

FABRIC 1.20 × 我的世界 MOD

第 10 课

制作会旋转的动态方块

BlockEntity 保存角度，Ticker 每刻更新



本课成就 建立 Block、BlockEntity、BlockEntityType、Ticker 与 Renderer 的完整动态方块链路。

建议 140-160 分钟 | [simon-minecraft-template-1.20](#) | Java 21

01 本课任务：只增加一个新能力

BlockState 适合有限状态；连续角度和每刻动画交给 BlockEntity

方向、开关和等级都只有少量合法值，适合 BlockState。旋转角度却会持续改变，如果为每个角度创建状态，资源数量会失控。BlockEntity 可以保存对象自己的数据，Ticker 每刻更新，Renderer 再把角度画出来。

本课三颗知识钻石

- 理解普通 Block 与 BlockEntity 的职责边界。
- 注册 BlockEntityType 并用 getTicker 连接 tick 方法。
- 在客户端渲染器中根据 angle 旋转模型。

1

继承带实体方块

2

创建实体

3

注册类型

4

连接 Ticker

5

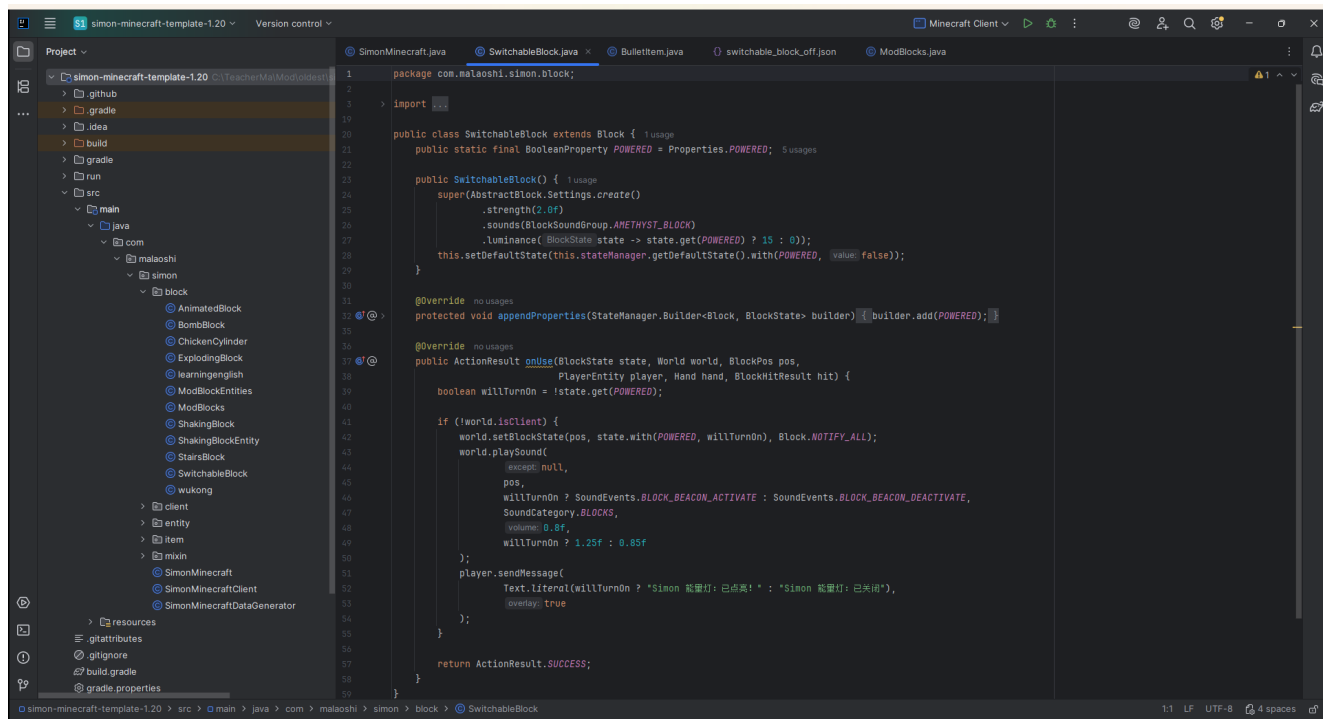
动态渲染

顺序感

第 7-9 课的属性仍然保留在 BlockState；只有连续变化或大量自定义数据进入 BlockEntity，不要把所有数据都搬过去。

02 在真实 IntelliJ 工程中定位文件

同一 simon-minecraft-template-1.20 工程界面，不是生成图



真实 IntelliJ 截图 · 左侧项目树 + Java 编辑器 + Gradle 工程

本课只打开这些位置

- src/main/java/com/malaoshi/simon/block/RotatingBlock.java
- src/main/java/com/malaoshi/simon/block/entity/RotatingBlockEntity.java
- src/main/java/com/malaoshi/simon/block/entity/ModBlockEntities.java
- src/main/java/com/malaoshi/simon/block/entity/RotatingBlockEntityRenderer.java
- src/main/java/com/malaoshi/simon/SimonMinecraftClient.java

