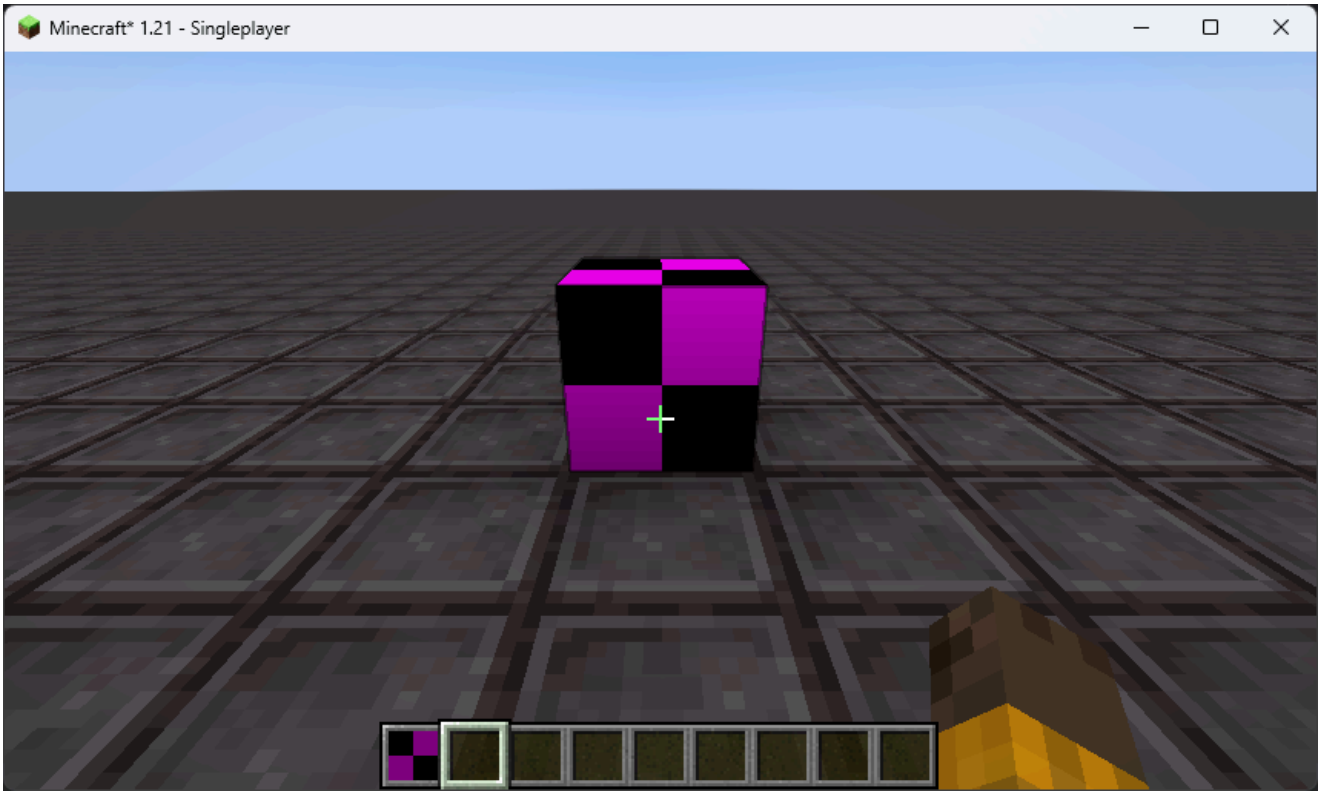
14 Přidání bloku do inventáře, textura bloku

Z minulé lekce nám ještě zbývá poslední krok, aby se nám blok zobrazil ve hře. Ve třídě `CustomBlock` musíme do metody `onInitialize` přidat počáteční nastavení třídy `ModBlocks`, aby se nám přidaly všechny bloky z této třídy (zatím máme jen jeden). To uděláme pomocí metody `initialize`, kterou jsme si do třídy `ModBlocks` přidávali.

