Game Engine Development II

Week 12

Hooman Salamat

Audio Implementation

Objectives

- Implement different music themes in the background
- Implement sound effects that correspond to game events such as explosions
- Integrate sound effects in the 2D world to convey a feeling of spatial sound

Music Themes

- We want to play background music depending on the state we are currently in
- We have two themes
  - One for the menu
  - One for the game
- To do this, we'll define an `enum`:

```
namespace Music
{
    enum ID
    {
       MenuTheme,
       MissionTheme,
    };
}
```

Music Themes (cont'd.)

- Now we'll create a class that has an interface to play music:

```
class MusicPlayer : private sf::NonCopyable
{
public:
   MusicPlayer();
   void play(Music::ID theme);
   void stop();
   void setPaused(bool paused);
   void setVolume(float volume);
private:
   sf::Music mMusic;
   std::map<Music::ID, std::string> mFilenames;
   float mVolume;
};
```

Music Themes (cont'd.)

- SFML does NOT suppose the MP3 format (gasp)
  - We'll use OGG instead
  - You can use the free Audacity to convert formats
    - http://audacity.sourceforge.net
  - Now back to our class methods:

```
MusicPlayer::MusicPlayer(): mMusic(), mFilenames(), mVolume(100.f)
{
   mFilenames[Music::MenuTheme] = "Media/Music/MenuTheme.ogg";
   mFilenames[Music::MissionTheme] = "Media/Music/MissionTheme.ogg";
}
```

Loading and Playing

```
void MusicPlayer::play(Music::ID theme)
{
    std::string filename = mFilenames[theme];
    if (!mMusic.openFromFile(filename))
          throw std::runtime_error("Music " + filename + " could not be loaded.");
    mMusic.setVolume(mVolume);
    mMusic.setLoop(true);
    mMusic.play();
}

void MusicPlayer::stop()
{
    mMusic.stop();
}

void MusicPlayer::setPaused(bool paused)
{
    if (paused)
          mMusic.pause();
    else
          mMusic.play();
}
```

Sound Effects

- We also create an enum for sound effects and a typedef for the resource holder of `sf::SoundBuffer`:

```
namespace SoundEffect
{
    enum ID
    {
          AlliedGunfire,
          EnemyGunfire,
          Explosion1,
          Explosion2,
          LaunchMissile,
          CollectPickup,
          Button,
    };
}

typedef ResourceHolder<sf::SoundBuffer, SoundEffect::ID> SoundBufferHolder;
```

Sound Effects (cont'd.)

```
class SoundPlayer : private sf::NonCopyable
{
public:
   SoundPlayer();
   void play(SoundEffect::ID effect);
   void play(SoundEffect::ID effect, sf::Vector2f position);
   void removeStoppedSounds();
   void setListenerPosition(sf::Vector2f position);
   sf::Vector2f getListenerPosition() const;
private:
   SoundBufferHolder mSoundBuffers;
   std::list<sf::Sound> mSounds;
};
```

Sound Effects (cont'd.)

```
SoundPlayer::SoundPlayer(): mSoundBuffers(), mSounds()
{
   mSoundBuffers.load(SoundEffect::AlliedGunfire,
       "Media/Sound/AlliedGunfire.wav");
   mSoundBuffers.load(SoundEffect::EnemyGunfire,
       "Media/Sound/EnemyGunfire.wav");
   ...
}

void SoundPlayer::play(SoundEffect::ID effect)
{
   mSounds.push_back(sf::Sound(mSoundBuffers.get(effect)));
   mSounds.back().play();
}
```

Sound Effects (cont'd.)

```
void SoundPlayer::removeStoppedSounds()
{
   mSounds.remove_if([] (const sf::Sound& s)
   {
       return s.getStatus() == sf::Sound::Stopped;
   });
}
```

