**Personal report**

**-Name of the game-**

**Patrick Engesser INF4**

**Group members:** Florian Bertele, Patrick Beinlich, Lars Kalkuhl, Philipp Grüger

**What did I do and how it works**

I was responsible for the item classes as well as the inventory and its functionality.

**Items**

For the item classes I decided to create an abstract class called “Item”, which contains the core elements and functions that every item should have:

* Id
* Mining-level
* Getter and setter for those attributes

Three abstract classes inherit from this class and are variations that an item can have. These classes have more specified attributes and functions:

* Resource
  + Doesn’t need extra things
* Tool
  + Attack power
  + Getter and setter
* Construct
  + Array list of items
  + Clone method
  + Getter and setter

I created the following resource classes:

Block-ID Block Mining-level Block-ID Block Mining-level

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | Air | 0 | 8 | Diamond | 4 |
| 1 | Dirt | 1 | 9 | Leaves | 1 |
| 2 | Stone | 2 | 10 | Water | 0 |
| 3 | Wood | 5 | 11 | Sand | 1 |
| 4 | Iron-ore | 2 | 20 | Plant | 1 |
| 5 | Cooper-ore | 2 | 26 | Water surface | 0 |
| 6 | Silver-ore | 2 |  |  |  |
| 7 | Gold-ore | 3 |  |  |  |

The following tool classes:

Tool-ID Tool Mining-level Attack power

|  |  |  |  |
| --- | --- | --- | --- |
| 12 | Pickaxe | 2 | 10 |
| 13 | Stronger pickaxe | 3 | 10 |
| 14 | Strongest pickaxe | 4 | 10 |
| 15 | Axe | 5 | 20 |
| 16 | Sword | 1 | 20 |

Ant the following construct classes:

Construct-ID Construct Mining-level Components

|  |  |  |  |
| --- | --- | --- | --- |
| 21 | Char | 1 | 1 x Wood |
| 22 | Desk | 1 | 2 x Wood |
| 23 | Door | 1 | 1 x Wood |
| 24 | Campfire | 1 | 1 x Wood |
| 25 | Furnace | 1 | 1x Wood  3 x Stone |

The mining-level can be described as follows:

* 0 -> not mineable - 3 -> mineable with a stronger pickaxe
* 1 -> mineable without a tool - 4 -> mineable with the strongest pickaxe
* 2 -> mineable with a pickaxe - 5 -> mineable with an axe

All classes set their attributes within the constructor and the construct classes all overwrite the clone method that they inherit from the abstract construct class. The constructs hold an array list of resources, because of that, they can’t be simply cloned. Therefor they create a new construct within the clone method. There is a method in the control class of the game, that checks the mining level of a block and the mining level of the current item in use by the player. If the mining level is high enough, the player can mine a block from the world. If a construct is mined, the components go back into the inventory.

**Inventory**

The inventory consists of an array of item stacks. An item stack holds an array of items. That allows item stacking within the inventory and enables us to easily show the current number of items in a slot. At the start of a game the inventory holds a pickaxe, an axe and a sword. If an item gets added to the inventory, a method from the inventory checks if there is an existing stack of that item. If not, a new stack is generated. If a stack does exist, the add method from the item stack class gets called and the item gets added to the array list of the item stack and the item count is increased. When an item gets removed from the inventory, the inventory calls the remove method from the item stack, which returns the item and decreases the item count. To solve an issue regarding null pointer exceptions, that occur if there are no objects in parts of the inventory, the inventory is filled with air blocks when first generated. The same happens if an item stack gets completely removed from the inventory.

**Construct menu**

The construct menu holds all the implemented constructs. When a new world is generated, the menu automatically generates one instance per construct and adds them to the menu. If the player now wants to create a construct, and has enough resources for that, the existing construct is cloned. The check, if there are enough resources for a construct, is done in the control class by a method which goes through the inventory and checks if the corresponding item stacks are there and if the number of resources in these stacks is high enough.

Sadly, at the end of the project we run out of time and couldn’t put the graphics for the constructs into the game. However, the infrastructure explained above is in place, so the effort to fully implement the construct menu graphically isn’t too high.