CustomBlock.java

```
@Override
public void onInitialize() {
    // This code runs as soon as Minecraft is in a mod-load-ready state.
    // However, some things (like resources) may still be uninitialized.
// Proceed with mild caution.
    LOGGER.info("Hello Fabric world!");
    ModBlocks.initialize();
}
```

Když nyní hru spustíme, tak bychom měli mít blok přidaný do hry. Můžeme jej ale získat jen pomocí příkazu `/give @s custom-block:condensed_dirt`. Blok také zatím nemá žádnou texturu a vypadá následovně:



Přidání do kreativního inventáře

Blok se do inventáře přidává stejným způsobem jako obyčejný item, protože přidáváme item, který reprezentuje daný blok. Ukážeme si, jak přidat item na konkrétní místo v záložce. Tím,

že vytváříme variantu hlíny, tak jej můžeme umístit za hlínu.

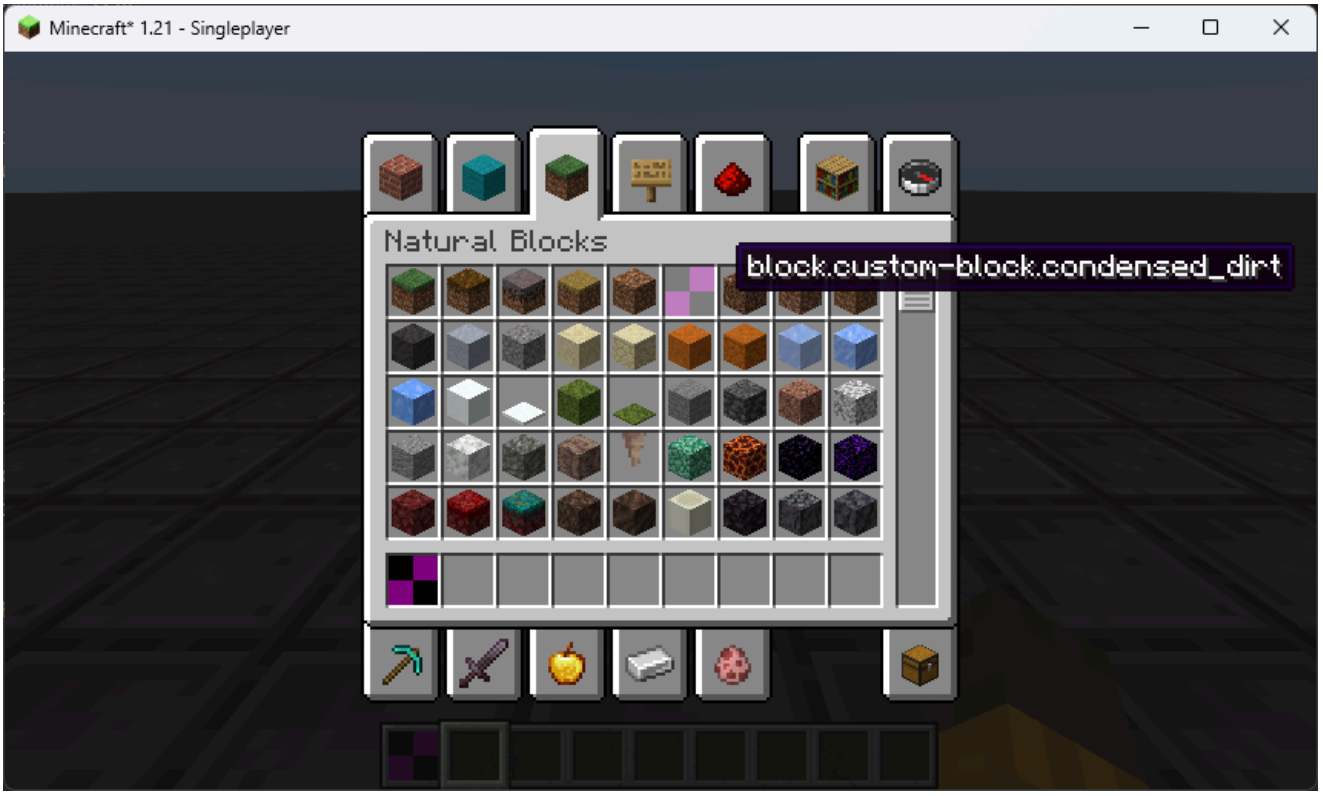
Kód na přidání itemu do kreativního inventáře píšeme do metody `initialize` ve třídě `ModBlocks`.