GUI Sounds

```
class State
{
public:
   struct Context
   {
        ...
        MusicPlayer* music;
        SoundPlayer* sounds;
   };
};

Button::Button(State::Context context)
: ...
, mSounds(*context.sounds)
{
   ...
}
```

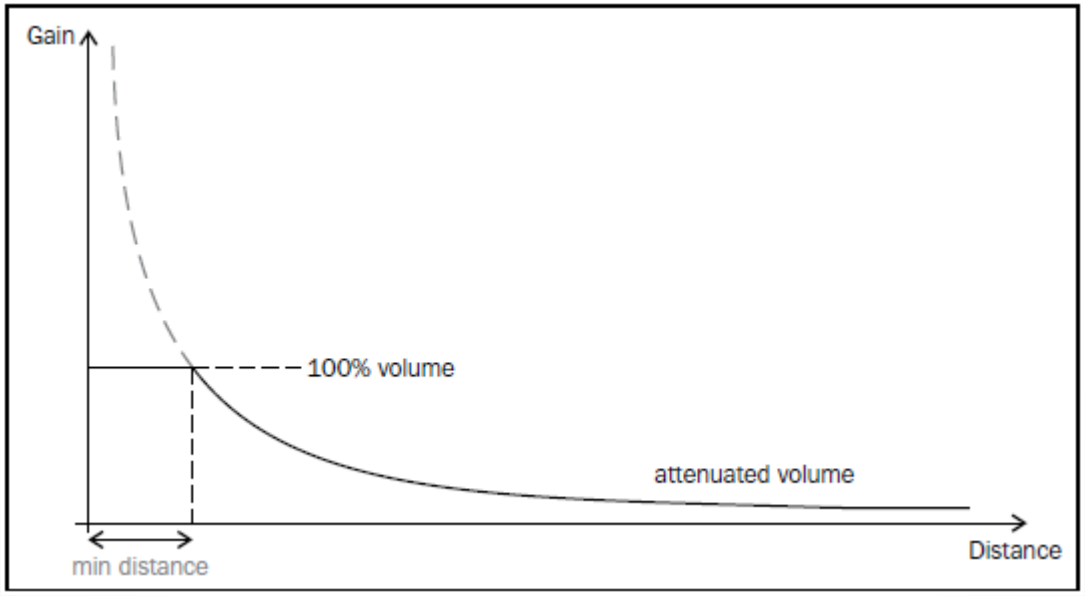
GUI Sounds (cont'd.)

```
void Button::activate()
{
   ...
   mSounds.play(SoundEffect::Button);
}

MenuState::MenuState(StateStack& stack, Context context)
: State(stack, context)
, ...
{
   auto playButton = std::make_shared<GUI::Button>(context);
   auto settingsButton = std::make_shared<GUI::Button>(context);
   auto exitButton = std::make_shared<GUI::Button>(context);
   ...
   context.music->play(Music::MenuTheme);
}
```

