use Adobe Reader to complete

(Type in fields)

Sprint4 – Callbacks

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:	05/21/2022

Submission Details

Final ***Changelist*** number:	147992
Verified build:	[✓ Yes] No
Required Configurations:	x86 - DEBUG (only)
YouTubeLink:	https://youtu.be/Es-wZhga67o

Discussion (What did you learn):

This sprint was more difficult than I expected. While callbacks themselves weren't too bad, I had to do a lot of reworking to my code. I used the stitched callback, where I only used one voice. This made it easier to do the panning part of the demo.

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 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*

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
    - Voice Callbacks – creating custom callbacks
    - Trigger when voices change state
        - End Buffer
        - End Stream
        - End looping
        - etc
    - Stitch sound calls together
        - Dynamically extend and append audio streams together

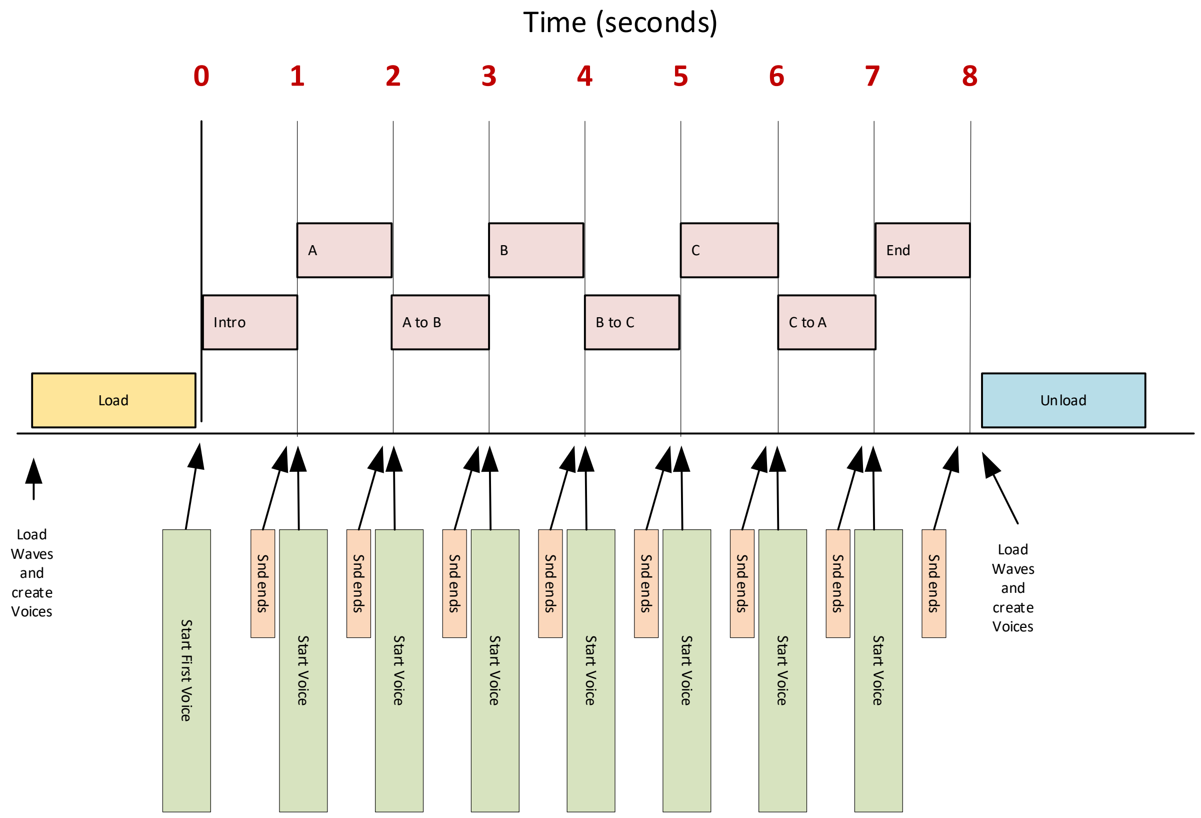
Assignments

0. ***Setup directory Sprint 4***
    a. Copy your contents of your Sprint3 directory into Sprint4
        i. Do all your development in this directory for this Sprint
    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 Voices and callbacks***
    a. Look up Voice callbacks
    b. Explore the demos
    c. Experiment with Data Buffers
    d. Experiment with Voices calls starting

Setup:

- Given 8 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(code)
            - 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



Demo:

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

- This is triggered in the update() method of the game thread
  - 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 all the mono wave data needed for demo initiated on the game side
    - Put together a load
    - This is important… needs to be preloaded in memory
  - It's OK to have the playlist table on the Audio Thread side
    - Create the callbacks for the voices (option 1 or 2)
    - Playlist – a table to associate the sound wave to a name/buffer
  - Audio side cannot load the wave data, that has to be initiated on the game side
    - Load all waves in LoadContent()
  - For Panning demo… use std::this_thread::sleep_for()
    - Easier to demo the panning with these sleeps…

Start the demo

- Start Demo – hit the <SPACE> key to trigger it
  - One time only… it starts the intro voice
- On the ***Game THREAD***
  - At 0 seconds
    - Pan center, volume 50%
    - → 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, volume 50%
  - At 20 seconds
    - Pan Left, volume 50%
  - At 30 seconds
    - Pan Center, volume 50%
- Repeat the panning
  - Center, Right, Left – 10 seconds apart
    - Do this until the audio ends

2. 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. Expectation of the demo is around 5 minutes long for this PA

3. Discuss on YouTube Demo

a. Build and explain the code
b. Run the demo
c. Make sure you record in stereo

4. 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
- YouTube recording

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