Item přidáváme pomocí metody `modifyEntriesEvent` ze třídy `ItemGroupEvents`, kde jako argument uvádíme, do jaké skupiny chceme item přidat. V našem případě to bude skupina přírodních bloků, protože tam se nachází hlína. Skupinu itemů získáme ze třídy `ItemGroups` pomocí `ItemGroups.NATURAL`.

Když máme získanou skupinu, do které chceme item přidat, tak použijeme metodu `register` pro úpravu této skupiny. Jako argument uvádíme tzv. lambda funkci (nemusíte řešit, co přesně to je). Lambda funkce je ve tvaru `parametr -> kód`. V našem případě je parametr skupina itemů (můžeme nazvat `group`). V kódu funkce můžeme do této skupiny přidat item pomocí metody `addAfter` - metoda přidává item za nějaký již existující item. Jako první argument uvádíme item, **za který** chceme něco přidat - v našem případě `Items.DIRT`. Ve třídě `Items` najdete všechny itemy. Jako druhý argument uvádíme item, který chceme přidat - v našem případě je to item k bloku. Ten získáme pomocí metody `asItem` ze třídy `Block` - `CONDENSED_DIRT.asItem()`.

ModBlocks.java

```
1  public static void initialize() {
2  ItemGroupEvents.modifyEntriesEvent(ItemGroups.NATURAL)
3          .register(group -> {
4              group.addAfter(Items.DIRT, CONDENSED_DIRT.asItem());
5          });
6  }
```



Přidání textury

Ukázkovou texturu můžete stáhnout na tomto odkazu.

Na to, aby se nám textura zobrazovala ve hře, potřebujeme udělat několik věcí.

1. přidat obrázek s texturou
2. přidat model bloku
3. přidat stav bloku

Textury se stejně jako u itemů přidávají do složky `resources`. Ve složce `resources/assets/custom-block` si vytvoříme novou složku `textures/block`.

> **Vytvoření více složek**
>
> Při vytváření nové složky je možné vytvořit více složek v sobě. To uděláte tak, že jednotlivé složky oddělíte znakem `/`. Například tedy `textures/block` vám vytvoří složku `textures` a v ní složku `block`.

Do této nově vytvořené složky přesuneme staženou texturu. Poté musíme texturu přejmenovat stejně jako se jmenuje náš blok. Soubor přejmenujeme kliknutím pravým tlačítkem myši a zvolíme *Refactor* > *Rename* a zadáme nový název souboru `condensed_dirt.png`.

Přidání modelu

Dále potřebujeme přidat model bloku. To je soubor, který popisuje, jaký tvar a texturu má blok mít. My použijeme úplně nejjednodušší model a to je obyčejná krychle, která má na všech stranách stejnou texturu.

Modely se opět přidávají do složky `resources`. Ve složce `resources/assets/custom-block` si vytvoříme novou složku `models/block`.

V této složce si dále vytvoříme nový soubor s názvem bloku `condensed_dirt.json`.

Opět se jedná o `json` soubor, kde budeme mít definovaný model. Stejně jako u itemu, tak i model bloku může vycházet z jiného existujícího modelu. V našem případě použijeme model `minecraft:block/cube_all`, který reprezentuje výše popsanou krychli. Dále přidáme texturu, která se aplikuje na všechny strany (vlastnost `all`) `custom-block:block/condensed_dirt`. Celý soubor bude vypadat následovně:

models/block/condensed_dirt.json

```
1  {
2    "parent": "minecraft:block/cube_all",
```

```
  "textures": {
    "all": "custom-block:block/condensed_dirt"
  }
}
```

Stav bloku

Jako poslední musíme přidat stav bloku. Tyto soubory reprezentují, jaké jednotlivé modely a textury může blok mít. Dále mezi těmito stavy můžeme přepínat pomocí kódu. I když se náš blok nijak nemění, tak musíme přidat alespoň jeden základní stav.

Stavy se nachází ve složce `resources`. Ve složce `resources/assets/custom-block` si vytvoříme novou složku `blockstates`.

V této složce si dále vytvoříme nový soubor s názvem bloku `condensed_dirt.json`.

