

# DM 2231 GAMES DEVELOPMENT TECHNIQUES

## 2015/16 SEMESTER 1

---

Week 7 – Implementing Game Audio

# MODULE SCHEDULE

Week	Dates	Topic	Remarks	Public Holidays
1	20-Apr-2015 to 24-Apr-2015	Module Introduction / 3D Game Programming	Issue Assignment 1	
2	27-Apr-2015 to 1-May-2015	Game Application		1 May. Labour Day
3	4-May-2015 to 8-May-2015	User Input		
4	11-May-2015 to 15-May-2015	Camera and GUI #1		
5	18-May-2015 to 22-May-2015	Camera and GUI #2		
6	25-May-2015 to 29-May-2015	Basic Game Physics	Submit Assignment 1	
7	1-Jun-2015 to 5-Jun-2015	Implementing Game Audio (E-learning)		1 Jun. Vesak Day
8	8-Jun-2015 to 12-Jun-2015	Mid-Sem Break		
9	15-Jun-2015 to 19-Jun-2015	Mid-Sem Break		
10	22-Jun-2015 to 26-Jun-2015	2D Game Programming #1	Issue Assignment 2	
11	29-Jun-2015 to 3-Jul-2015	2D Game Programming #2		
12	6-Jul-2015 to 10-Jul-2015	2D Game Programming #3		
13	13-Jul-2015 to 17-Jul-2015	Game Data		17 Jul. Hari Raya Puasa
14	20-Jul-2015 to 24-Jul-2015	Design Pattern #1		
15	27-Jul-2015 to 31-Jul-2015	Design Pattern #2		
16	3-Aug-2015 to 7-Aug-2015	Basic Artificial Intelligence (E-learning)		7 Aug. SG50 Public Holiday
17	10-Aug-2015 to 14-Aug-2015	Good Programming Practices	Submit Assignment 2	10 Aug. National Day



# RECAP ON LAST WEEK'S LECTURE

- We have discussed about the main issues with Basic Game Physics
  - Introduction to Physics
  - Newtonian Physics
  - Vectors, Position, Velocity
  - Movements
  - Collision Detection Methods
    - Sphere - Sphere
    - Ray – Plane
    - Sphere - Plane

# TABLE OF CONTENT

- Implementing Game Audio
    - Why Audio ?
    - Generation
    - Types
    - Common Formats
    - Incident Sound
    - Music
    - Common Audio Libraries
    - irrKlang
      - How to use irrKlang
      - How to use irrKlang's 3D sound
-

# WHY AUDIO ?

- Immersive
  - Helps gamers to feel like they are “immersed” in the game
- Interesting
  - Enhance the gameplay
    - Just like sound effects in movies enhance the story
- Feedback
  - Gives an audio cue
    - Plane flying from left to right

# GENERATION

- Analogue
  - Not in this digital age
- Digital
  - encode in '0' or '1' after sampling
- Synthesized
  - mathematically generated by hardware
  - created by generating electrical signals of different frequencies

# TYPES

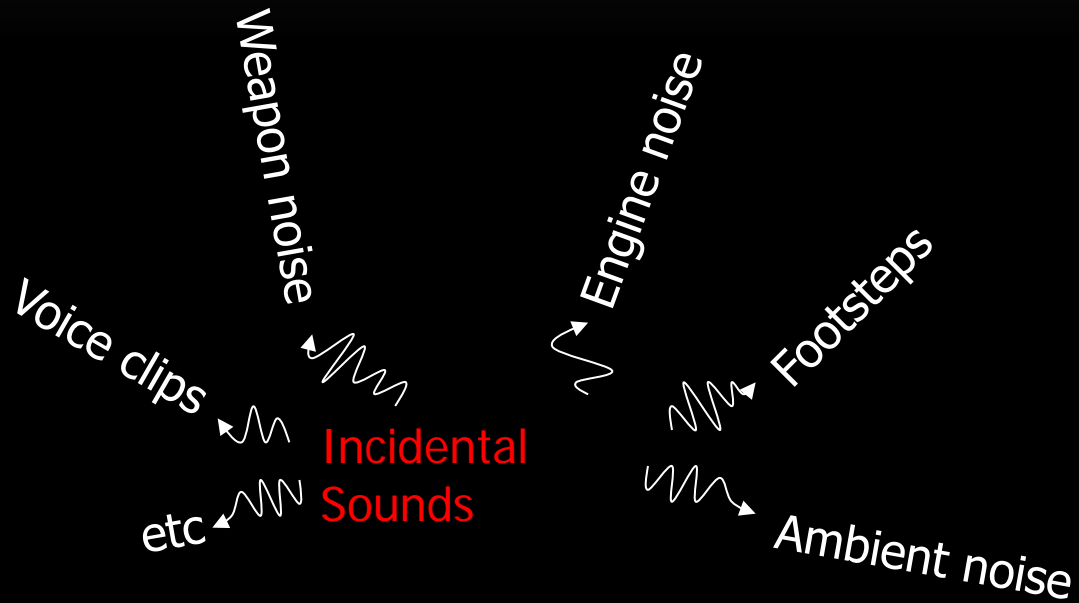
- Sound ( short play )
  - Sound Effects
    - Incident
      - bullet hit wall
    - Dialogue
- Music ( longer play )
  - Background
    - Intro
    - Gameplay music

