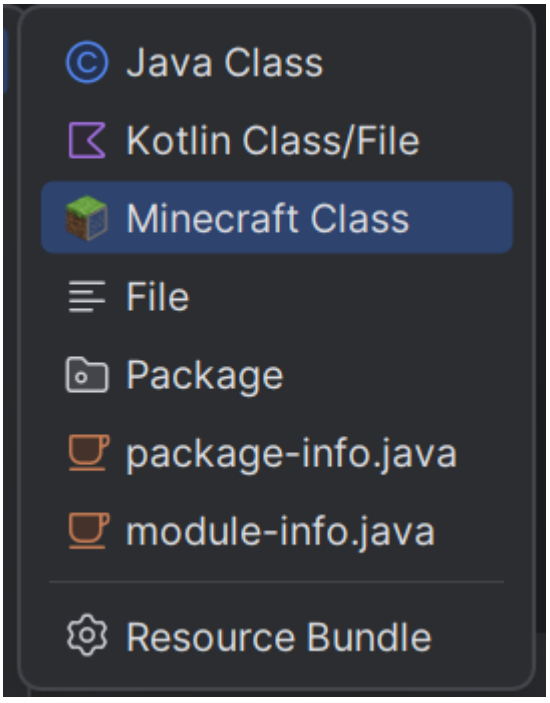
21 Nastavení hitboxu a kliknutí na blok

V této kapitole si ukážeme, jak nastavit hitbox bloku (tvar pevné části bloku). Dále si ukážeme, jak můžeme přidat funkčnost bloku a po kliknutí na blok něco spustit. S tím si ukážeme, jak vypsat zprávu do chatu a jak tuto zprávu formátovat (barva a styl textu). V příští lekci si ukážeme, jak na místě bloku vytvořit explozi.

Nastavení hitboxu bloku

V základu je hitbox ve tvaru krychle. Pokud chceme hitbox změnit, tak to musíme udělat v kódu.

Na to si musíme nejprve pro blok vytvořit vlastní třídu. Třídu si vytvoříme ve složce `src/main/java/com/example/block`. Tvorbu si můžeme díky rozšíření *Minecraft development* zjednodušit a místo vytvoření nové *Java* třídy můžeme zvolit vytvoření *Minecraft* třídy.



Dále ve výběru zvolíme *Block* a do textového pole napíšeme název bloku. Blok pojmenujeme `ExplodingBlock`. Automaticky se nám vytvoří základní struktura pro třídu bloku.

