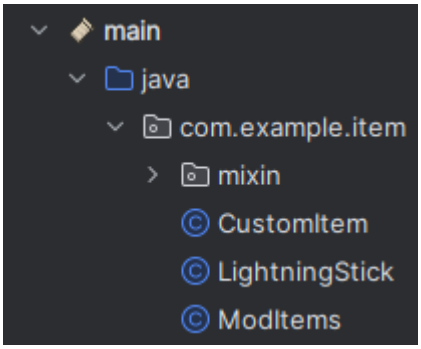
08 Blesková hůlka

V této lekci si ukážeme, jak upravit chování itemu, když jej hráč použije. Přidáme si tedy hůlku, která vytvoří blesk, když s ní hráč klikne pravým tlačítkem myši.

Abychom mohli upravit chování itemu, tak si nejprve pro item musíme vytvořit novou třídu. Tu můžeme pojmenovat například `LightningStick` Třídu vytvoříme kliknutím pravým tlačítkem myši na složku a zvolíme *New* > *Java Class*. Třídu vytvoříme ve stejné složce, jako máme naše ostatní třídy (`ModItems` , `CustomItem`).

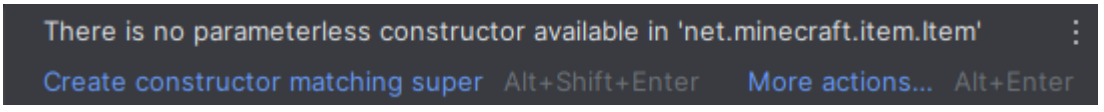



```
1   public class LightningStick {
2   }
```

Když vytváříme nový item, tak chceme, aby byl stejný jako všechny ostatní itemy ve hře a navíc měl nějaké námi upravenou funkčnost nebo vlastnosti. Proto musíme říct, že tato třída rozšiřuje třídu základního itemu. To uděláme tak, že za název třídy doplníme `extends Item` .

```
1   public class LightningStick extends Item {
2   }
```

Nyní nám editor bude psát následující chybu:



Tato chyba znamená, že třída `Item` má základní konstruktor (pro nastavení itemu), ale naše třída ho nemá. Stačí kliknout na *Create constructor matching super* a konstruktor se nám automaticky doplní. Případně si jej můžeme doplnit ručně.

```
1   public class LightningStick extends Item {
2       public LightningStick(Settings settings) {
3           super(settings);
4       }
5   }
```

Volání `super` znamená, že se volá metoda ze třídy, kterou rozšiřujeme.

Použití itemu

Abychom změnili, co se stane, když hráč item použije (klikne pravým tlačítkem myši), tak musíme použít jednu z předem definovaných metod, které u základního itemu nic nedělají. Máme dostupné následující metody:

- `use`
  - kliknutí pravým tlačítkem myši do vzduchu
- `useOnBlock`
  - kliknutí pravým tlačítkem na blok
- `useOnEntity`
  - kliknutí pravým tlačítkem na entitu

My použijeme metodu `use`, ostatní si ukážeme později.

Stačí napsat `use` a v nápovědě by se nám všechny tyto metody měly nabídnout. Stačí tedy vybrat tu, kterou chceme použít a opět se nám doplní část kódu.

```
1  public class LightningStick extends Item {
2      public LightningStick(Settings settings) {
3          super(settings);
4      }
5
6      @Override
7      public TypedActionResult<ItemStack> use(World world, PlayerEntity
   user, Hand hand) {
8          return super.use(world, user, hand);
9      }
10 }
```

> **Označení** `@Override`
>
> Tímto označujeme metodu, kterou přepisujeme chování základní třídy. V našem případě je to třída `Item`.

Můžeme si všimnout, že se nám automaticky doplnily i parametry metody. Tato metoda se volá automaticky vždy, když chce hráč použít item a Minecraft tuto metodu volá právě s těmito parametry. Díky tomu máme v metodě dostupné informace o světě `world`, o hráči, který item použil `user` a dokonce i informace o tom, ve které ruce hráč item drží `hand`.

Stejně jako v konstruktoru se v základu zavolá metoda z obecnější třídy (slovo `super`), to ale můžeme změnit a můžeme si napsat vlastní kód.

Kód, který chceme, aby se provedl při použití itemu budeme psát nad řádek s výrazem `return` (vrácení hodnoty). Vytvoříme si tedy nad tímto řádkem nový řádek, kam budeme psát kód.

Kontrola spuštění

Jako první musíme zkontrolovat, jestli se kód spouští u hráče nebo na serveru. To je kvůli tomu, že kdyby se kód spustil jen u hráče, tak by se mohlo stát, že se o tom server nedozví a tuto akci uvidí jen hráč nebo se naopak spustí dvakrát.