# COMMON FORMATS

- Wav
  - Midi
  - Mp3
  - Ogg
- 
- Which format is not free for use?



# INCIDENT SOUND



# INCIDENT SOUND - CONTD

- Storage
- Digital format
  - wav
  - ogg
  - mp3
- Individual file
  - No effects
  - Effects added by audio engine

# INCIDENT SOUND - CONTD

- Simulate Environment Effects
  - Location ( left to right)
    - Source
    - Listener
  - Space
    - Big or small rooms eg: echoes
  - Travel
    - Blocked ( reduced volume, filter sound)
    - Moving (volume changes)
    - Distance (soft and loud)

# NATURE

- Looped Sound
  - Repeated again and again
- Background
  - Played in the background
  - Lower volume?
- Mood
  - Enhance the mood and realism of the gameplay?
  - Example: Walking through thick forest, sound of rustling leaves is played.
    - Does player feel that he is walking in a thick forest?

# IMPLEMENTATION

*Adaptive*



Game Music



*Linear*

# IMPLEMENTATION - CONTD

- Linear
  - Unresponsive to situation in game.
  - Splash screen, credits, menu screen, etc
- Adaptive
  - Changes with situation in game
- Examples :
  - Player enters china town, plays oriental music
  - Player enter battle, play exciting music
  - Player enters darkened corridor, play creepy music

# ADAPTIVE

- Environment
  - Location
  - Situation
- Changes
  - Tempo
  - Pitch
  - Tune
  - Instrument

# MIDI

- Musical Instruments Digital Interface
  - Agreed standard describing music in notes, instruments, effects
  - Does not store digitised sound information
  - MIDI sequence describes how a piece of music is to be played
  - At  $t=20\text{ms}$ , play middle C and hold that note for 3 seconds, then release it quickly. Use the violin sound to play it. Use a small amount of pitch bend.
  - At  $t=130\text{ms}$ , play D and hold for 2 seconds, then release it slowly. Use the cello sound. Don't use any pitch bend or after touch.



# SYNTHESIZERS

- Use Midi sequence to play music
  - Like pianist playing music score
- Implemented using hardware or software
  - Most modern computers has Midi synthesizer audio chips onboard.
  - Most operating systems has software support for Midi
  - Which to choose?

# COMMON AUDIO LIBRARIES

- FMod
  - <http://www.fmod.org>
- Open Audio Library (OpenAL)
  - <http://www.openal.org>
  - Cross-platform 3D audio API appropriate for use with gaming applications and many other types of audio applications.
  - Started by Creative Technology

# COMMON AUDIO LIBRARIES

- Miles Sound System
  - <http://www.radgametools.com/miles.htm>
  - Cross-platform (15 platforms)
  - Comes with editing tools
  - Not free!
- irrKlang
  - <http://www.ambiera.com/irrklang/>
  - High level 2D and 3D sound engine and audio library
    - Plays WAV, MP3, OGG, FLAC, MOD, XM, IT, S3M and more file formats
    - Usable in C++ and all .NET languages (C#, VisualBasic.NET, etc).
  - Cross-platform (Windows, Mac OS X, Linux)

# IRRKLANG

- Features
  - Powerful high level API for playing back sound in 3D and 2D applications like games, scientific visualizations and multimedia applications.
  - Free for non commercial use.
  - The advanced version of irrKlang is 'irrKlang pro'
    - Small license fee
    - Can be used in commercial products.

