
  - 101 - Fiddle
  - 102 - Bassoon
  - 103 - Oboe2_mono
  - 104 - SongA
  - 105 - SongB

Demo:

Start Demo 1 –hit the "1" 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 1
- The demo should play from there.
  - No user intervention needed – just need the timer triggers working.

Part A: Load

- Setup your playlists
  - Load all the mono wave data for 101-105 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/threads do the work

Part B: preset pan test

- Timer: 0 seconds
  - Play 101 with pan in center
- Timer: 3 seconds
  - Play 101 with pan 100% left
- Timer: 6 seconds
  - Play 101 with pan 100% right

Part C: runtime panning adjustment with write only

- Timer: 10 seconds
  - Play 102 with pan 100% left and move it to 100% right
    - By setting the attribute directly
      - Every 1ms change the panning…
  - Smoothly for 2 seconds

- Timer: 15 seconds
  - Play 102 with pan 100% right and move it to 100% left
    - By setting the attribute directly
      - Every 1ms change the panning
  - Smoothly for 2 seconds

Part D: runtime volume adjustment with a read modify write

- Timer: 20 seconds
  - Play 103 with volume at 0% and ramp up the volume smoothly to 100%
  - Smoothly across 2 seconds
    - Do this by 1st – reading the current volume
    - Then add a delta to the volume and set the attribute directly
      - Every 1ms change the volume
- Timer: 25 seconds
  - Play 103 with volume at 100% and ramp down the volume smoothly to 0%
  - Smoothly across 2 seconds
    - Do this by 1st – reading the current volume
    - Then add a delta to the volume and set the attribute directly
      - Every 1ms change the volume

Part E: Stereo effect from mono

- Timer: 30 seconds
  - Play 104 with pan 100% left
  - Play 105 with pan 100% right
- Timer: 35 seconds
  - Print in the Debugger's Output screen the time 104 has been playing in seconds
    - (since it started playing)
    - Need to use the timer…no hard coding numbers
- Timer: 38 seconds
  - Print in the Debugger's Output the time 104 has been playing in seconds
    - (since it started playing)
    - Need to use the timer…no hard coding numbers
- Timer: 60 seconds
  - Print in the Debugger's Output screen the time 104 has been playing in seconds
    - (since it started playing)
    - Need to use the timer…no hard coding numbers
  - Stop 104
- Timer: 72 seconds
  - Print in the Debugger's Output screen the time 105 has been playing in seconds
    - (since it started playing)
    - Need to use the timer…no hard coding numbers
  - Stop 105

Part F: Instancing several sounds

- Timer: 80 seconds
  - Snd A = Start 102
  - set vol to 40%
- Timer: 80.5 seconds
  - SndB = Start 102
  - set vol to 40%
- Timer: 81 seconds
  - SndC = Start 102
  - set vol to 40%
- Timer: 81.5 seconds
  - SndD = Start 102
  - set vol to 40%
- Timer: 81.5 seconds
  - Stop SndA
  - Stop SndB
  - Stop SndC
- Timer: 82 seconds and beyond
  - Let SndD – play and die without intervention

Demo 2: Voice Stitching – using XAudio2 Callbacks

Setup:

- See above description on Seinfeld:
  - Given 8 simple mono wave samples
    - Sampled at 48Khz, 32-bit
- Create 8 separate simple waves
  - Intro_mono
  - A_mono
  - AtoB_mono
  - B_mono
  - BtoC_mono
  - C_mono
  - CtoA_mono
  - End_mono
- On controlling playlist
  - SndID 201 – is the controller for this playlist

Demo:

Start Demo 2 –hit the "2" 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 2
- The demo should play from there.
  - No user intervention needed – just need the timer triggers working.
- Print the name of each wave as it stitched in the XAudio2 Callback
  - Since only one wave is stitched at a time
  - The names should be printed at intervals proportional to the individual wave playback
    - They shouldn't be burst on the screen
    - Instead one at a time… with delays between them

Part A: Load

- Setup your playlists
  - Load all the mono wave data needed for 201 initiate on the game side
  - It's OK to have the playlist table on the Audio Thread side
    - Create the callbacks for the voices (option 1 or 2)
  - Audio side 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/threads do the work

Part B: Start the demo

- On the ***Game THREAD***
  - At 0 seconds
    - Play 201, pan center, volume 80%
      - → Print the name of each wave as it stitched in the XAudio2 Callback
        - Since only one wave is stitched at a time
        - The names should be printed at intervals proportional to the individual wave playback
          - They shouldn't be burst on the screen
          - Instead one at a time… with delays between them
  - At 10 seconds
    - Pan Right 201, volume 80%
  - At 20 seconds
    - Pan Left 201, volume 80%
  - At 30 seconds
    - Pan Center 201, volume 80%
- Repeat the panning pattern
  - Center, Right, Left – 10 seconds apart
    - Do this until the audio ends

Exit the Game cleanly

- Send the Kill command
  - Key Q
- Then close the window by hitting escape
- MUST show the game shutting down cleanly with no memory leaks.

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 when shutting down?
- Submitted the YouTube link to perforce?
- Is it recorded clearly, loudly, and in stereo?

Hints

Most assignments will have hints in a section like this.

- Dig into the material read the online blogs…
  - Lots and lots of information
- Use the Piazza FORUMs
  - Read, explore, ask questions