List of Symbols, Figures and Tables

1. Figure of the Madal
2. Figure showing the interaction between objects
3. Figure showing Abstraction Layers of Several SDL Platforms
4. OOP: Object Oriented Programming
5. SDL: Simple DirectMedia Layer

Main Report

1.0 Introduction

Our project was entitled as “MADAL in C++” which was basically a digital version of Madal. As we all know Madal is the most popular and widely used hand drum in Nepal. This typical Nepalese percussion instrument is the backbone of most of the Nepali folk music. So, we made a peculiar approach to bring this traditional musical instrument in the digital world. We have used the advantages of object oriented programming in our project with some taste of Simple DirectMedia Layer (SDL). Madal was basically a simple program which can produce all the beats of a real life Madal in response to specified keystrokes or mouse clicks input by the user. The program had easy to use and simple to navigate user interface which makes the experience more enjoyable and productive to some extent. Our entire project was based on the topics like Madal, SDL and Object Oriented Programming which are described in later part of the report.

1.1 Madal

The Madal used mainly for rhythm-keeping in Nepalese folk music is the most popular and widely used hand drum in Nepal. The Madal consists of a cylindrical body with a slight bulge at its center and heads at both ends, one head larger than the other. It is usually played horizontally in a seated position with both heads played simultaneously. This typical Nepalese percussion instrument is the backbone of most of Nepali folk music. The Madal originated in the Magar community of Nepal. Basically Madal can produce five different beats i.e. Dhing, Taang, Taak, Khaa and Dhaang.

The ancient name of Madal was Mardal. Typically a wooden log is carved so as to form a hollow cavity called Ghar. The heads of the drum are made of double-layered goat skins and a black paste made of flour, iron filings, and egg is burned in to a circular area in the center of each head. This circle adds weight to the head and significantly alters the sound of the drum, giving it a bell-like quality. The heads are fixed to the body of the drum by leather strips running the length of the body and an additional loose strip of leather which can be looped behind the performer's knees while playing. The larger and smaller heads are often referred to as male and female respectively.

Figure: Model of Madal

1.2 Object Oriented Programming

Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which may contain data in the form of fields, often known as attributes and code in the form of procedures, often known as methods. A feature of objects is that an object's procedures can access and often modify the data fields of the object with which they are associated. In OOP, computer programs are designed by making them out of objects that interact with one another. Beside object and class there are many other features of OOP like encapsulation, inheritance, polymorphism, etc. There is significant diversity of OOP languages, but the most popular ones are class-based, meaning that objects are instances of classes which typically also determine their type. Many of the most widely used programming languages are multi-paradigm programming languages that support object-oriented programming to a greater or lesser degree, typically in combination with imperative procedural programming. Significant object-oriented languages include Java, C++, C#, Python, PHP, Ruby, Perl, Delphi, Objective-C, Swift, Common Lisp, and Smalltalk.

|  |
| --- |
| Object2 |
| Data2 |
| Methods |

|  |
| --- |
| Object1 |
| Data1 |
| Methods |

|  |
| --- |
| Object3 |
| Data3 |
| Methods |

Figure: Interaction between Objects.

1.3 Simple DirectMedia Layer (SDL)

Simple DirectMedia Layer (SDL) is a cross-platform software development library designed to provide a low level hardware abstraction layer to computer multimedia hardware components. Software developers can use it to write high-performance computer games and other multimedia applications that can run on many operating systems such as Android, iOS, Linux, Mac OS X, Windows and other platforms. SDL manages video, audio, input devices, CD-ROM, threads, shared object loading, networking and timers. For 3D graphics it can handle an OpenGL or Direct3D context. The library is internally written in C and, depending on the target platform, C++ or Objective-C, and provides the application programming interface in C, with bindings to other languages available. It is free and open-source software subject to the requirements of the zlib License since version 2.0 and with prior versions subject to the GNU Lesser General Public License. Under the zlib License, SDL 2.0 is freely available for static linking in closed-source projects, unlike SDL 1.2. SDL is extensively used in the industry in both large and small projects. Over 700 games, 180 applications, and 120 demos have also been posted on the library website. A common misconception is that SDL is a game engine, but this is not true. However, the library is well-suited for building an engine on top of it.

SDL is a wrapper around the operating-system-specific functions that the game needs to access. The only purpose of SDL is to provide a common framework for accessing these functions for multiple operating systems (cross-platform). SDL provides support for 2D pixel operations, sound, file access, event handling, timing and threading. It is often used to complement OpenGL by setting up the graphical output and providing mouse and keyboard input, since OpenGL comprises only rendering.