Samotný soubor obsahuje vlastnost `variants`, kde je seznam všech možných modelů bloku. Každá varianta musím být pojmenovaná. Základní varianta žádné jméno nemá a použijeme tedy jen prázdné uvozovky. Pro každou variantu dále uvádíme, jaký model se má použít - vlastnost `model`. V našem případě je to model `custom-block:block/condensed_dirt`.

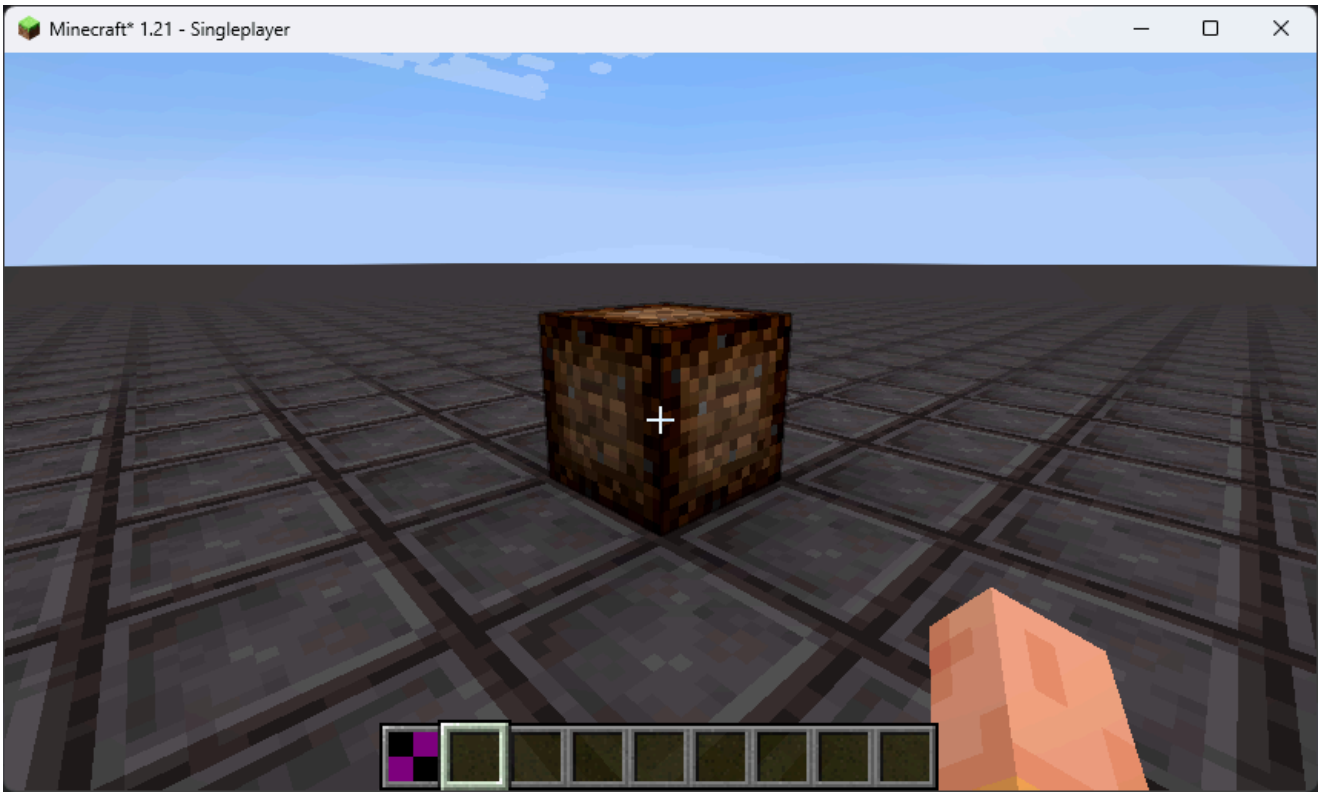
Celý soubor bude vypadat následovně:

blockstates/condensed_dirt.json

```
{
  "variants": {
    "": {
      "model": "custom-block:block/condensed_dirt"
    }
  }
}
```

Zobrazení ve hře

Nyní bychom měli blok vidět ve hře. Textura bude aplikovaná zatím jen na blok a item bude mít pořád neznámou texturu. Je to proto, že pro item jsme zatím žádnou texturu nepřidali. To si ukážeme v následující lekci.


Minecraft* 1.21 - Singleplayer