src/main/java/com/example/block/ExplodingBlock.java

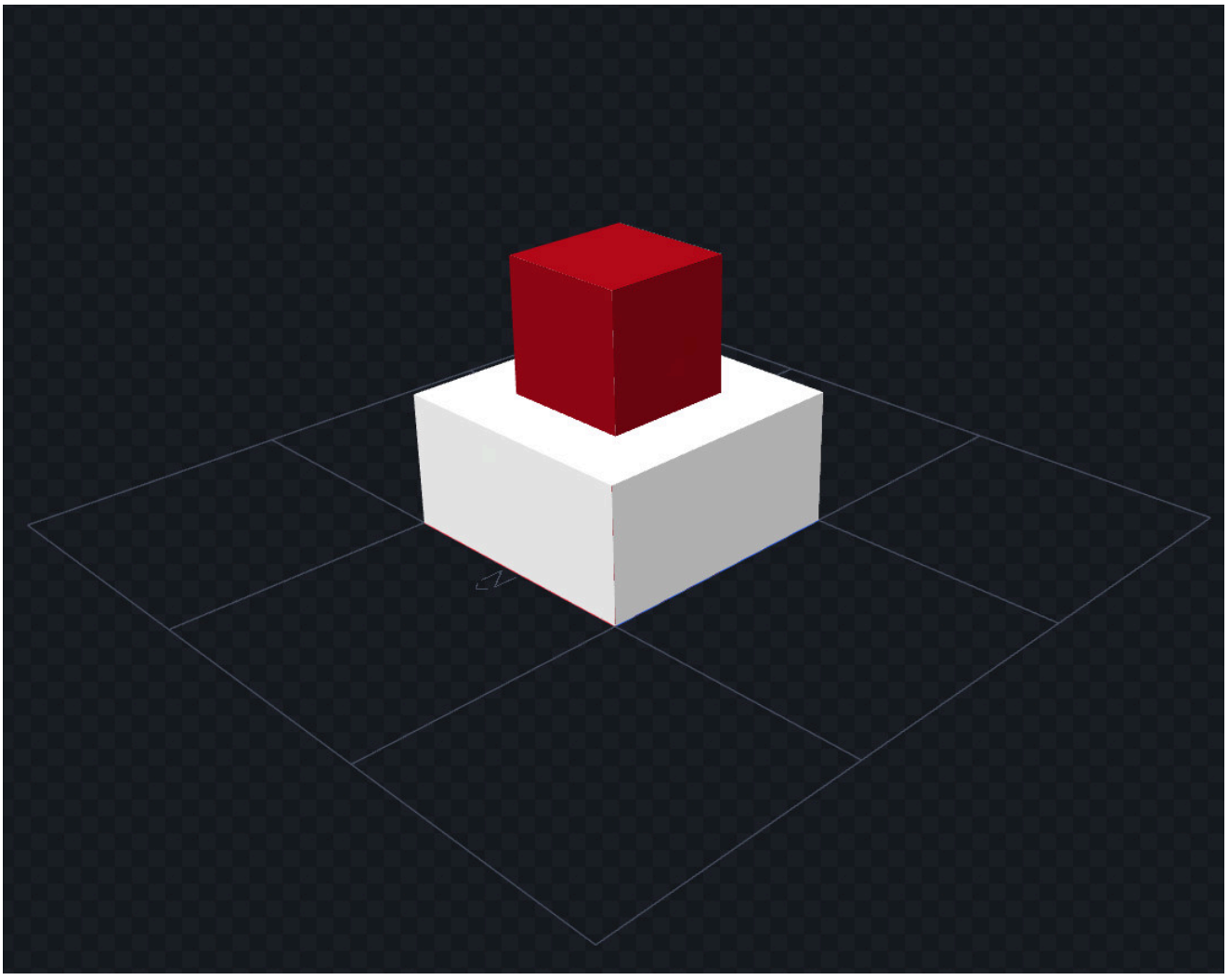
```
1  public class ExplodingBlock extends Block {
2      public ExplodingBlock(Settings settings) {
3          super(settings);
4      }
5  }
```

Na to abychom mohli změnit hitbox bloku, tak musíme přepsat metodu `getOutlineShape` ze třídy `Block`.

src/main/java/com/example/block/ExplodingBlock.java

```
1  @Override
2  protected VoxelShape getOutlineShape(BlockState state, BlockView world,
   BlockPos pos, ShapeContext context) {
3      return super.getOutlineShape(state, world, pos, context);
4  }
```

Tato metoda vrací `VoxelShape`, což je trojrozměrný objekt, který definujeme určitým stylem. V Minecraftu se tyto objekty vždy skládají z kvádrů. Funguje to vlastně podobně jako modelování v Blockbench, ale tady musíme objekt vytvořit pomocí kódu.



Blok, který jsem v předchozích lekcích tvořil se skládá ze dvou částí. Proto si nejprve vytvoříme tyto dvě části a poté je spojíme do jedné a to bude náš hitbox. Abychom nemuseli hitbox tvořit pokaždé, když si o něj hra zažádá, tak si vytvoříme neměnné proměnné, do kterých si tyto části i výsledný hitbox uložíme.

Více částí

> Pokud má váš blok více částí, tak buď můžete každou tuto část přidat i v kódu nebo můžete vytvořit nějaký zjednodušený tvar pro hitbox.

Samotný `VoxelShape` vytvoříme pomocí metody `createCuboidShape` ze třídy `Block`, kde jen zadáme souřadnice protějších rohů našeho kvádru.

