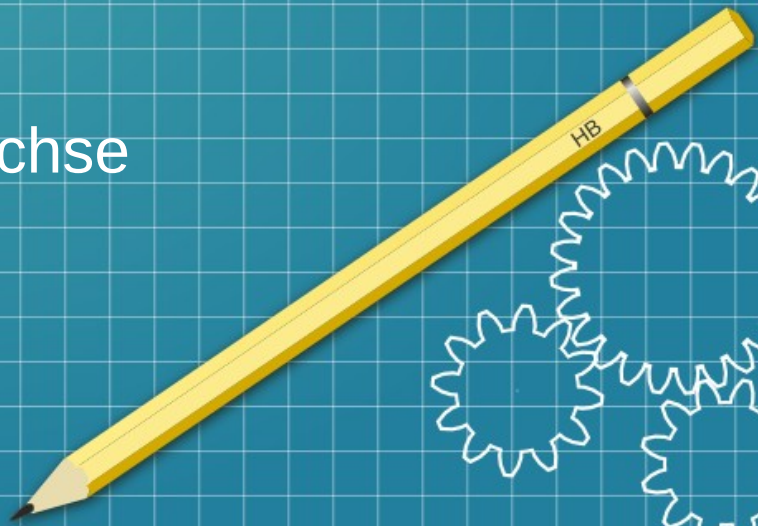
Bush

Büsche und solche Gewächse



Busch als spezieller Block

- Functional Interfaces
- Lambdas
- Idee: Weihnachtsbaum als Busch
  - mit Lebkuchen als Frucht
  - leuchtet im Dunkeln
  - wird mit einem Stummel gepflanzt
  - hat eine besondere Darstellung (cross)

Eigene Block-Klasse

- Erbt von SweetBerryBushBlock
  - Kann damit wachsen
  - Hat Früchte
  - Ist nicht blockierend, aber langsamer beim Durchlaufen

- onUse
  - Bei Rechtsklick
- getPickStack
  - Was beim Abbau rausfällt
- Erbt z.B. AGE

```java
public class ChristmasTree extends SweetBerryBushBlock {

  public ChristmasTree(Settings settings) {
    super(settings);
  }

  @Override
  protected ItemStack getPickStack(WorldView world, BlockPos pos,
      BlockState state, boolean includeData) {
    return new ItemStack(ModItems.GINGERBREAD);
  }

  @Override
  protected ActionResult onUse(BlockState state, World world,
      BlockPos pos, PlayerEntity player, BlockHitResult hit) {
    if (state.get(AGE) > 1) {
      dropStack(world, pos, new ItemStack(ModItems.GINGERBREAD));
      BlockState blockState = state.with(AGE, 1);
      world.setBlockState(pos, blockState, Block.NOTIFY_LISTENERS);
      world.emitGameEvent(GameEvent.BLOCK_CHANGE, pos,
          GameEvent.Emitter.of(player, blockState));
      return ActionResult.SUCCESS;
    } else {
      return super.onUse(state, world, pos, player, hit);
    }
  }
}
```

Functional Interfaces

- Packe mir eine Kiste mit einem „Ding“
- Wird erst später bekannt gegeben, was genau
- Aber das Ding kann schon was, z.B. hüpfen
- Später: Kiste packen
  - 1 Kiste mit einem Frosch
  - 1 Kiste mit einem Hasen
  - 1 Kiste mit einem Flummi

Lambda

- Verkürzte Schreibweise für einen Funktionsaufruf

```
List<String> tiere = Arrays.asList("Hund", "Katze", "Maus");

tiere.forEach(new Consumer<String>() {
  @Override
  public void accept(String x) {
    System.out.println(x);
  }
});

tiere.forEach(x -> System.out.println(x));
```

Registrieren von speziellen Blöcken

- Die Blockinstanz wird beim Erzeugen angegeben
  - x -> new ChristmasTree(x);
- aber erst später aufgerufen
  - function.apply(settings…)
- Auch bisherige Blöcke könnten so umgebaut werden
  - x -> new Block(x);