Tuto informaci máme v parametru `world.isClient`. Kontrolu provádíme pomocí výrazu `if`, kde do závorky napíšeme podmínku, kterou chceme zkontrolovat. Kód, který chceme aby se provedl, když bude podmínka splněná píšeme do složených závorek.

```
1  public TypedActionResult<ItemStack> use(World world, PlayerEntity user,
   Hand hand) {
2      if (world.isClient) {
3
4      }
5      return super.use(world, user, hand);
6  }
```

Pokud se kód spouští u hráče, tak nechceme dělat nic. Tato metoda musí vracet výsledek použití itemu. Možné hodnoty najdeme ve třídě `TypedActionResult`. Pokud nechceme dělat nic, tak použijeme metodu `pass`. Tam musíme jako parametr uvést informace o itemu, který hráč drží v ruce. Tyto informace získáme z proměnné `user` pomocí metody `getStackInHand`. Tam musíme předat informaci o tom, v jaké ruce použitý item drží - to máme v proměnné `hand`.

```
1  public TypedActionResult<ItemStack> use(World world, PlayerEntity user,
   Hand hand) {
2      if (world.isClient) {
3          return TypedActionResult.pass(user.getStackInHand(hand));
4      }
5      return super.use(world, user, hand);
6  }
```

Vytvoření blesku

Nyní se přesuneme k samotnému vytvoření blesku. Blesk se v Minecraftu bere jako entita. Na to, abychom nějakou entitu přidali do světa musíme určit, kde se má objevit (souřadnice) a poté určit, o jakou entitu se jedná.

Pozice

Jako pozici můžeme vzít pár bloků před hráčem. Vytvoříme si tedy proměnnou `blockPos` typu `BlockPos` a do ní si uložíme pozici hráče `user.getBlockPos()`. Tím bychom ale blesk vytvořili přímo na souřadnicích hráče, takže ještě musíme souřadnice o něco posunout. To uděláme pomocí metody `offset`, kde jako parametry zadáme směr, kterým chceme souřadnici posunout a o kolik bloků ji chceme posunout. Směr, kterým se hráč dívá zjistíme pomocí `user.getHorizontalFacing()` a vzdálenost zadáme jako celé číslo.

```
1  public TypedActionResult<ItemStack> use(World world, PlayerEntity user,
   Hand hand) {
2      if (world.isClient) {
3          return TypedActionResult.pass(user.getStackInHand(hand));
4      }
5
6          BlockPos blockPos =
   user.getBlockPos().offset(user.getHorizontalFacing(), 10);
7          return super.use(world, user, hand);
8  }
```

Entita blesku

Entitu blesku si uložíme do proměnné typu `LightningEntity` s názvem `lightning`. Do ní uložíme nový objekt třídy `LightningEntity`, kde jako parametry uvádíme typ entity a informace o světě, kam chceme entitu přidat. Typy entit máme dostupné ve třídě `EntityType` a blesk má hodnotu `LIGHTNING_BOLT`. Informace o světě máme uložené v parametru `world`.

```
1  public TypedActionResult<ItemStack> use(World world, PlayerEntity user,
   Hand hand) {
2      if (world.isClient) {
3          return TypedActionResult.pass(user.getStackInHand(hand));
4      }
5
6          BlockPos blockPos =
   user.getBlockPos().offset(user.getHorizontalFacing(), 10);
7
8          LightningEntity lightning = new
   LightningEntity(EntityType.LIGHTNING_BOLT, world);
9          return super.use(world, user, hand);
10 }
```

Přidání entity do světa

Nejprve musíme entitě nastavit pozice, kde se má objevit. To uděláme pomocí metody `setPosition`, která je dostupná u všech entit. Jako parametr uvádíme pozice ve světě. V proměnné `blockPos` máme uložené pozice bloku a ty můžeme převést na tvar, který

vyžaduje tato metoda tak, že použijeme metodu `toCenterPos()`, která nám vrátí souřadnice středu bloku.

```
public TypedActionResult<ItemStack> use(World world, PlayerEntity user, Hand hand) {
    if (world.isClient) {
        return TypedActionResult.pass(user.getStackInHand(hand));
    }

        BlockPos blockPos = user.getBlockPos().offset(user.getHorizontalFacing(), 10);

        LightningEntity lightning = new LightningEntity(EntityType.LIGHTNING_BOLT, world);

        lightning.setPosition(blockPos.toCenterPos());
        return super.use(world, user, hand);
}
```

