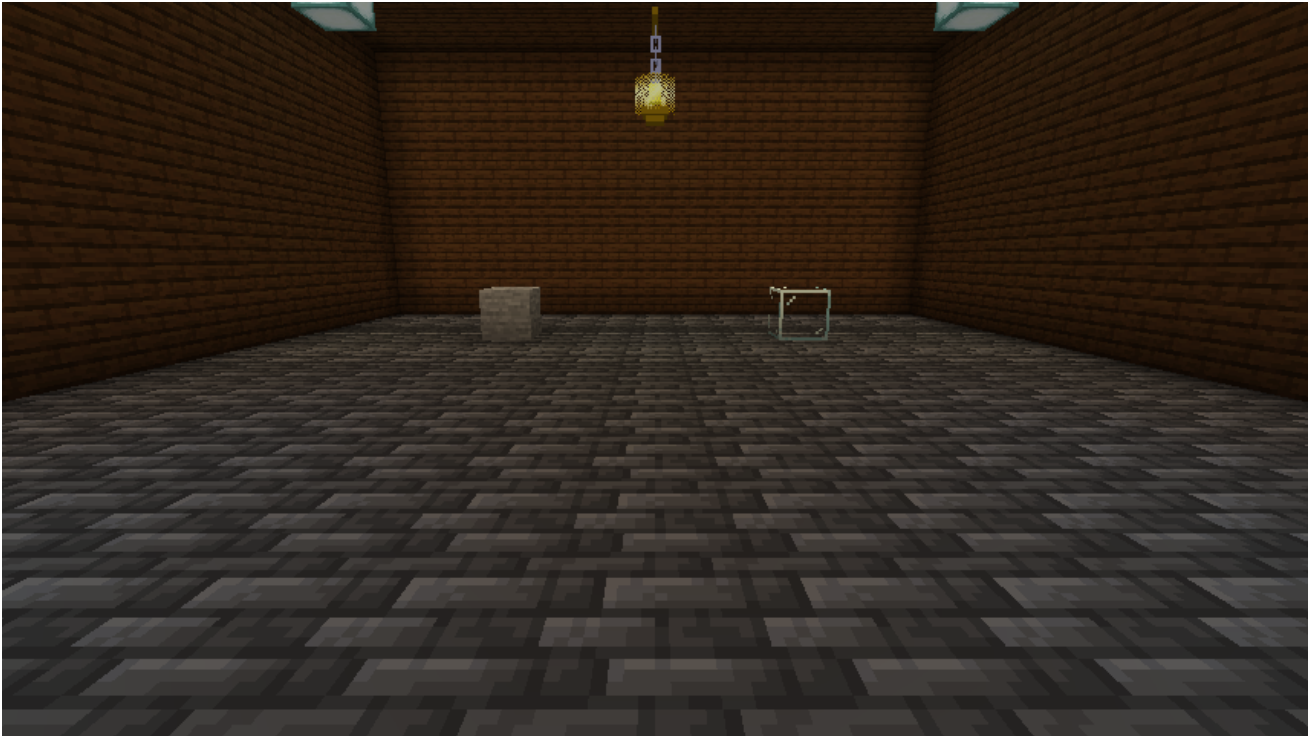


FABRIC 1.20 / JDK 21

第 4 课

给方块添加真实属性

硬度、发光、透明与右键声音



原教材中的真实 Fabric 客户端截图。

本课作品

制作会发光、透明边缘正确，并能右键播放音效或自己录音的吊灯。

150–180 分钟 | [simon-minecraft-template-1.20](#)

从材料对照实验走到录音导入，最后用六个面触发六种声音。

前置知识：完成第 3 课，吊灯模型、贴图、注册与客户端渲染均正常。

teacher-ma.de/minecraft-mod-lesson.php?lesson=4&lang=zh

01 / 16

先比较属性，再给方块配音

在第 3 课吊灯的基础上，一次增加一个可验证的能力。

模型决定外观，Settings 决定材料手感、发光和透明。完成对照实验后，再让吊灯右键播放声音：先验证原版音效，再接入自己的录音。

1. 比较硬度、亮度与透明效果，保留第 3 课模型与贴图。
 2. 区分材料音效与交互音效，让每次右键只触发一次播放。
 3. 把单声道 OGG、sounds.json 与 SoundEvent 连接起来。
1. 材料硬度 / 2. 合适工具 / 3. 材料音效 / 4. 亮度 / 5. 透明渲染