Application (Multimedia)

SDL Library

etc

Xlib

Framebuffer

DirectX

Linux

Windows

etc

Hardware

Figure: Abstraction Layers of Several SDL Platforms

2.0 Problem Statement

We all know that Nepal is very rich in cultures and traditions so, it’s our responsibility to conserve and promote every aspects of our traditions. That might be the reason behind choosing such an interesting topic for our project. As the people are modernizing, they are gradually forgetting their valuable cultures. The influence of modernization can be directly observed in the music field as many of the old traditional musical instruments like Madal, Sarangi, etc. are being replaced by the new western instruments like Drums, Guitars, etc. There is the possibility that those culturally significant musical instruments could vanish some days. Seeking out the best possible solution for this problem, we came across the decision of modernizing our culture and tradition without losing their essence. To be more particular, we can promote and preserve Madal by just making the digital version of it so, that the new generation can easily access it and be more productive with it. A digital version of Madal is easy to use which indeed encourage people to make its excessive use.

3.0 Objective

We made a digital version of Madal in C++ programming language and the entire objective of our project was to preserve and promote our culturally significant musical instrument “Madal”. Similarly, we also wanted the new generation know about our culturally significant musical instruments. We wanted people to be more productive with the Madal. Likewise our important motive was to make people easily play Madal. As we all know, to play any musical instrument we need a specific skills and to gain that skills we need to go through a long training process which may be tedious and bit frustrating. Most of the people just give up during this training session. So, our digital version of Madal is the best solution for long training process. In our program we have use simple user interface that anyone can easily understand. The user only need to give some specific inputs and the rest job in done by program itself. So, it’s not a tedious job which was one of the primary objective of our project. Therefore, our project has the objective of modernizing our cultural and traditional aspects without losing their essence.

4.0 Scope and Limitation

As our program is for good cause, there is indeed a great scope. We all want to promote our cultures and traditions as they are our assets which allure many tourists and ultimately support in our economic development. Furthermore, our program also has the objective of preserving our traditions so, anyone working on the particular field can be massively advantaged. Likewise our program is the digital representation of musical instrument so, it can basically have a huge impact on the musical field. A new instrument with peculiar sound is always in demand in musical industry and through our program we can somehow meet that demand. Our program in reliable, easy to use and importantly works for conserving our traditional values so, it has a great scope.

However, the program we made was completely based on our knowledge and could include some limitations. First of all, our program does not have any feature of recording the played beats which can be a great issue for someone desiring of recording the beats. Likewise, the sound produced in our program is not totally clear and include some white noise because almost all the sounds that we have used in our program were downloaded from different online sources. Therefore the sound produced in our program doesn’t bear utmost clarity.

5.0 Implementation

We were able to make a digital version of Madal but how did we actually implement all the fundamentals of Madal in a digital program? If you are wondering about the same query then it’s time to clarify it. First of all let me brief you about the features of SDL we have used in program. Our entire program is based on some significant features provided by SDL like <SDL\_mixer.h> which include some flags and functions and allows us to play sound clips, we’ve also used vital features like SDL\_Window, SDL\_Surface, SDL\_Event, SDL\_LoadBMP, SDL\_BlitSurface, SDL\_CreateWindow, SDL\_GetWindowSurface, SDL\_PollEvent, SDL\_MOUSEBUTTONDOWN, SDL\_KEYDOWN, SDL\_GetKeyboardState, SDL\_GetMouseState, SDL\_BUTTON, SDL\_SCANCODE, SDL\_UpdateWindowSurface, SDL\_DestroyWindow under the <SDL.h> header file. In a similar way the program adjust all necessary parameters so that whenever the user input specified keystrokes or mouse click respective sounds are produced. In this way the complete program works.

5.1 Algorithms

1. Start

2. Initialize everything

SDL\_Init(SDL\_INIT\_EVERYTHING);

3. Create Window.

SDL\_CreateWindow(const char\* title, int x, int y, int w, int h, Uint32 flags);

4. Initialize Audio

Mix\_OpenAudio(44100,MIX\_DEFAULT\_FORMAT,2,2048);

5. Make objects for audio and image class.

Audio dhing("dhing.wav"), taang("taang.wav"), taak("taak.wav"), khaa("khaa.wav"), dhaang("dhaang.wav");

Image back("back.bmp), forg("forg.bmp"), forf("forf.bmp"), forh("forh.bmp"), forj("forj.bmp"), forspace("forspace.bmp");

6. Create an event

SDL\_Event ev;

