

## Milestone 2

### 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:

Date:

### Submission Details

Final **Changelist** number:

Verified build:                      Yes                      No

Required Configurations:

YouTube Link:

### Self Grading Section:

Required Features:

Basics

XAudio2 Voices

Attributes (volume, pan, pitch, etc.)

Loading Waves (initiated from game side)

Instancing playing the multiple instances of the same SndCall

Stitching

Seamless transitions between voices using Voice Callbacks

Using Voice Buffers or Stitch Voices together

Priorities

Creating a priority table (6 entries)

Preemptively cancelling or calling SndCalls based on their priorities

Using "time playing" for cases where there are multiple calls with same priority

Game User Callbacks

On Game Side – user supplied callback for StreamEnd()

Callback for the game to trigger an event on playing end

#### Asynchronous File

File load callback when the file is loaded and ready on the Audio Thread

#### Audio Management

Handles

Buffer (wave data)

Voice (management)

Other systems for management

#### Memory Leaking

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
  - 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 MS2
    - Play the demo and add your commentary in real-time
  - Watch your video
    - Verify that video clear and can you hear the commentary audio in stereo?
- Note: Weekly Sprints cannot be longer that **10:00 mins**
  - If you go over... do it again
  - **NEED TO see compiling and discussion of code with demo**
- 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 MS2 directory
  - Verify it

### Pdf form (this document)

- *Submit this PDF to performe*
  - *Fill in form*
    - *Name, changelist, etc...*
    - *YouTube Link*
  - *Submit back to performe*
    - *Check it out and Submit it back to performe 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 Performe**

- Submit your work as you go to performe several times (at least 5)
  - As soon as you get something working, submit to performe
  - 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 4 ...
- 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 will work “as-is” do not add files...

## Due Dates

- See Piazza for due date
- Submit program performe in your student directory assignment supplied.
- Fill out your this **MS2 pdf** and add **YouTube link**
  - **ONLY** use Adobe Reader to fill out form, all others will be rejected.
  - Fill out the form and discussion for full credit.
- **Need to see discussion of code, compiling and demo in VIDEO**
  - Demo and explain each prototype feature:
    - YouTube Video Demo (public link must work, no sign in required)
  - Demo of the milestone
    - If I don't hear and see the explanation, I assume you didn't do the feature

## Summary of Milestone

- Basics sound calls
  - Vol, Panning, Pitch, Start, Stop, sound durations, etc
- Stitching calls together by Voice Callbacks
- Priority System
- User Callbacks
- Async File loading

## General

- ➔ **Do all your working in MS2 directory**
  - ➔ Copy PA5 or PA6 to start the MS2 directory
  - ➔ **You need to use File\_Slow methods for milestone2**
    - i. **No other file system allowed**
  - ➔ Make sure you submit this project many times to performe as you develop
    - i. You need to submit the project and the video for this milestone
- ➔ **Demo cannot LEAK resources or memory**
  - ➔ Make sure you shut down all resources and threads correctly

- ➔ 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
- ➔ Need to see that there is no Memory Leaks on exit
- ➔ **For Demo timing...**
  - ➔ Use `std::this_thread::sleep_for()` to control the time...
  - ➔ If you need to sequence actions in the demo
- ➔ **Do not have any Threads spin directly**
  - ➔ 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
- ➔ **Sound not specified**
  - ➔ Default Attributes:
    - i. Vol: 70%
    - ii. Pan: Center
    - iii. Pitch: Original
    - iv. Priority: 200
- ➔ **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 4
    - 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 1: BASICS

### Demo 1: Basics

- ➔ You need to use File\_Slow methods for milestone2
- ➔ All Sound wave assets need to be loaded/initiated from Game Side

#### 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
  - 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 1<sup>st</sup> – 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 1<sup>st</sup> – 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

#### Demo 2: Voice Stitching – using XAudio2 Callbacks

- ➔ You need to use File\_Slow methods for milestone2
- ➔ All Sound wave assets need to be loaded/initiated from Game Side

#### Setup:

- See above description on Senfeld:
  - 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
  - Audio side cannot load the wave data, that has to be initiated on the game side
- Load all the callback 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

## Demo 3: Priority

### Demo 3: Priority

- ➔ You need to use `File_Slow` methods for milestone2
- ➔ All Sound wave assets need to be loaded/initiated from Game Side

#### 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
  - Handle, priority, time playing
  - For example: (3 handles – in the active table)
    - ----- Active Table -----
    - 0xAAAA0001: 10 1500 ms
    - 0xAAAA0004: 50 1500 ms
    - 0xAAAA0005: 75 200 ms
- Sound call for this demo is more of a placeholder
  - Keep the volume down to 10% for all of these call
  - Same priority kills snd call that has oldest equal priority
- 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**
    - **`File_Slow::Open()`**
    - **`File_Slow::Read()`**
    - **`File_Slow::Seek()`**
    - **`File_Slow::Tell()`**
    - **`File_Slow::Close()`**