版本锁定

Minecraft 1.20 · Yarn 1.20+build.1 · Fabric API 0.83.0+1.20。当前模板 fabric.mod.json 要求 Java 21。

03 方块创建实体，并把 Ticker 接上

先读懂，再逐行输入；不要整段复制后跳过理解

RotatingBlock.java + RotatingBlockEntity.java

```
public class RotatingBlock extends BlockWithEntity {
    public RotatingBlock(Settings settings) {
        super(settings);
    }

    @Override
    public BlockEntity createBlockEntity(
        BlockPos pos, BlockState state) {
        return new RotatingBlockEntity(pos, state);
    }

    @Override
    public <T extends BlockEntity> BlockEntityTicker<T>
    getTicker(World world, BlockState state,
        BlockEntityType<T> type) {
        return checkType(type,
            ModBlockEntities.ROTATING_BLOCK_ENTITY,
            RotatingBlockEntity::tick);
    }
}

public static void tick(World world, BlockPos pos,
    BlockState state, RotatingBlockEntity entity) {
    if (world.isClient) {
        entity.angle = (entity.angle + 2.0f) % 360.0f;
    }
}
```

程序怎么读

每个方块位置拥有独立实体和 angle。只在客户端增加渲染角度，避免把纯视觉动画反复同步给服务器。

04 渲染器读取 angle 并旋转模型

Java 与资源文件必须使用同一个 MOD_ID 和注册名

```
RotatingBlockEntityRenderer.java

@Override
public void render(
    RotatingBlockEntity entity, float tickDelta,
    MatrixStack matrices,
    VertexConsumerProvider providers,
    int light, int overlay) {

    matrices.push();
    matrices.translate(0.5, 0.5, 0.5);
    matrices.multiply(RotationAxis.POSITIVE_Y
        .rotationDegrees(entity.angle + tickDelta * 2));
    matrices.translate(-0.5, -0.5, -0.5);

    BlockState state = entity.getCachedState();
    blockRenderManager.getModelRenderer().render(
        entity.getWorld(),
        blockRenderManager.getModel(state),
        state, entity.getPos(), matrices,
        providers.getBuffer(RenderLayers.getBlockLayer(state)),
        false, entity.getWorld().random, 0L,
        OverlayTexture.DEFAULT_UV);
    matrices.pop();
}
```

连接点

push/pop 保护外部矩阵；先移到方块中心再旋转。tickDelta 用于补间，让 20 TPS 的逻辑更新看起来更平滑。

05 动态方块的五段连接

Block → BlockEntity → Type → Ticker → Renderer

Block 决定碰撞、交互，并在位置上创建 BlockEntity。

BlockEntity 保存每个方块独立的 angle 等数据。

BlockEntityType 告诉注册表哪一种实体属于哪些方块。

Ticker 每 tick 调用一次更新方法。

Renderer 客户端读取 angle，修改矩阵并绘制模型。

课堂检查

注册顺序很重要：先注册方块，再构建引用该方块的 BlockEntityType；客户端渲染器只能放在 client 入口注册。

06 进入游戏验证结果

真实 Fabric 客户端截图，由 Minecraft ScreenshotRecorder 保存



游戏实拍 · 模板 RotatingBlock 的 BlockEntityRenderer 正在旋转并伸缩模型

检查 1

放置后每个动态方块都独立旋转。

检查 2

动画只在客户端更新，不污染服务端逻辑。

检查 3

破坏方块后对应 BlockEntity 会一起移除。

检查 4

重新构建后客户端与服务器类没有环境冲突。

实机证据

截图来自本地 runClient 的复制测试世界；课程生成后临时截图代码已撤销，模板重新执行 Gradle build 成功。

07 故障地图与通关挑战

方块完全不显示

确认 `getRenderType` 与渲染器方案一致。

`angle` 永远是 0

确认 `getTicker` 返回正确类型并调用 `tick`。

整个场景一起旋转

`render` 开头 `push`，结束前 `pop`。

服务器启动崩溃

渲染器注册只能放在 `ClientModInitializer`。

Boss 挑战

增加旋转速度 `speed`，并在 `writeNbt/readNbt` 中保存；退出世界再进入后，验证每个方块仍保留自己的速度。

下一课

第 11 课可以继续学习 NBT 持久化与右键计数器，让 `BlockEntity` 数据跨存档保存。