7. While(SDL\_PollEvent(&ev)!=0)

{ if(ev.type==SDL\_MOUSEBUTTONDOWN)

{ int x,y;

SDL\_GetMouseState(&x,&y);

if(x>164&& x<249 && y>157&& y<246)

{

dhing.play();

}

else if(x>419&& x<506 && y>157&& y<246)

{

taang.play();

}

else if(x>294&& x<382 && y>210&& y<300)

{

dhaang.play();

}

else if(x>164&& x<253 && y>331&& y<424)

{

khaa.play();

}

else if(x>420&& x<509 && y>333&& y<423)

{

taak.play();

}

}

else if(ev.type == SDL\_KEYDOWN)

{ switch(ev.key.keysym.sym)

{

case SDLK\_f:

dhing.play();

break;

case SDLK\_g:

taang.play();

break;

case SDLK\_h:

taak.play();

break ;

case SDLK\_j:

khaa.play();

break;

case SDLK\_SPACE:

dhaang.play();

break;

}

}

8. Get the keyboard state and the mouse state.

press = SDL\_GetKeyboardState(NULL);

SDL\_GetMouseState(&x,&y);

9. if(press[SDL\_SCANCODE\_F] || (x>164&& x<249 && y>157&& y<246) && SDL\_BUTTON(1) )

{

forf.draw(windowSurface);

}

else if(press[SDL\_SCANCODE\_G] || (x>419&& x<506 && y>157&& y<246) && SDL\_BUTTON(1))

{

forg.draw(windowSurface);

}

else if(press[SDL\_SCANCODE\_H] || (x>420&& x<509 && y>333&& y<423) && SDL\_BUTTON(1))

{

forh.draw(windowSurface);

}

else if(press[SDL\_SCANCODE\_J] || (x>164&& x<253 && y>331&& y<424) && SDL\_BUTTON(1))

{

forj.draw(windowSurface);

}

else if(press[SDL\_SCANCODE\_SPACE] || (x>294&& x<382 && y>210&& y<300) && SDL\_BUTTON(1))

{

forspace.draw(windowSurface);

}

else

{

back.draw(windowSurface);

}

10. Update the window surface.

SDL\_UpdateWindowSurface(window);

11. Destroy all elements of windows.

SDL\_DestroyWindow(window);

12. End

5.2 Flowchart

Initialize Everything

SDL\_Init(SDL\_INIT\_EVERYTHING);

Create Window.

SDL\_CreateWindow(const char\* title, int x, int y, int w, int h, Uint32 flags);

Initialize Audio

Mix\_OpenAudio(44100,MIX\_DEFAULT\_FORMAT,2,2048);

Make objects for Image and Audio classes.

Audio dhing("dhing.wav"),……….dhaang("dhaang.wav");

Image back("back.bmp),…….. forspace("forspace.bmp");

Create an event

SDL\_Event ev;

bool running = true;

running = false

While(running)

False

if(ev.type==SDL\_MOUSEBUTTONDOWN)

SDL\_GetMouseState(&x,&y);

if(x>164&& x<249 && y>157&& y<246)

dhing.play();

……………….

else if(ev.type == SDL\_KEYDOWN)

switch(ev.key.keysym.sym)

case SDLK\_f

dhing.play();

break;

…………….

Get the keyboard state and the mouse state.

press = SDL\_GetKeyboardState(NULL);

SDL\_GetMouseState(&x,&y);

if(press[SDL\_SCANCODE\_F] || (x>164&& x<249 && y>157&& y<246) && SDL\_BUTTON(1) )

forf.draw(windowSurface);

………………….

Upadte the window surface.

SDL\_UpdateWindowSurface(window);

Destroy all elements of windows.

SDL\_DestroyWindow(window);

6.0 Conclusion and Enhancements

We are glad to have such an interesting topic for our third semester project under the Object Oriented Programming subject. The topic itself was challenging as we needed an intensive knowledge of SDL, OOP and also the working mechanism of real life Madal but we gave our best to complete our project. Likewise can contribute more by making a good user interface, reducing the noise in sound produced and reducing any flaws in the program.