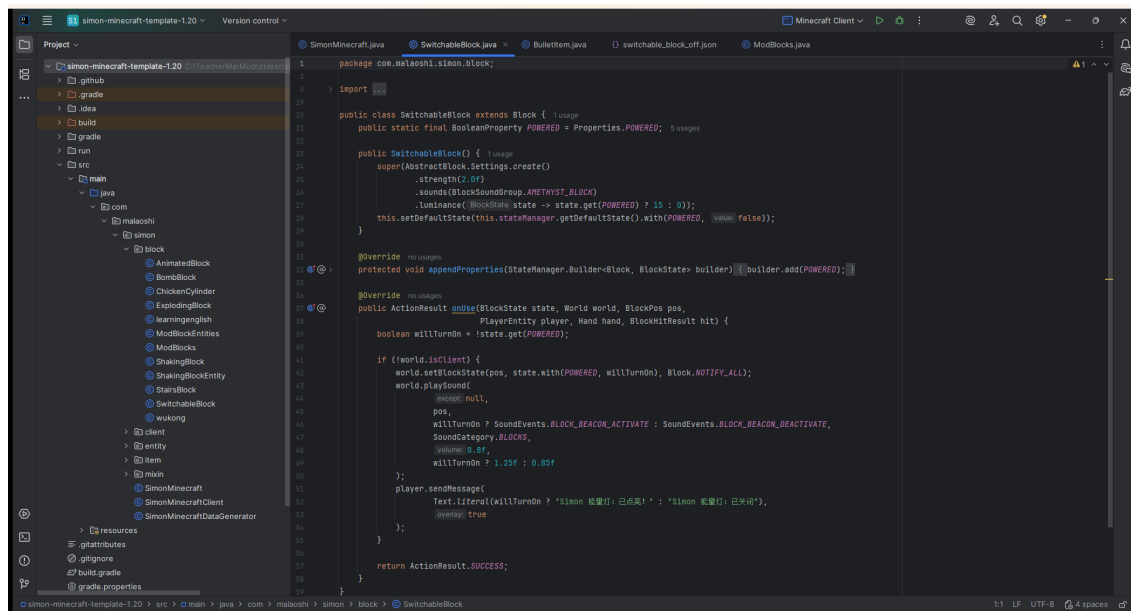
分阶段完成

先完成属性；再完成原版右键音效；最后导入录音并尝试六面声音。每次只改一处，保存测试结果。

02 / 16

在 IntelliJ 定位文件

继续使用同一个 Fabric 1.20 工程。



真实 IntelliJ 工程：项目树、Java 编辑器与 Gradle 工程。

本课文件 · 相对于工程根目录

```
src/main/java/com/malaoshi/simon/block/ModBlocks.java
src/main/java/com/malaoshi/simon/SimonMinecraftClient.java
src/main/resources/assets/simon-minecraft/models/block/light.json
src/main/resources/assets/simon-minecraft/textures/block/light_texture.png
src/main/resources/assets/simon-minecraft/textures/block/light_upper.png
```

版本锁定

本课固定使用 Minecraft 1.20、Yarn 1.20+build.1、Fabric API 0.83.0+1.20。指定模板使用 JDK 21 运行，Java 编译目标为 17；不要混用新版教程的方法签名。

03 / 16

用 Settings 描述基础行为

先检查原有吊灯注册，不要再注册第二个 light。

```
ModBlocks.java · LIGHT_BLOCK

LIGHT_BLOCK = registerBlock("light",
    new Block(FabricBlockSettings.create()
        .strength(1.0f)
        .luminance(state -> 15)
        .nonOpaque()));
```

strength(1.0f)	设置破坏硬度。
luminance(state -> 15)	返回 0–15 的亮度；这里始终返回 15。
nonOpaque()	声明它不是不透明整方块，配合客户端透明渲染。

04 / 16

为透明边缘选择 Cutout

客户端渲染只放在客户端入口。

SimonMinecraftClient.java · 类结构示例

```
public class SimonMinecraftClient
    implements ClientModInitializer {
    @Override
    public void onInitializeClient() {
        BlockRenderLayerMap.INSTANCE.putBlock(
            ModBlocks.LIGHT_BLOCK,
            RenderLayer.getCutout());
    }
}
```

把 putBlock 调用合并到现有 onInitializeClient(), 保留其他初始化。需要的 import 为 ClientModInitializer、BlockRenderLayerMap、RenderLayer 和 ModBlocks; 渲染相关 import 与第 3 课相同。

两个设置配合使用

nonOpaque() 描述材料遮挡属性, RenderLayer.getCutout() 控制如何绘制透明像素。透明边缘显示错误时要同时核对 PNG 的 alpha 与这两处代码。