#### Demo:

- Start Demo 3 –hit the “3” 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

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

- 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)
- 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

#### Demo 4: User Callbacks

### Demo 4: User Callbacks

- You need to use File\_Slow methods for milestone2
- All Sound wave assets need to be loaded/initiated from Game Side

#### Setup:

- Given 4 simple mono wave samples
  - Sampled at 48Khz, 32-bit
- Create 4 separate simple playlists (scripts) – 1 sound wave, one sound ID
  - 401 – Dial
  - 402 – MoonPatrol
  - 403 – Sequence
  - 404 - Donkey
- We are demoing Game user callbacks
  - Create 4 unique callbacks (need to be unique instance)
    - On **StreamEnd** of the sound call
  - Have each callback
    - Using active sound call table (you cannot hard code these values)
      - **Print the duration the sound played**
      - **Print the sound handle ID**
      - **Print the pan value**
      - **Print the name of the wave file**
    - Remember you have unique game user callbacks so you can hand type the wave name in each callback
    - But only the wave name!

#### Demo:

Start Demo 4 –hit the “4” 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 4

- 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 401-404 initiate on the game side
    - It's OK to have the playlist table on the Audio Thread side
  - Audio side cannot load the wave data, that has to be initiated on the game side
- Setup your playlists
  - Load all the mono wave data needed for demo initiated on the game side
    - Put together a load
      - Load all the wave data needed for 401-404 initiate on the game side
      - Audio side cannot load the wave data, that has to be initiated on the game side
    - This is important... needs to be preloaded in memory
- For timing...
  - USE `std::this_thread::sleep_for()` to control the time...
- Initiate all 4 game user callbacks
  - Done on the game side
  - Call them A, B, C, and D
    - Corresponding to 401, 402, 403, 404 sound calls
- Load all the timer events for this demo at once – let the timer do the work

#### Start the demo

- Timer: 0 seconds
  - SndA = Play 401, pan: 100% left, GameCallback\_A
  - SndB = Play 402, pan: 100% right, GameCallback\_B
  - SndC = Play 403, pan: 100% left, GameCallback\_C
- Timer: 3.5 seconds
  - SndD = Play 404, pan: 100% left, GameCallback\_D

Nothing else – the callbacks to the work

➔ Make sure you are printing the correct material in the user callback on "StreamEnd" of the voice

## Demo 5: Async Loading

### Demo 5: Async Loading

- ➔ You need to use `File_Slow` methods for milestone2
- ➔ All Sound wave assets need to be loaded/initiated from Game Side

#### 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 – Alarm
  - 503 – Beethoven
- We are demoing Asynchronous loading and game user file load callback
  - Create 1 unique game user file load callback
    - Printing 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 `FILE_SLOW` class to simulate latency**
    - **`File_Slow::Open()`**
    - **`File_Slow::Read()`**
    - **`File_Slow::Seek()`**
    - **`File_Slow::Tell()`**
    - **`File_Slow::Close()`**

**Demo:**

Start Demo 5 –hit the “5” key to trigger it

- This is triggered in the update() method of the game
  - Read the keyboard input
  - Then go with your Demo 5
- 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, 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**

- Timer: 0 seconds
  - SndA = Play 501, vol: 30%, pan: 100% left, Priority default (optional)
  - 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)
  - SndB = Play 502, vol: 30%, pan: 100% left, Priority default (optional)
- Timer: 5 seconds
  - SndB = Play 502, vol: 30%, pan: 100% left, Priority default (optional)
- Timer: 10 seconds
  - SndB = Play 502, vol: 30%, pan: 100% left, Priority default (optional)
- Timer: 15 seconds
  - SndB = Play 502, vol: 30%, pan: 100% left, Priority default (optional)
- Timer: 20 seconds
  - SndB = Play 502, vol: 30%, pan: 100% left, Priority default (optional)



- Timer: 25 seconds
  - SndB = Play 502, vol: 30%, pan: 100% left, Priority default (optional)
- 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
  - Stop SndA (501)
- Timer: 60 seconds
  - Stop SndC

Nothing else – the callbacks to the work

### ***Exit the Game cleanly***

- Send the Kill command
  - Key Q
- Then close the window by hitting escape

### Validation

*Simple checklist to make sure that everything is submitted correctly*

- Submitted project to perform correctly
  - Is the project compiling and running without any errors or warnings?
  - Is the submission report filled in and submitted to perform?
  - Follow the verification process for perform
    - 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 perform?
- 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