World of Deez

Pawel Makles

(K21002534)

1 User Level Description

This game is largely open world but has a main storyline which the player can choose to follow to complete the game, as well as other side-quests which can be done alongside or after the story.

In this game, you are one of millions of citizens in an isolated city home to a Beastman[1] society, the city being loosely based off Animacity[2] from the series Brand New Animal.[4] The plot revolves around Sylvasta who is deeply intertwined with the city and owns the city's medical centre.

There have been reports that their research has been highly unethical and leaks have been coming out from former employees which have been riling up protests throughout the city. Your goal is to figure out what's really happening and to put a stop to it.

I ended up not adding any information about the main character as to add to the open world, 'your adventure', feeling. The story is also loosely based off Utopia,[5] in which a group of people find out the truth behind a manuscript and find themselves in the middle of a conspiracy.[10]

2 Implementation

The project is split up into several packages, most of which work independently of each other, in general, the `Game` uses the `commands` package (providing actions the player can perform) along with the `world` (providing the things that the player can interact with in the game) package to run the game. They are laid out as follows:

2.1 Commands

This package provides the classes required to parse and run commands, it includes the `CommandManager` which registers objects of type `Command`. More about parsing commands is in the challenge task section.

This package also includes a subpackage called `core` containing all of the 'core' commands required to run any game world, this includes things such as picking items up, quiting the game, and so forth.

2.2 Entities

This package provides the basic tools required to create simple and complex entities within the world, more about entity detail is discussed later.

It also contains a subpackage `actions` which contains common interfaces which entities can implement to allow the ways that they can be interacted with to be quite modular.

2.3 Content / Campaign

The `content.campaign` package contains a lot of custom content derived from the base game used to construct the story and the story world.

2.4 World

This package has the basic building blocks for creating worlds including the `World` and `Room` classes which can be easily extended for more functionality. The only limitations are that traveling between rooms must be done in a direction specified in the `Direction` enum and that the `Location` of any entity must be either a room, inventory or neither.

2.5 Util

This package contains a bunch of small utility classes, including:

- **BlueJ.java**:[7] Contains improved methods on detecting whether the application is currently running in BlueJ. (taken from my previous coursework assignment)

- **Localisation.java**: A pretty straightforward implementation of a localisation engine. Simply maps (period-separated) keys to their respective (nested) keys in a language file.

- **Search.java**: Methods for searching through various data structures in the game.

- **Tree.java**: A very simple implementation of a tree with basic traversal methods.

2.6 Dialogue & IO & UI & Events

These are all discussed later on in the challenge task section.

3 Base Tasks

This section describes how I implemented the basic task requirements.

- *"The game has several locations the player can walk through."*

  I began by first designing the world map (see Figure 1), using Excalidraw,[6] then implemented each location as a `Room`.

  To build the world, I made a `World` class to house all the locations which exist in the game world and then extended this using the `CampaignWorld` class which builds the story world, creates rooms and registers events.

  I ended up adding 10 locations into my game:

  - **City Centre**: Centre of the city connecting major areas with the coast.
  - **Apartments**: This is the player's residence.
  - **Street**: This is the main city street connecting important buildings.
  - **Shop**: The local city shop where the player frequents to get necessary items.
  - **Back Alley**: This is where the player can start the main story mission.
  - **Coastline**: There are two coasts, one on the city side and one on the mainland.
  - **Forest**: The forest grants access to the Worm Hole and to other side-quests.
  - **Worm Hole**: For challenge task 3.

  The layout is heavily inspired by Animacity, although since there's no available map of the actual city, I made my own interpretation based off various pieces of art, (see Figure 2). I used an existing real life location to determine the size of the river.[8]

- *"There are items in some rooms that may or may not be picked up by players."*

  To achieve this, I considered all entities to be items which may or may not be picked up by other entities, each entity has its own `Inventory` which is in effect a list of other entities which it is holding.

- *"Each item has a weight and the player can only carry items up to a certain weight."*

  To do this, I added a new private field `weight` of type double to the `Entity` class, which is used to store the entity's weight. It is a double as I wanted access to fractional weights (say 0.01kg) and I wanted to have access to `Stream::mapToDouble` for summations.

  For the second part, I made it so each `Inventory` has a maximum weight it can store, which by default is set to 0 as each entity has an inventory but may not necessarily have the ability to store anything.

  When putting anything in an inventory, we check that the following is satisfied:

  $$\text{currentWeight} + \text{itemWeight} \leq \text{maxWeight}$$

  To determine the weight of common items, I referred to a document I found online published by the City of York Council.[9]

- *"Player can win."*

  The player may win by completing the main story mission (detailed in the walkthrough) which sets a flag that the game has been completed, the player may choose to keep playing in the open world or run *win* to end the game.