05 / 16

常用属性速查

不要把材料行为、光照与交互音效混在一起。

strength(hardness)	控制破坏时间；双参数版本可分别指定硬度与爆炸抗性。
requiresTool()	要求正确工具；还需配合工具标签与掉落表，否则可能挖掉却不掉落。
sounds(...)	设置放置、脚步、破坏等材料音效，不会自动实现右键播放。
luminance(...)	返回 0–15，可按 BlockState 决定亮度。
nonOpaque() + Cutout	处理非整方块遮挡与透明像素绘制。

亮度不等于透明度

luminance 控制发光，PNG alpha 与渲染层控制透明。把亮度调高不会修复黑色透明边缘。

06 / 16

在游戏中比较属性

用石头、吊灯与玻璃做对照。



真实游戏截图：石头在左、发光吊灯在中、透明玻璃在右。

1. 在暗处观察吊灯照明范围，确认亮度设置生效。
2. 检查透明边缘没有黑块，并与玻璃对比。
3. 在生存模式比较不同 strength 的挖掘速度。
4. 修改资源后重载，修改 Java 后重启，再观察变化。

原教材截图由真实 Fabric 客户端保存；当时的临时截图代码已撤销，模板重新执行 build 成功。这里保留原始实机证据，你仍需验证自己的改动。

07 / 16

属性排错与对照实验

确认基础属性后，再进入声音工作坊。

吊灯不亮	核对 luminance 返回 15，修改 Java 后重启客户端。
透明区域发黑	检查 PNG alpha、nonOpaque() 与 Cutout。
徒手挖掉不掉落	检查 requiresTool()、工具标签与掉落表。
RenderLayer 报错	确认渲染代码位于 ClientModInitializer，不在服务端入口。

对照挑战

在复制的测试世界中依次测试 strength(0.2f)、strength(3.0f) 和 requiresTool()，记录挖掘时间与掉落结果。一次只改一个条件。

08 / 16

声音工作坊：材料声与交互声

先选一个可靠的原版声音，排除录音资源问题。

Settings.sounds(...) 控制放置、脚步、破坏等材料声音。

Block.onUse(...) 处理右键交互，可以主动播放指定音效。两者可以共存。

1. 玩家右键触发 onUse / 2. 服务端只播放一次 / 3. 附近客户端播放本地资源

先用紫水晶清脆声验证右键流程，再换录音。这样可以分清“交互没触发”和“音频路径或编码错误”。以下 onUse 签名适用于指定的 Minecraft 1.20 模板。

09 / 16

新建响应右键的吊灯类

新建 SoundLampBlock.java, 使用完整类代码。

文件位于 src/main/java/com/malaoshi/simon/block/SoundLampBlock.java。

SoundLampBlock.java · 完整文件

```
package com.malaoshi.simon.block;

import net.minecraft.block.Block;
import net.minecraft.block.BlockState;
import net.minecraft.entity.player.PlayerEntity;
import net.minecraft.sound.SoundCategory;
import net.minecraft.sound.SoundEvent;
import net.minecraft.util.ActionResult;
import net.minecraft.util.Hand;
import net.minecraft.util.hit.BlockHitResult;
import net.minecraft.util.math.BlockPos;
import net.minecraft.world.World;

public class SoundLampBlock extends Block {
    private final SoundEvent sound;

    public SoundLampBlock(Settings settings, SoundEvent sound) {
        super(settings);
        this.sound = sound;
    }

    @Override
    public ActionResult onUse(BlockState state, World world,
        BlockPos pos, PlayerEntity player, Hand hand,
        BlockHitResult hit) {
        if (hand != Hand.MAIN_HAND) return ActionResult.PASS;
        if (!world.isClient) {
            world.playSound(null, pos, sound,
                SoundCategory.BLOCKS, 0.8f, 1.0f);
        }
        return ActionResult.success(world.isClient);
    }
}
```

为什么只播放一次

主手检查避免双手重复调用; !world.isClient 让播放只发生在服务端; playSound 的第一个参数为 null, 表示不排除任何附近玩家, 包括点击者。success 表示这次交互已处理。

10 / 16

在原注册位置指定声音

用 SoundLampBlock 替换原来的 new Block。

ModBlocks.java · 替换原有注册

```
// Import at the top of ModBlocks.java:
import net.minecraft.sound.SoundEvents;

// Replace the existing LIGHT_BLOCK registration; do not add a second one.
LIGHT_BLOCK = registerBlock("light",
    new SoundLampBlock(FabricBlockSettings.create()
        .strength(1.0f)
        .luminance(state -> 15)
        .nonOpaque(),
        SoundEvents.BLOCK_AMETHYST_BLOCK_CHIME));
```