# IRRKLANG

Type	Usage Limitation	Details	Price
irrKlang - non commercial	Non commercial	<ul style="list-style-type: none"><li>• For non-commercial products</li></ul>	0€
irrKlang Pro hobbyist	(unlimited)	<ul style="list-style-type: none"><li>• For hobbyist developers (individuals, not companies)</li><li>• Unlimited amount of products which are sold for a low price (less than 20€).</li></ul>	65 €
irrKlang Pro indie	(unlimited)	<ul style="list-style-type: none"><li>• For independent developers</li><li>• Unlimited amount of commercial products which are sold for a price less than 27€.</li></ul>	290 €
irrKlang Pro full	(single product)	<ul style="list-style-type: none"><li>• For use in one commercial product.</li></ul>	490 €

# IRRKLANG

- Supported File Formats
  - RIFF WAVE (\*.wav)
  - Ogg Vorbis (\*.ogg)
  - MPEG-1 Audio Layer 3 (\*.mp3)
  - Free Lossless Audio Codec (\*.flac)
  - Amiga Modules (\*.mod)
  - Impulse Tracker (\*.it)
  - Screem Tracker 3 (\*.s3d)
  - Fast Tracker 2 (\*.xm)
- Extend the playable file formats using plugins by contacting the developer.

For mp3 playback in your application, you may need to purchase an mp3 license from Thompson Multimedia.

# IRRKLANG

- Supported platforms
  - Windows 98, ME, NT 4, 2000, XP, Vista, Windows 7
    - DirectSound 3
    - DirectSound 8
    - WinMM
  - Linux / \*nix
    - ALSA
  - Mac OS X (x86 as well as PPC)
    - CoreAudio

# IRRKLANG

- Other Features
  - Buffered and streamed audio playback, in 2D and 3D.
  - Exact seeking and position retrieval in streams and buffered sounds.
  - Simple API
  - High level resource management and auto detection: The engine can load, cache and/or stream sound data for you automatically, depending on what is best for performance and memory usage.
  - Sound effects
    - 3D sounds - Doppler effect
    - 2D effects - Echo, reverb, distortion, flanger and more.
  - Multi/Single-threaded modes
  - Low level audio output manipulation: Panning, volume and 3d position.



# HOW TO USE IRRKLANG

```
#include <iostream>
#include <irrKlang.h>
using namespace irrklang;

int main(int argc, const char** argv)
{
    // start the sound engine with default parameters
    ISoundEngine* engine = createIrrKlangDevice();

    if (!engine)
        return 0; // error starting up the engine

    // play some sound stream, looped
    engine->play2D("somefile.mp3", true);

    char i = 0;
    std::cin >> i; // wait for user to press some key

    engine->drop(); // delete engine
    return 0;
}
```

Include the header file for irrKlang.  
Use the irrklang namespace

Initialise the irrKlang sound engine.  
YOU ONLY NEED TO DO THIS  
ONCE!

Play a 2D sound using the engine.

Delete the irrKlang sound engine  
before quitting the program.  
YOU ONLY NEED TO DO THIS  
ONCE!

# HOW TO USE IRRKLANG'S 3D SOUND

```
#ifdef WIN32
#include <windows.h>
#include <conio.h>
inline void sleepSomeTime() { Sleep(100); }
#endif

#include <iostream>
#include <stdio.h>
#include <irrKlang.h>

using namespace irrklang;

#pragma comment(lib, "irrKlang.lib") // link with irrKlang.dll

int main(int argc, const char** argv)
{
    // start the sound engine with default parameters
    ISoundEngine* engine = createIrrKlangDevice();

    if (!engine)
        return 0; // error starting up the engine
}
```

Initialise the irrKlang sound engine.  
YOU ONLY NEED TO DO THIS ONCE!

