happy-bird 音效控制 实验分报告

袁谱博

音效部分

方案1 OpenAL（未被采用）

`OpenAL` 是一个开源的音频硬件-软件接口库。该接口库中的函数允许程序员指定对象和操作，产生高质量的音频输出，特别能是在听众周围的声源营造3D排列的多声道输出效果。 OpenAL API旨在跨平台的实现且易于使用，其函数命名风格与语法均类似于OpenGL。

在本项目中使用的函数声明如下：

```
void alGenBuffers(
      ALsizei n,
      ALuint *buffers
);

void alDeleteBuffers(
      ALsizei      n,
      ALuint      *buffers
);

ALboolean alIsBuffer(
      ALuint      buffer
);

void alBufferData(
      ALuint      buffer,
      ALenum      format,
      const ALvoid *data,
      ALsizei      size,
      ALsizei      freq
);

void alBufferf(
      ALuint      buffer,
      ALenum      param,
      ALfloat      value
);
```

```
void alBuffer3f(
      ALuint      buffer,
      ALenum      param,
      ALfloat      v1,
      ALfloat      v2,
      ALfloat      v3
);

void alBufferfv(
      ALuint      buffer,
      ALenum      param,
      ALfloat      *values
);

void alBufferi(
      ALuint      buffer,
      ALenum      param,
      ALint      value
);

void alBuffer3i(
      ALuint      buffer,
      ALenum      param,
      ALint      v1,
      ALint      v2,
      ALint      v3
);

void alBufferiv(
      ALuint      buffer,
      ALenum      param,
      ALint      *values
);

void alGetBufferf(
      ALuint      buffer,
      ALenum      pname,
      ALfloat      *value
);

void alGetBuffer3f(
      ALuint      buffer,
      ALenum      pname,
      ALfloat      *v1,
      ALfloat      *v2,
      ALfloat      *v3
);

void alGetBufferfv(
      ALuint      buffer,
      ALenum      pname,
      ALfloat      *values
);
```

```
void alGetBufferi(
      ALuint      buffer,
      ALenum      pname,
      ALint      *value
);

void alGetBuffer3i(
      ALuint      buffer,
      ALenum      pname,
      ALint      *v1,
      ALint      *v2,
      ALint      *v3
);

void alGetBufferiv(
      ALuint      buffer,
      ALenum      pname,
      ALint      *values
);

void alGenSources(
      ALsizei      n,
      ALuint      *sources
);

void alDeleteSources(
      ALsizei      n,
      ALuint      *sources
);

boolean alIsSource(
      ALuint      source
);

void alSourcef(
      ALuint      source,
      ALenum      param,
      ALfloat      value
);

void alSource3f(
      ALuint      source,
      ALenum      param,
      ALfloat      v1,
      ALfloat      v2,
      ALfloat      v3
);

void alSourcefv(
      ALuint      source,
      ALenum      param,
      ALfloat      *values
```

```
);

void alSourcei(
      ALuint      source,
      ALenum      param,
      ALint      value
);

void alSourcei(
      ALuint      source,
      ALenum      param,
      ALint      v1,
      ALint      v2,
      ALint      v3
);

void alSourceiv(
      ALuint      source,
      ALenum      param,
      ALint      *values
);

void alGetSourcef(
      ALuint      source,
      ALenum      param,
      ALfloat      *value
);

void alGetSource3f(
      ALuint      source,
      ALenum      param,
      ALfloat      *v1,
      ALfloat      *v2,
      ALfloat      *v3
);

void alGetSourcefv(
      ALuint      source,
      ALenum      param,
      ALfloat      *values
);

void alGetSourcei(
      ALuint      source,
      ALenum      pname,
      ALint      *value
);

void alGetSource3i(
      ALuint      source,
      ALenum      param,
      ALint      *v1,
      ALint      *v2,
```

```
      ALint      *v3
);

void alGetSourceiv(
      ALuint      source,
      ALenum      param,
      ALint      *values
);

void alSourcePlay(
      ALuint      source
);

void alSourcePlayv(
      ALsizei      n,
      ALuint      *sources
);

void alSourcePause(
      ALuint      source
);

void alSourcePausev(
      ALsizei      n,
      ALuint      *sources
);

void alSourceStop(
      ALuint      source
);

void alSourceStopv(
      ALsizei      n,
      ALuint      *sources
);

void alSourceRewind(
      ALuint      source
);

void alSourceRewindv(
      ALsizei      n,
      ALuint      *sources
);

void alSourceQueueBuffers(
      ALuint source,
      ALsizei n,
      ALuint* buffers
);

void alSourceUnqueueBuffers(
      ALuint source,
```

```
      ALsizei n,
      ALuint* buffers
);

void alListenerf(
      ALenum      param,
      ALfloat      value
);

void alListener3f(
      ALenum      param,
      ALfloat      v1,
      ALfloat      v2,
      ALfloat      v3
);

void alListenerfv(
      ALenum      param,
      ALfloat      *values
);

void alListeneri(
      ALenum      param,
      ALint      value
);

void alListener3i(
      ALenum      param,
      ALint      v1,
      ALint      v2,
      ALint      v3
);

void alListeneriv(
      ALenum      param,
      ALint      *values
);

void alGetListenerf(
      ALenum      param,
      ALfloat      *value
);

void alGetListener3f(
      ALenum      param,
      ALfloat      *v1,
      ALfloat      *v2,
      ALfloat      *v3
);

void alGetListenerfv(
      ALenum      param,
      ALfloat      *values
```

```
);

void alGetListeneri(
      ALenum      param,
      ALint      *value
);

void alGetListener3i(
      ALenum      param,
      ALint      *v1,
      ALint      *v2,
      ALint      *v3
);

void alGetListeneriv(
      ALenum      param,
      ALint      *values
);

void alEnable(
      ALenum      capability
);

void alDisable(
      ALenum      capability
);

ALboolean alIsEnabled(
      ALenum      capability
);

ALboolean alGetBoolean(
      ALenum  param
);

Aldouble alGetDouble(
      ALenum  param
);

ALfloat     alGetFloat(
      ALenum  param
);

Alint alGetInteger(
      ALenum  param
);

void alGetBooleanv(
      ALenum      param,
      ALboolean      *data
);

void alGetDoublev(
```

```
      ALenum      param,
      ALdouble      *data
);

void alGetFloatv(
      ALenum      param,
      ALfloat      *data
);

void alGetIntegerv(
      ALenum      param,
      ALint      *data
);

const ALchar * alGetString(
      ALenum      param
);

void alDopplerFactor(
      ALfloat value
);

void alSpeedOfSound(
      ALfloat value
);

ALboolean alIsExtensionPresent(
      const ALchar *extname
);

void * alGetProcAddress(
      const ALchar *fname
);

ALenum alGetEnumValue(
      const ALchar *ename
);

ALCcontext * alcCreateContext(
      ALCdevice *device,
      ALCint* attrlist
);

ALCboolean alcMakeContextCurrent(
      ALCcontext *context
);

void alcProcessContext(
      ALCcontext *context
);

void alcSuspendContext(
      ALCcontext *context
```

```
);

void alcDestroyContext(
      ALCcontext *context
);
```

