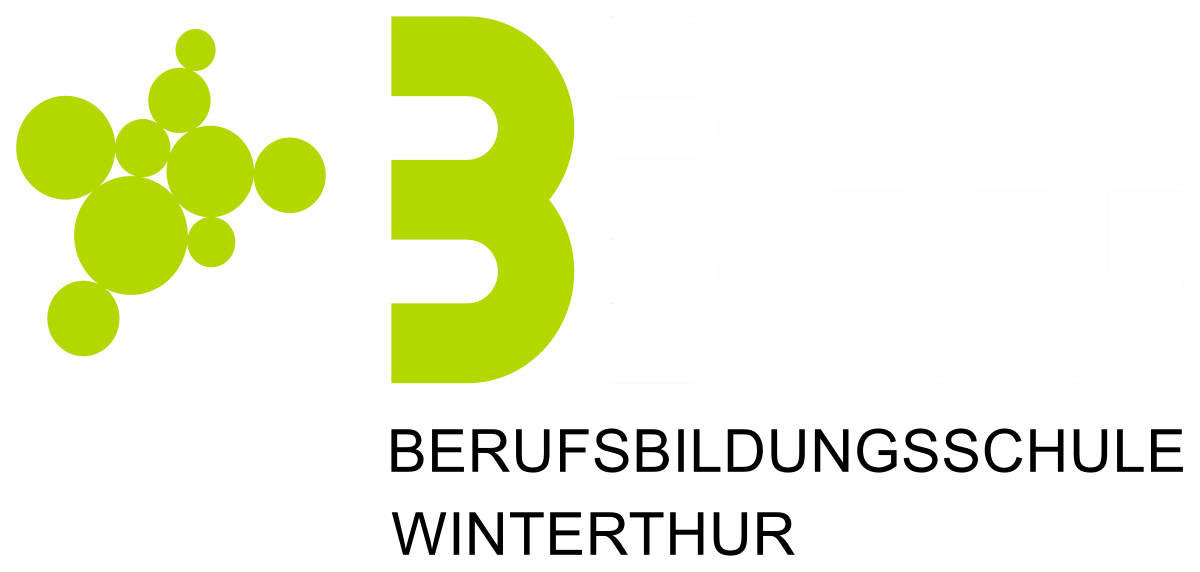
Zork Dokumentation

Bossi Matteo und Rädler Julien

https://github.com/bbw19/zorg.git

Inhalt

[Konzept 2](#_Toc54363716)

[Aufbau 3](#_Toc54363717)

[Spielregeln 4](#_Toc54363718)

[Gewonnen 5](#_Toc54363719)

[Back 5](#_Toc54363720)

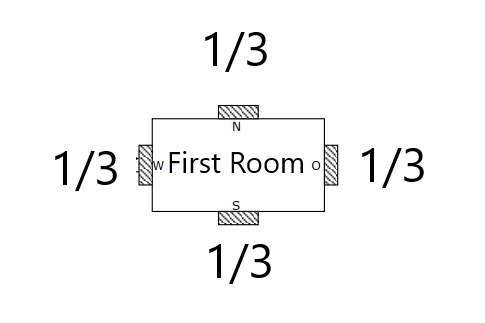
[Gegenstände 5](#_Toc54363721)

[Map 5](#_Toc54363722)

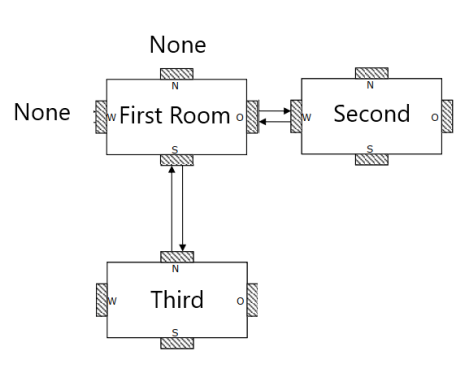
[Gegenstand im Raum 5](#_Toc54363723)

# Konzept

Zuerst wird ein erster Raum in der Mitte auf der Position (x : 0, y : 0) erzeugt.



Um diesen Raum werden dann an den Vier Positionen W, N, O, S mit einer gewissen Wahrscheinlichkeit weitere Räume erzeugt.



Diese Räume werden in einer Hash-Map gespeichert, wobei die Koordinaten der Key sind und das Raum-Objekt das Value. Sobald der erste Raum alle Vier Optionen durchlaufen hat, geht es weiter zum zweiten Raum, wo dasselbe noch mal ausgeführt wird. Es wird aber überprüft ob auf den vier Positionen um den Raum bei (x = 0, y : 0) währen das (x : -1, y : 0), (x : , y : 1), (x : 1, y : 0), (x = 0, y = -1) schon Räume sind. Weiter ob das Limit an Räumen erreicht ist. Die Koordinaten werden in als Vektor2-Klasse gespeichert.

# Aufbau

|  |  |
| --- | --- |
|  | ICommandRunner ist ein Interface, das von den Commands implementiert wird.  Color ist ein Enum das die ANSI-Codes für die Farben im Terminal enthält.  In der Command-Klasse werden die Commands registriert.  In der Room-Klasse findet die random romm generation statt.  In Map wird die Matrix und der Rest der Terminalausgabe für die Map-Funkrion erzeugt.  Parser liest die Terminal-Commands und called den entsprechenden Command.  Room hat eine Position und weiss über seine eigenen Koordinaten die Position seiner Nachbar-Räume.  Vektor2 ist die Klasse, die die Koordinaten der Räume enthält und erfüllt die selbe Rolle wie die Java Point-Klasse. |

# Spielregeln

## Gewonnen

Als erstes muss man den Schlüssel mit dem «Pickup» Befehl aufheben. Der Schlüssel ist in einem Raum der zufällig ausgewählt wurde. Dort wurde die «HasKey» Variable auf true gesetzt. Danach muss man in den End-Raum gehen. Die Raum-Klasse hat ein Bolean «IsFinishRoom» der anzeigt, ob es der End-Raum ist. Wenn der End-Raum erreicht ist, kann man dort mit dem Use-Key Befehl das Spiel gewinnen.

## Back

Im Go-Command wird jedes Mal, wenn der Go-Command aufgerufen wird, die entsprechende Richtung (north, east, south, west) in die ArrayList «actions» geschrieben. Im Back-Command wird die Richtung dann invertiert und zur zweitletzten ArrayList-Position zurückgekehrt. Die letzte Position wird schliesslich gelöscht.

## Gegenstände

Es existiert ein Key, dieser wird benötigt, um das Spiel im richtigen Raum zu gewinnen. Er wird bei der Erstellung des Spielfelds zufällig in einem Raum platziert. Dazu wird die «hasKey» Variable auf true gesetzt. Mit dem Pickup-Command wird die «hasKey» Variable des Raumes auf false gesetzt und die Variable playerHasKey aus der Game-Klasse auf true.

## Map

Es kann eine Karte mit dem Befehl map ausgegeben werden. Dazu wird eine Matrix auf der Basis der MapSize generiert. Die Quadratwurzel der MapSize bestimmt wie viele Räume nebeneinander und übereinander stehen. Es können nur Quadrat zahlen verwendet werden. Dann wird überprüft, ob auf dieser Koordinate ein Raum existiert. Aus der ArrayList mit den Räumen. Wenn ein Raum existiert, wird dieser mit seinen Attributen gerändert (hat key, ist end Raum). Wenn nicht wird ein Platzhalter ausgegeben.

## Gegenstand im Raum

Räume können einen Key haben oder nicht. Das wird über einen Boolean gestäuert.