Game Audio Programming

EECS 4462 - Digital Audio

November 12, 2020

Most material in this slide set is from
Game Audio Programming 2: Principles and Practices,
by Guy Somberg, CRC Press

YORK UNIVERSITÉ UNIVERSITY

Audio for movies vs games

- Similarities
  - 3 main audio components: Music, Dialogue, Sound effects
  - Need quality audio for all three
    - Music: Appropriate genre, well recorded, different music for different parts of the movie/game
    - Dialogue: Clearly recorded with no background noise
    - Sound FX: Realistic sounds / foley
- https://www.youtube.com/watch?v=U_tqB4IZvMk

Audio Personnel

- Many different roles for audio people (depending on the size of the project)
  - Sound designers will decide create original sounds
  - Composers will write original music and music professionals will play it
  - Recording engineers will record all audio components
  - Audio post production personnel will put everything together in the end to create an immersive experience
  - A game audio programmer is also doing audio post production, but…

Audio for movies vs games

- Differences
  - Audio post production takes place after the visual aspect of a movie has been fixed
  - A movie is a passive experience
  - Games are interactive
  - Sounds must be mixed together on the fly depending on what the player and all the characters in the game are doing
  - Limited resources in terms of memory and CPU time will be available for audio

Game production lifecycle

- Game audio should be a part of game production from the beginning
- The three phases of game production and their milestones:

1. Preproduction ➔ First Playable (Vertical Slice)
2. Production ➔ Alpha (Content Complete)
3. Postproduction ➔ Beta (Content Finalized)

Audio middleware

- Keeps track of all the audio content, as well as the audio infrastructure, e.g.
  - Audio assets contain raw audio as well as information about volume, positioning, pitch etc.
  - Audio events are triggered by the game engine and contain information as to how an audio asset should sound, e.g. how it will attenuate over distance
  - Audio triggers are in-game entities that trigger audio events

Preproduction

- Decide on the audio modules that will be part of the game
  - Player
  - Cast
  - Levels
  - Environment
  - Music
- Brainstorm on ideas for all the above

Audio prototypes

- Small test cases to help us define triggers, events, and their interaction
- Audio assets can be simple but as realistic as possible at this point
- Examples
    - Surface types
    - Reverb areas
    - Equipment / vehicles
    - Environments
    - Other characters

Audio design layers

- Feedback
  - Indicates that something has happened, e.g. a beep in PONG
- Immersion
  - More realistic sounds, e.g. a racket flick, that may change based on speed or direction
- Emotion / Experience
  - The audio changes depending on whether the player is winning or losing, or gets louder as time goes by
- In preproduction, we focus on feedback and try to incorporate as much immersion as possible

Preproduction milestone

- **First Playable**
  - Limited / condensed version of the final product
- Showcases only a few basic options
- Main goal to set up and test the audio infrastructure of events and triggers

Production

- Fill up every element of the game
- Can use placeholders when audio is not available
  - E.g. say the name of the event, or the state of different characters, or of the sound effect
  - Many audio assets are created in post production where things like character clothes or shoes are finalized
- Debug placeholders, e.g. a beep, can be used when game states or parameters are not specified or missing
- Build towards immersion as much as possible

Production milestone

- **Alpha**
- The game is content complete, i.e. can be played from start to finish without crashes
- There is audio for every event even if it is a placeholder
- All audio assets conform to the EBU R128 Loudness Recommendation Standard
  - Measures loudness in loudness units (LU) across an audio asset, not only at the peak level
  - Much closer to how loudness is perceived
  - https://www.youtube.com/watch?v=iuEtQqC-Sqo

Postproduction

- Finalize all audio assets
- Test!
  - Typically done in pairs: One tester plays, the other mixes, i.e. adjusts volumes, effects etc.
  - Bug fixing
- Milestone: **Beta**

Dealing with multiple characters

- In many modern games, there is a variety of characters on the screen at at time that could generate audio
- Mixing audio from many sources can make the final mix sound muddy, and uses many resources
- The HW platform may also have a playback limit
- Solution: Virtual Channels
  - Some sources of audio (sometimes also called channels) are declared virtual, i.e. do not contribute audio
  - It is up to the audio programmer to decide how this will be implemented

Choosing virtual channels

- Assign a priority to each sound, and play only those sounds of highest priority
- The listener's distance to the sound must be factored in
  - Further sounds must have lower priority
- In modern HW, the global playback limit is not an issue, but audio programmers often impose local limits to avoid making the mix too muddy
  - For example, limit gunshot sounds to at most 20

The rule of two and a half

- Used in movie audio post production
- When one actor is walking, it is important that the sound of their footsteps match the visuals
- Similarly, when two actors are walking
- When we pass the threshold of 2.5 actors, it is no longer necessary to synchronize footsteps to actors
    - Audio of multiple people walking is sufficient

Mixing

- The act of bringing all audio assets (music, voice, SFX) together in a way that is enjoyable and supports the gameplay
- Offline mixing: Mixing happens in a separate program (usually a DAW). The result is fixed and is reproduced as is
- Games require **real time mixing**
  - Volumes and frequencies of audio assets are changed as the game is running

Real time mixing (passive)

- The behaviour of audio is authored in a static way, so that the audio signal changes dynamically by being routed to a DSP effect
- Example: **Ducking** music when dialogue is heard
- This can be done programmatically (active mixing) but is often done by side-chaining the dialogue audio bus on to a dynamic range compressor on the music bus