Appendices

Appendix I: Image and Audio classes

class Image{

private:

SDL\_Surface \* image = NULL;

public:

Image(const char file[] = NULL)

{

image = SDL\_LoadBMP(file);

}

void draw(SDL\_Surface \*destination)

{

SDL\_BlitSurface(image,NULL,destination,NULL);

}

void destruct()

{

SDL\_FreeSurface(image);

image = NULL;

}

};

class Audio{

private:

Mix\_Chunk \*sound = NULL;

public:

Audio(const char filename[] = NULL)

{

sound = Mix\_LoadWAV(filename);

}

void play()

{

Mix\_PlayChannel(-1,sound,0);

}

void destroy()

{

Mix\_FreeChunk(sound);

sound = NULL;

}

};

Appendix II: Initialization of Everything

int main(int argc, char \*argv[] )

{

SDL\_Window \*window = NULL;

SDL\_Surface \*windowSurface = NULL;

const Uint8 \*press;

SDL\_Init(SDL\_INIT\_EVERYTHING);

window = SDL\_CreateWindow("MADAL",SDL\_WINDOWPOS\_CENTERED, SDL\_WINDOWPOS\_CENTERED,706,474, SDL\_WINDOW\_SHOWN);

windowSurface = SDL\_GetWindowSurface(window);

if(Mix\_OpenAudio(44100,MIX\_DEFAULT\_FORMAT,2,2048)<0)

cout<<"ERROR PLAYING AUDIO ):"<<Mix\_GetError()<<endl;

Audio dhing("dhing.wav"), taang("taang.wav"), taak("taak.wav"), khaa("khaa.wav"), dhaang("dhaang.wav");

Image back("back.bmp"), forg("forg.bmp"), forf("forf.bmp"), forh("forh.bmp"), forj("forj.bmp"), forspace("forspace.bmp");

back.draw(windowSurface);

Appendix III: Main Loop

bool running = true;

SDL\_Event ev ;

while(running)

{

while(SDL\_PollEvent(&ev)!=0)

{

if(ev.type == SDL\_QUIT)

running = false;

if(ev.type==SDL\_MOUSEBUTTONDOWN)

{

int x,y;

SDL\_GetMouseState(&x,&y);

if(x>164&& x<249 && y>157&& y<246)

{

dhing.play();

}

else if(x>419&& x<506 && y>157&& y<246)

{

taang.play();

}

else if(x>294&& x<382 && y>210&& y<300)

{

dhaang.play();

}

else if(x>164&& x<253 && y>331&& y<424)

{

khaa.play();

}

else if(x>420&& x<509 && y>333&& y<423)

{

taak.play();

}

}

else if(ev.type == SDL\_KEYDOWN)

{

switch(ev.key.keysym.sym)

{

case SDLK\_f:

dhing.play();

break;

case SDLK\_g:

taang.play();

break;

case SDLK\_h:

taak.play();

break ;

case SDLK\_j:

khaa.play();

break;

case SDLK\_SPACE:

dhaang.play();

break;

}

}

}

press = SDL\_GetKeyboardState(NULL);

int x,y;

SDL\_GetMouseState(&x,&y);

if(press[SDL\_SCANCODE\_F] || (x>164&& x<249 && y>157&& y<246) && SDL\_BUTTON(1) )

{

forf.draw(windowSurface);

}

else if(press[SDL\_SCANCODE\_G] || (x>419&& x<506 && y>157&& y<246) && SDL\_BUTTON(1))

{

forg.draw(windowSurface);

}

else if(press[SDL\_SCANCODE\_H] || (x>420&& x<509 && y>333&& y<423) && SDL\_BUTTON(1))

{

forh.draw(windowSurface);

}

else if(press[SDL\_SCANCODE\_J] || (x>164&& x<253 && y>331&& y<424) && SDL\_BUTTON(1))

{

forj.draw(windowSurface);

}

else if(press[SDL\_SCANCODE\_SPACE] || (x>294&& x<382 && y>210&& y<300) && SDL\_BUTTON(1))

{

forspace.draw(windowSurface);

}

else {

back.draw(windowSurface);

}

SDL\_UpdateWindowSurface(window);

}

Appendix IV: Destroying All Elements

SDL\_DestroyWindow(window);

dhing.destroy();

taak.destroy();

taang.destroy();

khaa.destroy();

dhaang.destroy();

back.destruct();

forf.destruct();

forg.destruct();

forh.destruct();

forj.destruct();

forspace.destruct();

window = NULL;

Mix\_Quit();

SDL\_Quit();

return 0;

Appendix V: User Interface of the Program





References and Bibliography

[1] Simple DirectMedia Layer – SDL version 2.0.4(stable) Libsdl.org

[2] Setting up SDL in Dev C++ - Lazy Foo’ Productions (lazyfoo.net)

[3] Learn Basic Madal – YouTube: John ukmusic (Uploader)

[4] Arjun Sing Saud - C++ Programming with Object Oriented Approach (2069)