保留 registerBlock 辅助方法和创造栏回调，只替换 light 原来的构造部分，不要新增第二次注册。SoundLampBlock 与 ModBlocks 在同一包中。

BLOCK_AMETHYST_BLOCK_CHIME 清脆的紫水晶声。

ENTITY_PLAYER_LEVELUP 升级提示音。

BLOCK_ANVIL_LAND 铁砧落地声。

ENTITY_CAT_AMBIENT 猫叫声。

BLOCK_CHEST_OPEN 箱子打开声。

音量与音调

示例音量为 0.8、音调为 1.0。提高音调会更高、更快，降低音调会更低、更慢。改 Java 后重新 runClient，F3+T 不能更新 Java 类。

11 / 16

把自己的录音变成游戏资源

先做一段短而清晰的单声道录音。

录制 1–3 秒，例如“欢迎来到我的基地”。先听一遍，剪掉首尾静音与明显噪声；与麦克风保持合适距离，避免爆音。保存可继续编辑的音频工程，再单独导出游戏文件。

1. 录制并试听 / 2. 裁剪、转单声道 / 3. 导出 Ogg Vorbis / 4. 保存 welcome_01.ogg

在 Audacity 中通过“文件 → 导出 → 导出音频”选择 Ogg Vorbis，并导出 Mono（单声道）。采样率可以使用 44.1 kHz；不同版本界面的菜单名称可能略有不同。

格式与空间声音

应导出真正的 Ogg Vorbis，不是 Ogg Opus，也不是把 MP3 扩展名改成 .ogg。单声道适合从方块位置播放，让玩家感知距离和方向；立体声不适合这个实验。

用小写字母、数字和下划线命名 welcome_01.ogg，避免空格与中文文件名。先用自己的短录音完成测试；长音频与 stream 以后再研究。

12 / 16

用 sounds.json 连接事件与文件

声音事件 ID、资源名和文件路径是三个不同层次。

资源目录

```
src/main/resources/  
  assets/simon-minecraft/  
    sounds.json  
    sounds/  
      voice/  
        welcome_01.ogg  
  lang/  
    zh_cn.json  
    en_us.json
```

```
assets/simon-minecraft/sounds.json  
  
{  
  "my_voice": {  
    "subtitle": "subtitles.simon-minecraft.my_voice",  
    "sounds": [  
      {  
        "name": "simon-minecraft:voice/welcome_01",  
        "stream": false  
      }  
    ]  
  }  
}
```

事件 ID	Java 使用 <code>simon-minecraft:my_voice</code> ， <code>sounds.json</code> 的键为 <code>my_voice</code> 。
资源名	<code>name</code> 为 <code>simon-minecraft:voice/welcome_01</code> ，不是 Windows 路径。
实际文件	对应 <code>sounds/voice/welcome_01.ogg</code> 。 <code>name</code> 中不要再写 <code>sounds/</code> 或 <code>.ogg</code> 。

合并已有声音配置

如果 `sounds.json` 已存在，把 `my_voice` 合并进同一个 JSON 对象并检查逗号，不要把两份完整 JSON 直接串起来。
`stream: false` 适合本课的短音效。

13 / 16

注册 SoundEvent 并初始化

新建类后，在已有入口调用一次。

```
src/main/java/com/malaoshi/simon/sound/ModSounds.java

package com.malaoshi.simon.sound;

import net.minecraft.registry.Registries;
import net.minecraft.registry.Registry;
import net.minecraft.sound.SoundEvent;
import net.minecraft.util.Identifier;

public final class ModSounds {
    private static final Identifier VOICE_ID =
        new Identifier("simon-minecraft", "my_voice");
    public static final SoundEvent MY_VOICE = Registry.register(
        Registries.SOUND_EVENT, VOICE_ID,
        SoundEvent.of(VOICE_ID));

    private ModSounds() {}

    public static void registerModSounds() {
        // Calling this method initializes the static sound event.
    }
}
```

SimonMinecraft.java · 合并进现有入口

```
// Import at the top of SimonMinecraft.java:
import com.malaoshi.simon.sound.ModSounds;

// In the existing onInitialize(), before registering blocks:
ModSounds.registerModSounds();
ModBlocks.registerModBlocks();
// Keep the other initialization calls already in this template.
```

注册事件不等于加载音频

Registry 注册声音事件 ID，sounds.json 将 ID 连接到资源文件。registerModSounds() 的作用是触发类初始化，静态注册只执行一次。保留原有初始化调用，不要创建第二个 onInitialize()。

