计算机图形学 Project 报告

张珈芸 15307130339

文件组织

```
.
├── ./PJ1
|   ├── ./PJ1/main.py           // 源代码
|   ├── ./PJ1/demo.mov          // 截屏演示
|   └── ./PJ1/swan_lake.wav     // 音乐示例，需要.wav格式
├── ./PJ2
|   ├── ./PJ2/main.py           // 源代码
|   ├── ./PJ2/result.png        // 渲染结果
|   └── ./PJ2/texture.jpg       // 材质贴图
├── ./PJ3
|   ├── ./PJ3/flash.fla         // fla文件
|   └── ./PJ3/flash.mp4         // 导出的mp4格式视频
├── ./report.pdf
└── ./readme.md
```

1. 编程实现音乐节奏或旋律的可视化

- 使用 python 编程。使用到了 pyaudio, wave, librosa, numpy 库。
- pyaudio 和 wave 库用于读取和播放 .wav 格式音频。

```
# 创建播放器
p = pyaudio.PyAudio()
with wave.open(music_file, 'rb') as wf:
    # 打开音乐文件
    stream = p.open(
        format=p.get_format_from_width(wf.getsampwidth()),
        channels=wf.getnchannels(),
        rate=wf.getframerate(),
        output=True # 设置为True表示输出音频
    )
```

- numpy库用于傅里叶变换，将信号变换到频域。

```
# 读取音频数据
data = wf.readframes(nframes)
# 傅里叶变换，将信号变换到频域。
data_fft = np.real(np.fft.fft(np.fromstring(data, dtype=np.int16)))
```

- librosa 库用于获取音乐节奏。

```
y, sr = librosa.load(music_file)
tempo, beat_frames = librosa.beat.beat_track(y=y, sr=sr)
```

- pygame 库用于创建可视化窗口。使用`pygame.draw.rect()`函数，展示每个音频帧的频谱，并通过`color`参数和帧计数控制颜色变换。根据`beat_frames`在节奏点时刻增大可视化的bar高度，产生强调节奏点效果。

```
for n in range(0, data_fft.size, step):
    bar_height = min(abs(int(data_fft[n] * 1e-4)), 0.4 * window_height)
    # 根据节奏点增大bar_height，达到可视化节奏效果
    if iter in beat_frames:
        bar_height = min(bar_height + 5, 0.4 * window_height)
    pygame.draw.rect(screen, colors[int(n / data_fft.size * len(colors) - iter / 2) % len(colors)], pygame.Rec
    pygame.draw.rect(screen, colors[int(n / data_fft.size * len(colors) - iter / 2) % len(colors)], pygame.Rec
```

- 运行方式：`python main.py swan_lake.wav`， `swan_lake.wav`可替换成其他 **.wav** 格式音频文件路径。

2. 编程画一个真实感静态景物

- 使用 python 语言编程。使用 OpenGL 库。
- 在`draw_board()`函数中定义了立方体的点、面、坐标，并绘制3个显示的面。

```
# 绘制3个显示的面
glBegin(GL_QUADS)
# 前
glNormal3f(0.0, 0.0, 1.0)
for f in faces[0]:
    glTexCoord2fv(coords[f])
    glVertex3fv(vertices[f])
# 上
glNormal3f(0.0, 1.0, 0.0)
for f in faces[1]:
    glTexCoord2fv(coords[f])
    glVertex3fv(vertices[f])
# 左
glNormal3f(-1.0, 0.0, 0.0)
for f in faces[2]:
    glTexCoord2fv(coords[f])
    glVertex3fv(vertices[f])
glEnd()
```

- 设置立方体表面的纹理。

```
# 读取材质贴图
img = Image.open(filename)
img = np.asarray(img, dtype=np.uint8)
# 生成纹理
texture = glGenTextures(1)
# # 将texture纹理绑定到GL_TEXTURE_2D纹理目标上
glBindTexture(GL_TEXTURE_2D, texture)
# 纹理坐标超出边界时，采用GL_REPEAT
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT)
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT)
# 绘制图像大于或小于贴图尺寸时，采用GL_NEAREST
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST)
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST)
# 定义材质图片
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, img.shape[0], img.shape[1], 0, GL_RGB, GL_UNSIGNED_BYTE, img)
glEnable(GL_TEXTURE_2D)
glEnable(GL_TEXTURE_GEN_S)
glEnable(GL_TEXTURE_GEN_T)
```

- 设置光源

```
glShadeModel(GL_SMOOTH)
# 制定光源位置
glLightfv(GL_LIGHT0, GL_POSITION, [-.5, .5, 1.5, 1.])
# 设置材质对各种光的反光率
glMaterialfv(GL_FRONT, GL_DIFFUSE, [1., 1., 1., 1.])
glEnable(GL_LIGHTING)
glEnable(GL_LIGHT0)
glEnable(GL_DEPTH_TEST)
glDisable(GL_COLOR_MATERIAL)
```

3. 创作一个Flash动画

- 使用 Adobe Animate CC 创作flash动画。
- 角色和基础场景通过手绘完成。