ModBlocks: ChristmasTree einbauen

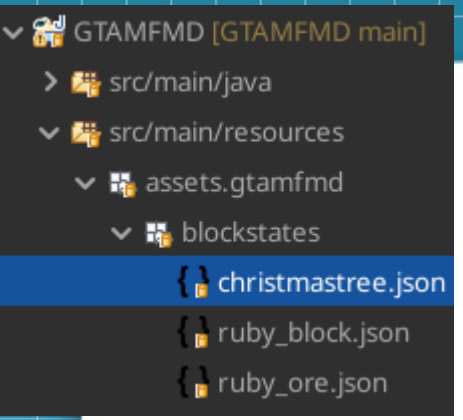
```
public static final Block CHRISTMASTREE = registerSpecialBlock(
    Identifier.of(Gtamfmd.MOD_ID, "christmastree"),
    AbstractBlock.Settings.create().ticksRandomly().noCollision().luminance(state -> 15),
    x -> new ChristmasTree(x));

private static Block registerSpecialBlock(Identifier identifier, Settings settings,
    Function<Settings, Block> function) {
  Block block = function.apply(settings.
      registryKey(RegistryKey.of(RegistryKeys.BLOCK, identifier)));
  registerBlockItem(identifier, block, new Item.Settings());
  return Registry.register(Registries.BLOCK, identifier, block);
}
```

- ticksRandomly() → wächst also zufällig
- noCollision() → kann man durchlaufen
- luminance → leuchtet (0 – 15)

Transparenz

- Damit der Weinnachtsbaum keinen schwarzen Hintergrund hat
  - Im Stammordner in der Client-Klasse, Methode onInitializeClient

```
public class GtamfmdClient implements ClientModInitializer {
  @Override
  public void onInitializeClient() {
    BlockRenderLayerMap.putBlock(ModBlocks.CHRISTMASTREE,
        BlockRenderLayer.CUTOUT);
  }
}
```