Práci si můžeme ulehčit tím, že se podíváme na to, jak je model bloku definovaný a souřadnice si vezmeme přímo z definice modelu (`custom_model.json`):

src/main/resources/assets/custom-block/models/block/custom_model.json

```
"elements": [
      {
         "from": [0, 0, 0],
         "to": [16, 8, 16],
         ...
      },
      {
         "from": [4, 8, 4],
         "to": [12, 16, 12],
         ...
      }
   ]
```

Souřadnice (rozměry) jednotlivých částí bloku najdeme ve vlastnosti `elements`, kde máme pro každý kvádr vlastnosti `from` a `to`. Stačí nám tyto čísla přepsat do metody `createCuboidShape` a tím máme práci hotovou.

Vytvoříme si tedy podle modelu bloku dvě části:

src/main/java/com/example/block/ExplodingBlock.java

```
private static final VoxelShape BOTTOM_SHAPE = Block.createCuboidShape(0, 0, 0, 16, 8, 16);
private static final VoxelShape TOP_SHAPE = Block.createCuboidShape(4, 8, 4, 12, 16, 12);
```

Části jsem pojmenoval podle toho, kde se nachází `BOTTOM_SHAPE` - spodní část, `TOP_SHAPE` - vrchní část.

Následně nám jen stačí tyto části spojit do jedné a opět uložit do proměnné, kterou si můžeme nazvat například `SHAPE`. Části můžeme spojovat pomocí metody `union` ze třídy `VoxelShapes`. Tvarů můžeme spojit kolik chceme.

src/main/java/com/example/block/ExplodingBlock.java

```
1   private static final VoxelShape SHAPE = VoxelShapes.union(BOTTOM_SHAPE,
    TOP_SHAPE);
```

Nyní už stačí abychom jen přepsali metodu `getOutlineShape` tak, aby vždy vracela výsledný `SHAPE` a tím máme hitbox hotový.

src/main/java/com/example/block/ExplodingBlock.java

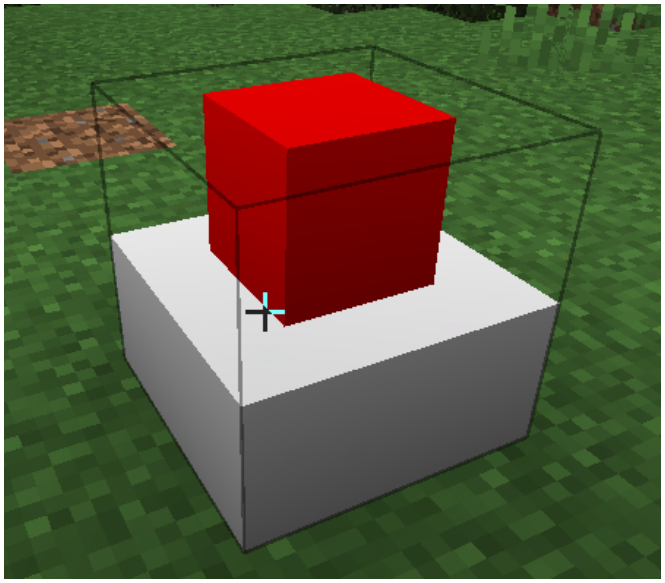
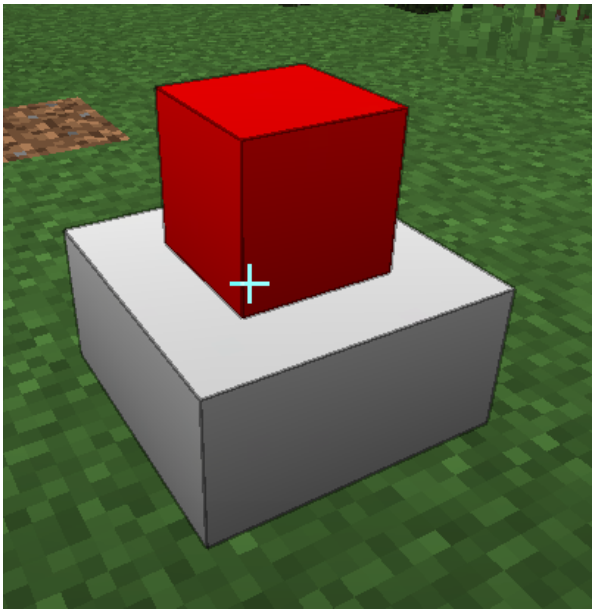
```
1   public class ExplodingBlock extends Block {
2       public ExplodingBlock(Settings settings) {
3           super(settings);
4       }
5
6       private static final VoxelShape BOTTOM_SHAPE =
    Block.createCuboidShape(0, 0, 0, 16, 8, 16);
7       private static final VoxelShape TOP_SHAPE = Block.createCuboidShape(4,
    8, 4, 12, 16, 12);
8       private static final VoxelShape SHAPE =
    VoxelShapes.union(BOTTOM_SHAPE, TOP_SHAPE);
9
10      @Override
11      protected VoxelShape getOutlineShape(BlockState state, BlockView
    world, BlockPos pos, ShapeContext context) {
12          return SHAPE;
13      }
14  }
```