Realtime mixing (active)

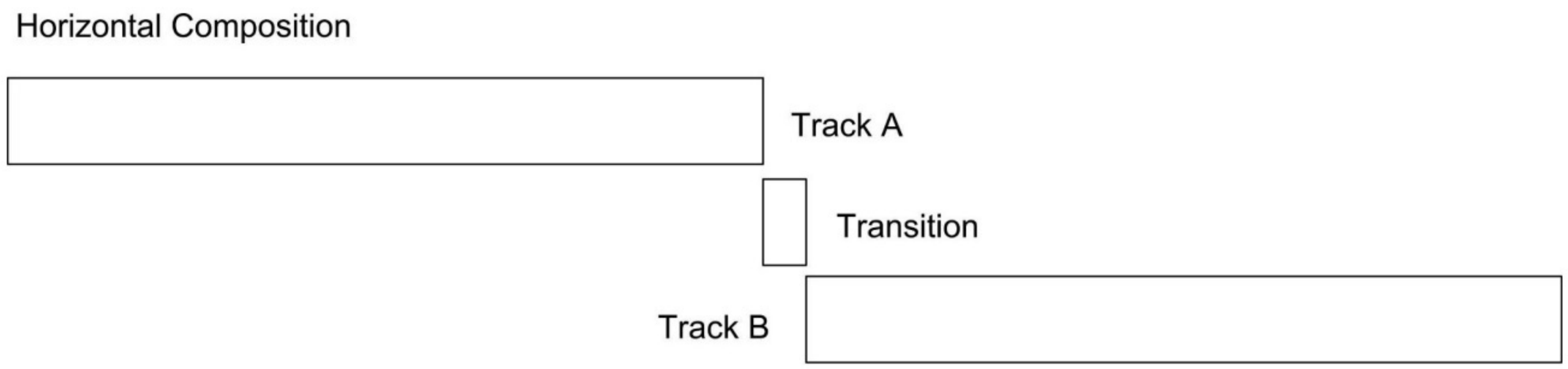
- Events in the gameplay manipulate the mix
- Audio assets are being changed on the fly
- Example: An explosion happens very close to the player. Instead of playing an explosion sound very loud, the game ducks the SFX bus and plays a ringing sound instead

Music transitions

- Music background for games is often pre-composed and recorded
- Looping the same audio file can quickly become boring
- Different levels in a game will typically have different music
- Dramatic events also require music transitions
- Two ways to accomplish this
  - Horizontal Composition
  - Vertical Composition

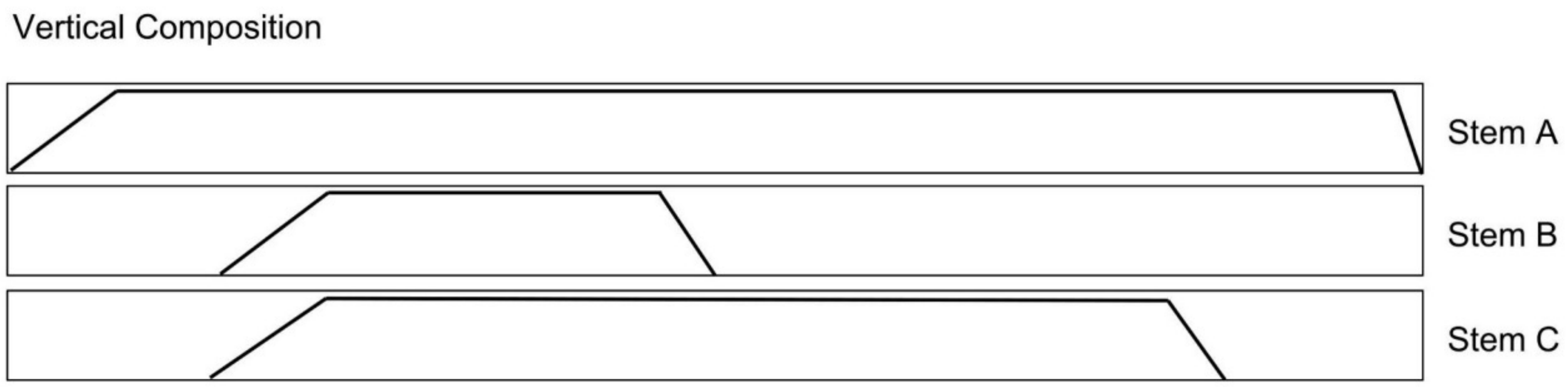
Horizontal Composition

- Music switches from one pre-recorded track to another
- Transition can be done through fade in/out, but more often through a **stinger**, a small piece of audio that corresponds to the event that required the transition, e.g. an explosion



Vertical Composition

- Different sets of instruments are recorded in separate tracks, called **stems**
- Based on game events, different stems are faded in or out, e.g. drums may come in for a fight scene, or piano for a more quiet part, while strings are playing throughout



MIDI in games

- To allow more flexibility in the tempo and texture of game music, modern audio middleware supports MIDI
- This allows for different instruments to be swapped in
- Each musical piece can be played in different keys
- Different scales can be used in different situations, e.g. major for action, minor for more somber parts
- Randomizing some notes in terms of pitch of velocity can allow for many variations
- CPU intensive as samples have to be created on the fly