除此之外，OpenAL 还开发出了拓展库ALUT，对OpenAL 的底层操作进行了封装。

使用OpenAL实现游戏音效效果是我们最初考虑的方案。然而最后该方案没有调试成功，未被采用。

方案2 WinAPI

使用Windows内置函数，palySound 和 mciSendString 函数实现播放wav文件。

playSound 函数原型如下：

```
BOOL PlaySound(
   LPCTSTR pszSound,
   HMODULE hmod,
   DWORD   fdwSound
);
```

然而该函数的调用会阻塞对图像的渲染，即在播放音乐时画面会静止，而mciSendString 则避开了这一问题。

如果要向mciSendString 传递 std::string::c_str() 方法得到的字符串，需在VS2017 中取消使用unicode的的选项，并使用强制转换 LPCSTR。

注意二者的使用都应`#include<windows.h>` ,对于 `mciSendString` 函数， 还应 `include<mmsystem.h>` ,

并导入库： `#pragma comment<lib."winmm.lib">`

`mciSendString` 函数原型如下：

```
MCIERROR mciSendString(
   LPCTSTR lpszCommand,
   LPTSTR  lpszReturnString,
   UINT    cchReturn,
   HANDLE  hwndCallback
);
```

在Windows系统下实现计算wav格式文件，只需读取wav文件头，根据小端格式读取wav文件大小，声道数，编码方式以及频率。我们在每一帧的渲染中测试播放条件，如果满足则进行函数的播放。

经测试，此方案可正常实现游戏中的音效效果。但美中不足是无法在 Unix 平台下使用。

Bullet API

Bullet 是一个刚体与碰撞检测的开源物理引擎，我们在项目中大量使用了Bullet提供的接口，用于实现粒子系统与刚体碰撞。

- BulletRigidBody 提供移动、旋转操作，不考虑缩放的刚体接口
- BulletOpenCL 异构计算优化
- LinearMath 实现底层数学计算