14 / 16

让吊灯播放你的欢迎词

替换声音参数，并提供中英文字幕。

ModBlocks.java · 使用自己的录音

```
// Import at the top of ModBlocks.java:
import com.malaoshi.simon.sound.ModSounds;

// Replace the sound argument in the same LIGHT_BLOCK registration:
LIGHT_BLOCK = registerBlock("light",
    new SoundLampBlock(FabricBlockSettings.create()
        .strength(1.0f)
        .luminance(state -> 15)
        .nonOpaque(), ModSounds.MY_VOICE));
```

lang/zh_cn.json · 合并字幕键

```
{
  "subtitles.simon-minecraft.my_voice": "\u540a\u706f\u5f1a\u6b22\u8fce\u6765\u5230\u6211\u7684\u57fa\u5730"
}
```

lang/en_us.json · 合并字幕键

```
{
  "subtitles.simon-minecraft.my_voice": "Lamp: Welcome to my base"
}
```

中文转义表示“吊灯：欢迎来到我的基地”。在游戏辅助功能设置中打开字幕，才能看到对应语言的文字。字幕不会自动把文字转成声音；声音来自你导出的 OGG 文件。

1. 注册 MY_VOICE 事件 / 2. 用 JSON 连接 OGG / 3. 右键触发播放

所有客户端都需要资源

服务端发送声音事件，不会把录音临时传给玩家。每位玩家的客户端都需要包含同一声音事件与音频资源的 MOD；只有服务端有文件时，其他玩家听不到录音。

15 / 16

进阶：六个面，六种声音

替换整个 onUse 方法，保留类、字段和构造器。

SoundLampBlock.java · 新增 import

```
import com.malaoshi.simon.sound.ModSounds;
import net.minecraft.sound.SoundEvents;
```

SoundLampBlock.java · 完整替换 onUse

```
@Override
public ActionResult onUse(BlockState state, World world,
    BlockPos pos, PlayerEntity player, Hand hand,
    BlockHitResult hit) {
    if (hand != Hand.MAIN_HAND) return ActionResult.PASS;
    if (!world.isClient) {
        SoundEvent chosen = switch (hit.getSide()) {
            case UP -> ModSounds.MY_VOICE;
            case DOWN -> SoundEvents.ENTITY_PLAYER_LEVELUP;
            case NORTH -> SoundEvents.BLOCK_AMETHYST_BLOCK_CHIME;
            case SOUTH -> SoundEvents.ENTITY_CAT_AMBIENT;
            case WEST -> SoundEvents.BLOCK_CHEST_OPEN;
            case EAST -> SoundEvents.BLOCK_ANVIL_LAND;
        };
        world.playSound(null, pos, chosen,
            SoundCategory.BLOCKS, 0.8f, 1.0f);
    }
    return ActionResult.success(world.isClient);
}
```

上面播放录音，下面播放升级音，北面紫水晶、南面猫叫、西面开箱、东面铁砧。hit.getSide() 是方块的世界方向，不是玩家视角里的左右；按 F3 核对方向。

一次选择一种声音

保留主手与服务端检查，并移除旧的无条件播放。快速连点会让多段音频重叠；测试时等上一段结束，后续可以尝试增加冷却。

验证声音与排查故障

先检查资源，再检查右键交互。

```
游戏聊天框 · 单独测试声音事件

/playsound simon-minecraft:my_voice block @s ~ ~ ~ 0.8 1
```

命令能播放后，再双手空着右键吊灯，避免手持物品的交互或潜行影响测试。

事件不存在	确认 ModSounds 已接入入口并重启客户端。
原版能响，录音不响	检查 sounds.json、资源名与 OGG 文件路径。
资源重载或解码失败	核对 JSON 语法，确认音频确实导出为 Ogg Vorbis。
一次点击响两次	检查主手与服务端条件，确认旧的播放代码已被替换。
声音没有距离感	使用 Mono，并检查游戏的方块音量与音效设置。
朋友听不到	确认对方客户端安装了同一声音事件与资源的 MOD。

1. 让一位附近玩家一起测试：点击者和旁边玩家都应听到声音。
2. 依次点击六个面，记录预期声音与实际声音。
3. 只改资源时更新构建输出并尝试 F3+T；改 Java 时停止并重新运行客户端。

官方参考

教程按指定的 Fabric 1.20 模板编写；查阅文档时请保持版本一致。

Fabric Wiki：声音

Yarn 1.20 API

Audacity：Ogg Vorbis 导出