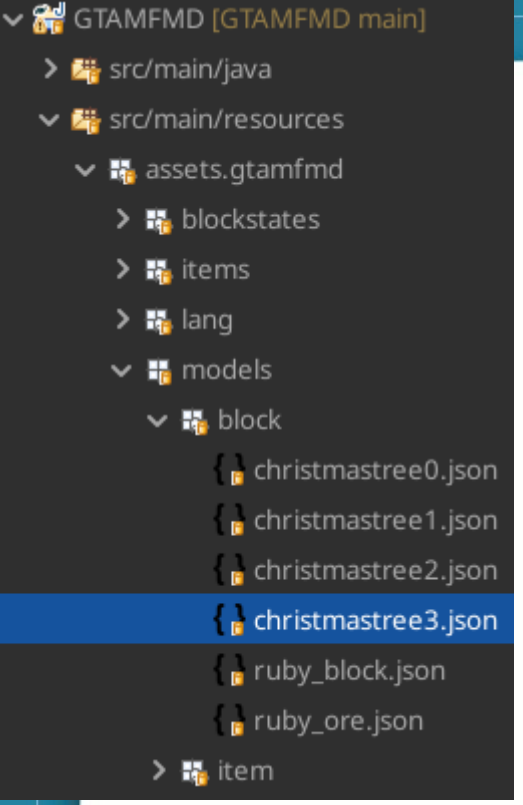


blockstate

Vier verschiedene Models, je nach age

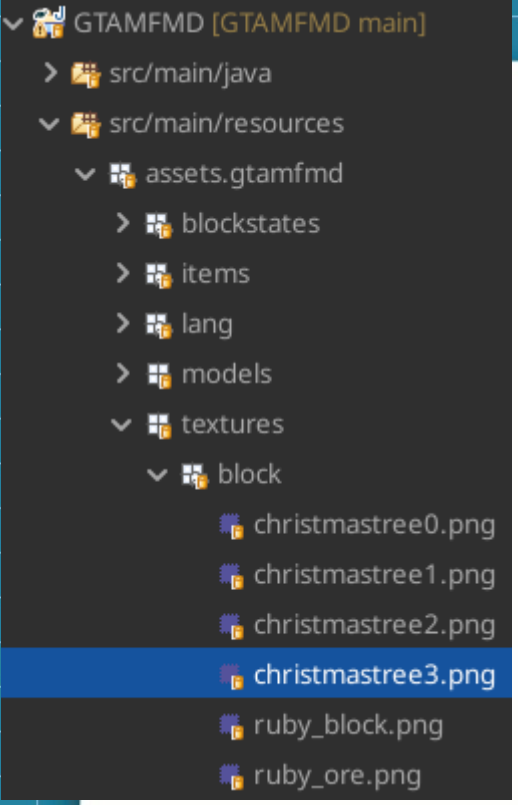
```
{
  "variants": {
    "age=0": { "model": "gtamfmd:block/christmastree0" },
    "age=1": { "model": "gtamfmd:block/christmastree1" },
    "age=2": { "model": "gtamfmd:block/christmastree2" },
    "age=3": { "model": "gtamfmd:block/christmastree3" }
  }
}
```

Block models

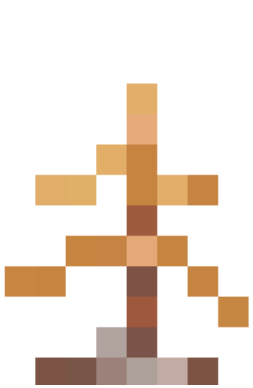



```
{
  "parent": "minecraft:block/cross",
  "textures": {
    "cross": "gtamfmd:block/christmastree3"
  }
}
```

Ebenso für die anderen 3 models



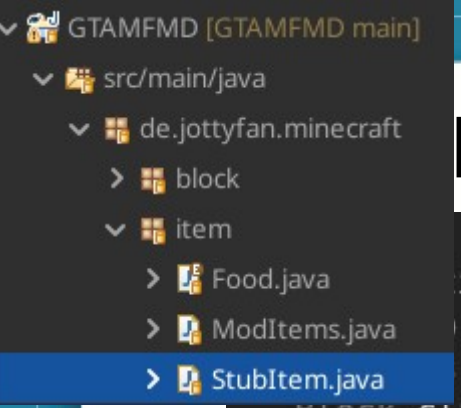
Texturen



Je model ein Bild

Gepflanzt aus Stummel

- Klasse ModStub erweitern um Pflanzaktion
  - Wenn der Block, auf den gehauen wird, Erde ist, dann pflanzen
  - Blöcke prüfen; darüber muss Luft sein
  - Wenn ein Stummel gepflanzt wurde, muss er in der Hand des Spielers verschwinden (bzw. bei mehreren Stummeln um 1 erniedrigt werden)



Methode StubItem.useOnBlock

```
ionResult useOnBlock(ItemUsageContext context) {
rld = context.getWorld();
 pos = context.getBlockPos();
Block clickedBlock = world.getBlockState(pos).getBlock();
if (SLASH_MAP.containsKey(clickedBlock)) {
  if (!world.isClient()) {
  }
} else if (Blocks.DIRT.equals(clickedBlock) && world.getBlockState(pos.up()).isAir()) {
  if (!world.isClient()) {
    world.setBlockState(pos.up(), ModBlocks.CHRISTMASTREE.getDefaultState());
    world.playSound(null, pos, SoundEvents.ITEM_CROP_PLANT,
        SoundCategory.BLOCKS);
    Hand hand = context.getHand();
    PlayerEntity player = context.getPlayer();
    ItemStack stack = player.getStackInHand(hand);
    stack.increment(-1);
    player.setStackInHand(hand, stack);
  }
}
return ActionResult.SUCCESS;
}
```

Arbeit sichern

- Git commit und push