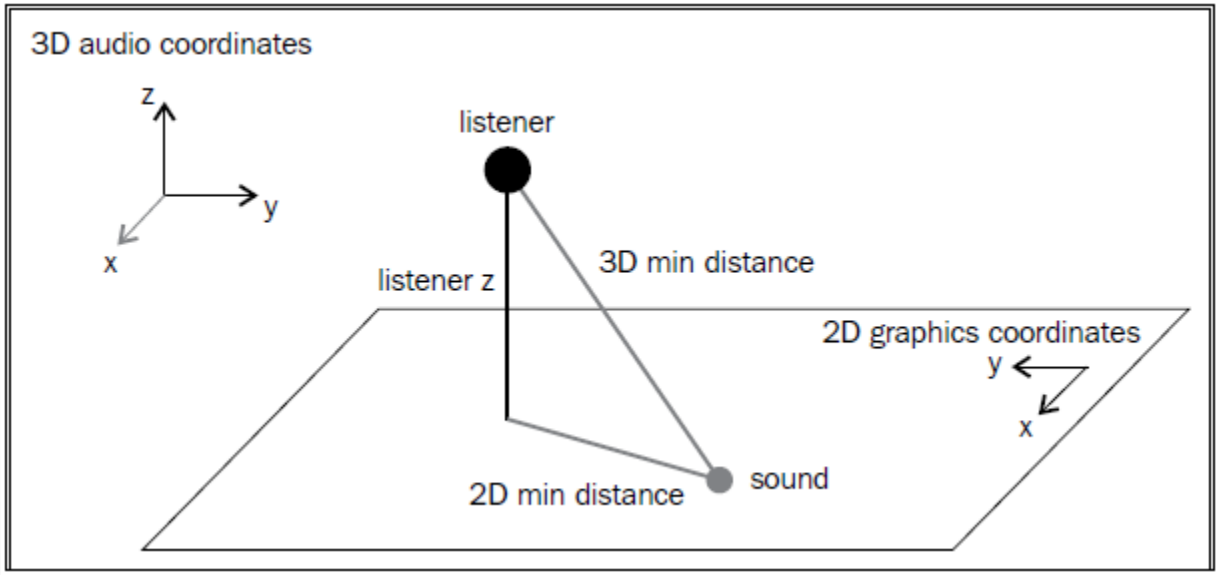
3D Sounds

- Close sounds are perceived louder than distant ones
  - Attenuation factor determines how fast a sound is attenuated depending on the distance



Positioning the Listener

- We place the listener in a plane different than the sound
  - The listener's Z coordinate has a greater value than zero



Playing Spatial Sounds

- In SoundPlayer.cpp, we create an anonymous namespace
  - For a few constants related to sound position

```
namespace
{
   const float ListenerZ = 300.f;
   const float Attenuation = 8.f;
   const float MinDistance2D = 200.f;
   const float MinDistance3D =
   std::sqrt(MinDistance2D*MinDistance2D + ListenerZ*ListenerZ);
}

void SoundPlayer::setListenerPosition(sf::Vector2f position)
{
   sf::Listener::setPosition(position.x, -position.y, ListenerZ);
}
```

Playing Spatial Sounds

```
void SoundPlayer::play(SoundEffect::ID effect, sf::Vector2f position)
{
    mSounds.push_back(sf::Sound());
    sf::Sound& sound = mSounds.back();
    sound.setBuffer(mSoundBuffers.get(effect));
    sound.setPosition(position.x, -position.y, 0.f);
     sound.setAttenuation(Attenuation);
    sound.setMinDistance(MinDistance3D);
    sound.play();
}

void SoundPlayer::play(SoundEffect::ID effect)
{
    play(effect, getListenerPosition());
}
```

Use Case

```
class SoundNode : public SceneNode
{
public:
    explicit SoundNode(SoundPlayer& player);
    void playSound(SoundEffect::ID sound,
    sf::Vector2f position);
    virtual unsigned int getCategory() const;
private:
    SoundPlayer& mSounds;
};

void Aircraft::playLocalSound(CommandQueue& commands, SoundEffect::ID effect)
{
    Command command;
    command.category = Category::SoundEffect;
    command.action = derivedAction<SoundNode>(
          std::bind(&SoundNode::playSound, _1, effect, getWorldPosition()));
    commands.push(command);
}
```

Use Case (cont'd.)

```
void Aircraft::checkProjectileLaunch(sf::Time dt, CommandQueue& commands)
{
    ...
    if (mIsFiring && mFireCountdown <= sf::Time::Zero)
    {
          playLocalSound(commands, isAllied() ?
          SoundEffect::AlliedGunfire : SoundEffect::EnemyGunfire);
          ...
    }
    if (mIsLaunchingMissile)
    {
          playLocalSound(commands, SoundEffect::LaunchMissile);
          ...
    }
}

void World::updateSounds()
{
    mSounds.setListenerPosition(
    mPlayerAircraft->getWorldPosition());
    mSounds.removeStoppedSounds();
}
```