# HOW TO USE IRRKLANG'S 3D SOUND

Create the handler to the 3D background sound.  
YOU ONLY NEED TO DO THIS  
ONCE!

```
// play some sound stream, looped, in 3D space
ISound* music = engine->play3D("../media/ophelia.mp3",
                                vec3df(0,0,0), true, false, true);

if (music)
    music->setMinDistance(5.0f);

printf("\nPlaying streamed sound in 3D.");
printf("\nPress ESC to quit, any other key to play sound at random
position.\n\n");
printf("+ = Listener position\n");
printf("o = Playing sound\n");

float posOnCircle = 0;
const float radius = 5;

while(true) // endless loop until user exits
{
    posOnCircle += 0.04f;
    vec3df pos3d(radius * cosf(posOnCircle),
                  0, radius * sinf(posOnCircle * 0.5f));
```

The minimum distance simply is the distance in which the sound gets played at maximum volume.

Calculate a position on a circular path

# HOW TO USE IRRKLANG'S 3D SOUND

```
engine->setListenerPosition(vec3df(0,0,0), vec3df(0,0,1));
```

Set the position of the listener

```
if (music)  
    music->setPosition(pos3d);
```

Set the position of the music

```
char stringForDisplay[] = "          +          ";
```

```
int charpos = (int)((pos3d.X + radius) / radius * 10.0f);  
if (charpos >= 0 && charpos < 20)  
    stringForDisplay[charpos] = 'o';
```

```
int playPos = music ? music->getPlayPosition() : 0;
```

```
printf("\rx:(%s)    3dpos: %.1f %.1f %.1f, playpos:%d:%.2d    ",  
        stringForDisplay, pos3d.X, pos3d.Y, pos3d.Z,  
        playPos/60000, (playPos%60000)/1000 );
```

```
sleepSomeTime();
```

# HOW TO USE IRRKLANG'S 3D SOUND

```
if (kbhit())
{
    int key = getch();

    if (key == 27)
        break; // user pressed ESCAPE key
    else
    {
        vec3df pos(fmodf((float)rand(),radius*2)-radius, 0, 0);

        const char* filename;

        if (rand()%2)
            filename = "../media/bell.wav";
        else
            filename = "../media/explosion.wav";

        engine->play3D(filename, pos);

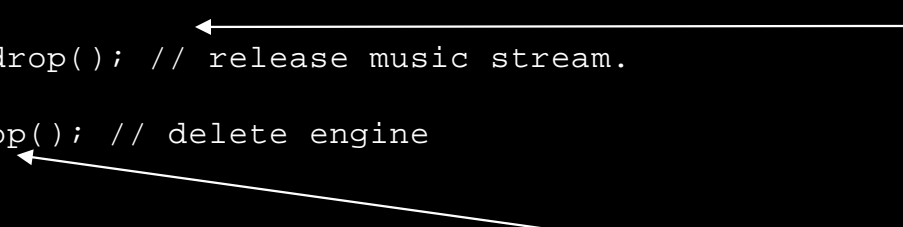
        printf("\nplaying %s at %.1f %.1f %.1f\n", filename, pos.X, pos.Y,
pos.Z);
    }
}
```

Randomly choose 1 sound to play

Play the sound!

# HOW TO USE IRRKLANG'S 3D SOUND

```
    }  
  }  
}  
  
// don't forget to release the resources as explained above.  
  
if (music)  
    music->drop(); // release music stream.  
  
engine->drop(); // delete engine  
return 0;  
}
```

A diagram consisting of two white arrows. The first arrow originates from the right side of the first blue text box and points to the line 'music->drop(); // release music stream.' in the code. The second arrow originates from the right side of the second blue text box and points to the line 'engine->drop(); // delete engine' in the code.

Delete the irrKlang music handler  
before quitting the program.  
YOU ONLY NEED TO DO THIS  
ONCE!

Delete the irrKlang sound engine  
before quitting the program.  
YOU ONLY NEED TO DO THIS  
ONCE!

# SUMMARY

- We have discussed about the main issues with Implementing Game Audio
  - Why Audio ?
  - Generation
  - Types
  - Common Formats
  - Incident Sound
  - Music
  - Common Audio Libraries
  - irrKlang
    - How to use irrKlang
    - How to use irrKlang's 3D sound