Aby se nám hitbox zobrazil ve hře, tak musíme změnit, jakou třídu používáme při přidání bloku do hry. To změníme ve třídě `ModBlocks`, kde u bloku `CUSTOM_MODEL` změníme `new Block()` na `new ExplodingBlock()`.

src/main/java/com/example/block/ModBlocks.java

```
1   public static final Block CUSTOM_MODEL = register(
2           new ExplodingBlock(AbstractBlock.Settings.create()),
3           CUSTOM_MODEL_KEY,
4           true
5   );
```

Tím máme hotovo a blok by se nám měl ve hře zobrazovat s novým hitboxem.

Před	Po
	

Kliknutí na blok

Nyní si ukážeme, jak bloku přidat nějakou funkčnost. Konkrétně si ukážeme, jak do chatu vypsat zprávu po kliknutí pravým tlačítkem myši. Příště si k tomu přidáme vytvoření výbuchu. Na to využijeme metodu `onUse()`. Metodu přidáme tak, že začneme psát `onUse` a poté vybereme metodu z nápovědy. Automaticky se nám pak doplní následující kód:

src/main/java/com/example/block/ExplodingBlock.java

```
1   @Override
2   protected ActionResult onUse(BlockState state, World world, BlockPos pos,
    PlayerEntity player, BlockHitResult hit) {
3       return super.onUse(state, world, pos, player, hit);
4   }
```

Tato metoda se zavolá vždy, když hráč klikne pravým tlačítkem myši na blok.

Vypsání zprávy do chatu

Když chceme poslat zprávu hráči, tak k můžeme využít metodu `sendMessage` ze třídy `PlayerEntity` (parametr `player`). Tato metoda přijímá argument typu `Text`, což je něco jiného, než `string`, který známe jako textový řetězec. Třída `Text` v Minecraftu reprezentuje text, u kterého ale navíc můžeme měnit styl. Základní text vytvoříme pomocí metody `literal`, kam jako argument napíšeme textový řetězec (`string`).

Celý kód na vypsání zprávy do chatu tedy bude vypadat následovně:

src/main/java/com/example/block/ExplodingBlock.java

```java
@Override
protected ActionResult onUse(BlockState state, World world, BlockPos pos,
PlayerEntity player, BlockHitResult hit) {
    player.sendMessage(Text.literal("Ahoj"));
    return super.onUse(state, world, pos, player, hit);
}
```

Když si nyní spustíme hru a klikneme pravým tlačítkem myši na náš blok, tak narazíme na problém - zpráva *Ahoj* se nám do chatu vypíše dvakrát. To je způsobené tím, že pokud hrajeme singleplayer, tak máme na počítači spuštěný jak server, tak clienta (stará se o renderování). Oba se nám snaží poslat danou zprávu a proto se zpráva vypíše dvakrát. To můžeme vyřešit jednoduše tak, že v kódu zkontrolujeme, zda se akce spustila na serveru a pokud ano, tak vypíšeme zprávu.