Samotnou entitu přidáme pomocí metody `spawnEntity` ze třídy `World`, kde jako parametr uvedeme entitu, kterou chceme do světa přidat.

```
public TypedActionResult<ItemStack> use(World world, PlayerEntity user, Hand hand) {
    if (world.isClient) {
        return TypedActionResult.pass(user.getStackInHand(hand));
    }

        BlockPos blockPos = user.getBlockPos().offset(user.getHorizontalFacing(), 10);

        LightningEntity lightning = new LightningEntity(EntityType.LIGHTNING_BOLT, world);

        lightning.setPosition(blockPos.toCenterPos());
        world.spawnEntity(lightning);
        return super.use(world, user, hand);
}
```

Úspěšné použití itemu

Na závěr musíme vrátit informaci o tom, že se item úspěšně použil. Na to opět použijeme třídu `TypedActionResult`, ale tentokrát použijeme metodu `success` - úspěch. Stejně jako u neúspěšného použití i tady musíme jako parametr předat item, který hráč použil.

Původní řádek s `return super` můžeme smazat.

>

```java
public TypedActionResult<ItemStack> use(World world, PlayerEntity user, Hand hand) {
    if (world.isClient) {
        return TypedActionResult.pass(user.getStackInHand(hand));
    }

    BlockPos blockPos = user.getBlockPos().offset(user.getHorizontalFacing(), 10);

    LightningEntity lightning = new LightningEntity(EntityType.LIGHTNING_BOLT, world);

    lightning.setPosition(blockPos.toCenterPos());
    world.spawnEntity(lightning);

    return TypedActionResult.success(user.getStackInHand(hand));
}
```

Přidání itemu do hry

Dále musíme item ještě přidat do hry. To uděláme stejně jako u předchozího itemu ve třídě `ModItems` a použijeme na to naši metodu `register`. A jako texturu můžeme použít obyčejnou tyčku, která už ve hře je.

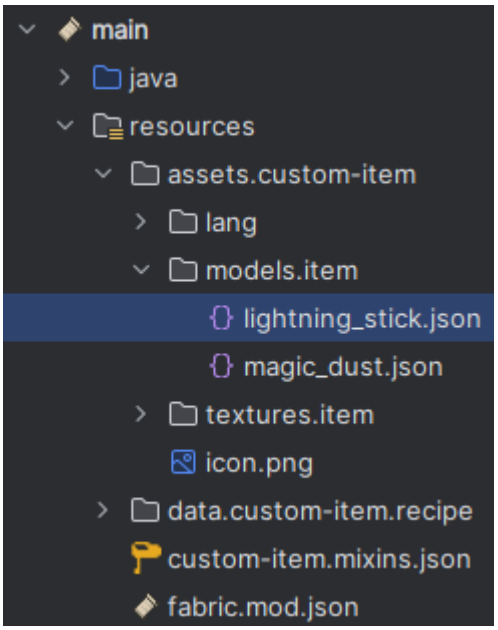
Registrace itemu

Rozdíl bude v tom, že nyní nemáme třídu `Item`, ale náš item používá třídu `LightningStick`. A proto, že naše metoda vrací jako hodnotu třídu `Item`, tak musíme určit, že ve skutečnosti se jedná o naši třídu `LightningStick` a ne o `Item` to uděláme tak, že před metodu `register` napíšeme do závorky `LightningStick`.

```java
public static final LightningStick LIGHTNING_STICK = (LightningStick) register(
        new LightningStick(new Item.Settings().maxCount(1)),
        "lightning_stick"
);
```

Přidání textury

Na přidání textury si musíme ve složce s modely vytvořit `json` soubor s názvem našeho itemu.



Tentokrát bude soubor velmi jednoduchý a jen určíme, že se má použít stejný model, jako pro item tyčky v Minecraftu.

```
{
  "parent": "minecraft:item/stick"
}
```

Když nyní spustíme hru, tak bychom měli mít nový item s id `custom-item:lightning_stick` a když s touto tyčkou klikneme pravým tlačítkem myši, tak by se před námi měl objevit blesk.

Item můžeme získat jen pomocí příkazu

```
/give @s custom-item:lightning_stick
```

Poznámka

Souřadnice blesku budou vždy posunuté ve směru jedné ze světových stran (sever, jih, východ, západ). To je způsobené tím, že metoda `getHorizontalFacing()` u hráče vrací pouze jednu ze světových stran a ne přesný směr, kterým se hráč dívá.

Kód by šlo upravit i tak, aby se blesk objevil opravu přímo před hráčem a bral se přesný směr, kterým se hráč dívá.