Tuto informaci můžeme zjistit ze třídy `World`, která obsahuje informace o světě, kde máme informaci `isClient`. Můžeme tedy kód upravit tak, že pokud se daná metoda spouští na straně clienta, tak se žádná akce neprovede. To uděláme tak, že vrátíme hodnotu `ActionResult.PASS`.

src/main/java/com/example/block/ExplodingBlock.java

```java
@Override
protected ActionResult onUse(BlockState state, World world, BlockPos pos,
PlayerEntity player, BlockHitResult hit) {
    if (world.isClient) {
        return ActionResult.PASS;
    }

    player.sendMessage(Text.literal("Ahoj"));
    return super.onUse(state, world, pos, player, hit);
}
```

Formátování textu

Když už umíme vypsat zprávu, tak můžeme chtít, aby zpráva i nějak hezky vypadala. Můžeme si chtít třeba nastavit barvu textu, nějaké zvýraznění, podtržení atd.

Na formátování textu máme ve třídě `Text` metodu `formatted`, kde uvádíme formát, který chceme aplikovat. Formát se vybírá ze třídy `Formatting` a máme zde následující možnosti:

Barvy

- `BLACK`
- `DARK_BLUE`
- `DARK_GREEN`
- `DARK_AQUA`

- `DARK_RED`
- `DARK_PURPLE`
- `GOLD`
- `GRAY`
- `DARK_GRAY`
- `BLUE`
- `GREEN`
- `AQUA`
- `RED`
- `LIGHT_PURPLE`
- `YELLOW`
- `WHITE`

Vzhled

- `OBFUSCATED` - měnící se znaky
- `BOLD` - tučný text
- `STRIKETHROUGH` - přeškrtnutý text
- `UNDERLINE` - podtržený text
- `ITALIC` - kurzíva

Formátů můžeme aplikovat i více a to tak, že jednotlivé formáty oddělíme čárkou.

Příklady použití

- vypsání zlatého textu

```
1   player.sendMessage(Text.literal("Ahoj").formatted(Formatting.RED));
```

- vypsání zlatého textu, který je zároveň podtržený:

```
1   player.sendMessage(Text.literal("Ahoj").formatted(Formatting.RED,Formattin
```

Skládání textů

Pokud bychom chtěli napsat složitější text, kde by některá slova vypadala jinak, tak musíme pro každé takové slovo vytvořit zvlášť `Text` s aplikovaným formátem. Například bychom chtěli přivítat hráče a napsat mu zprávu *Ahoj, vítej!* a každou část mít jiným stylem. Text můžeme skládat pomocí metody `append`, která je součástí třídy `Text`. Do této metody poté jako argument uvedeme text (opět typu `Text`), který chceme přidat.

Příklad použití:

```

```

```
player.sendMessage(Text.literal("Ahoj
").formatted(Formatting.RED,Formatting.UNDERLINE)
    .append(Text.literal(", vítej!").formatted(Formatting.GOLD)));
```

Celý kód s textem bude vypadat následovně:

src/main/java/com/example/block/ExplodingBlock.java

```
@Override
protected ActionResult onUse(BlockState state, World world, BlockPos pos,
PlayerEntity player, BlockHitResult hit) {
    if (world.isClient) {
        return ActionResult.PASS;
    }

    player.sendMessage(Text.literal("Ahoj").formatted(Formatting.RED,
Formatting.UNDERLINE)
            .append(Text.literal(", vítej!").formatted(Formatting.GOLD)));

    return super.onUse(state, world, pos, player, hit);
}
```

Pro zajímavost

Zkuste vymyslet, jak byste napsali duhový text (text, kde každé písmeno má jinou barvu).

Pokud si myslíte, že musíme každé písmeno napsat zvlášť a nastavit mu zvlášť barvu, tak je to správně. Kód by vypadal následovně:

```
player.sendMessage(Text.literal("D").formatted(Formatting.RED)
    .append(Text.literal("u").formatted(Formatting.GOLD))
    .append(Text.literal("h").formatted(Formatting.YELLOW))
    .append(Text.literal("o").formatted(Formatting.GREEN))
    .append(Text.literal("v").formatted(Formatting.BLUE))
    .append(Text.literal("ý ").formatted(Formatting.DARK_PURPLE))
    .append(Text.literal("t").formatted(Formatting.LIGHT_PURPLE))
    .append(Text.literal("e").formatted(Formatting.RED))
    .append(Text.literal("x").formatted(Formatting.GOLD))
    .append(Text.literal("t").formatted(Formatting.YELLOW)));
```