- *"There is a command *back* which takes you back to the last room."*

  I added two new private fields to the player entity, which were `previousRooms` and `retreatingDirection`, these are used to store the path back through the room and the direction which we need to go to get back there respectively. The direction is stored in order to run a check whether the player can actually go back in the direction they intend to, to verify this, the 'retreating direction' is used to call the method `canLeave` on the current room the player is in.

- *"Add at least four new commands."*

  I added several additional commands which are listed below:

  - *bag* : Allows the player to look at their or another entity's inventory.
  - *drop* : Drop any specified item from the player's inventory into current room.
  - *give* : Give a specified item to another entity, we ensure that the entity implements `IGiveable` and give the item to the entity using `IGiveable::give`.
  - *pet* : Pet a specified entity, has to implement `IPettable`, we use `IPettable::pet`.
  - *take* : Take any specified item from another entity's inventory.
  - *talk* : Initiate a conversation with an entity, must implement `ITalkwith`, we call `ITalkwith::talk` to start the conversation.
  - *use* : 'Use' an entity, we call `IUseable::use` to use the entity. (implemented by entity)
  - *where am i* : Tells the game to print out room information again, this is done by emitting `EventEntityEnteredRoom` again.

  See Figure 5 for an example of the help menu.

4   Challenge Tasks

This section describes how I implemented the challenge tasks and what I did in addition.

4.1   Required Tasks

- *"Add characters to your game."*

  I ended up adding 8 characters to my game, 3 of which can move on their own.

  - **Static NPCs**: There are several static NPCs which the player can interact with, talk with or complete missions for. Including the receptionist, city NPC, the old man, the security guard, the shopkeeper and Marie who is part of the main story.
  - **Dynamic NPCs**: There are 2 NPCs which appear or disappear depending on the state of the game, this includes the protestors and security guard.
  - **Moving NPCs**: There is a single NPC, the cat, which randomly follows a fixed path to move around the map. This is done by listening to `EventTick` from world events and then randomly deciding to move.

  Note on character creation: I decided to stick to mostly generic characters in the game, mainly since the people you encounter have not as much impact on the story, with the exception of Marie,[3] who I took straight from BNA,[4] see Figure 3, she is a sly and cunning character which I think perfectly fits the role I need.

- *"Extend the parser."*

  I entirely replaced how the parser worked, I began by implementing a basic model of `Command`, it took a few iterations but I settled on providing Regex `Pattern`s.

  This approach has several benefits:

  - Powerful Regex at low performance cost due to the low number of commands.
  - Ability to have named capture groups which are then interpreted as arguments.

  For each known `Command`, I would create a `Matcher` for each `Pattern`, execute it against the arbitrary command from the user and then pass it into a wrapper class `Arguments` which lets me safely pull out named groups, directions or any other argument type I need.

- *"Add a magic transporter room."*

  To do this, I added a `RoomWormHole` which I made implement `EventEntityEnteredRoom` to listen for when any entities entered the room, as soon as one is detected, a short animation is played and a room is selected at random to teleport the user to. Allowing the user to go to **any room** may interfere with my story so I chose to only spawn the user at any outside areas of the map.

4.2   World Event System

"Add a system for managing world events."

I began by creating a basic `Event` class which can be extended to hold any arbitrary data, it also provides basic functionality to prevent event propagation (see Figure 8).

I created an `EventSystem` class which would manage incoming and outgoing events, I decided that it should have the ability to handle any event and allow anything to create listeners for any object that extends Event.

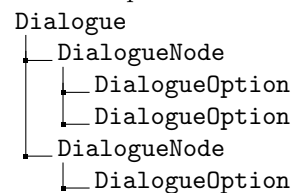To map these, I had a `HashMap` which mapped unique `Class<? extends Event>` to linked (as to keep order of registration) hash sets of `IEventListener<? extends Event>`. The interface simply provides a single method, `onEvent(Event e)`, which is called when the event is triggered.

Since Java has type erasure, it means my generic type annotations are gone once the program is compiled, so I can't do things such as object is 'instanceof' class. As such, I decided to ignore cast warnings and entirely rely on compiler type checking to verify that the objects provided are correct, I achieved this by adding a generic type argument to the add listener method and making the compiler check that the target class and event listener match, see Figure 15.

4.3   Dialogue

"Give NPCs interactive dynamic dialogue."

To implement dialogue, I made a simple data structure implemented as follows:

```
Dialogue
├─ DialogueNode
│   ├─ DialogueOption
│   └─ DialogueOption
└─ DialogueNode
    └─ DialogueOption
```

The `Dialogue` houses several nodes each of which have options which lead to other nodes. With this system, I can add dynamic conversations, such as with the shopkeeper, by extending the `DialogueNode` class and providing my own dynamically generated description and options.

For simpler tasks, such as changing game state through conversation, I can create a new `DialogueOption` with an `IDialogueHandler` which provides a method that the dialogue system calls to determine whether anything else needs to happen and what the next node is.

4.4 Terminal Emulator
"Implement a terminal emulator."

I began by creating a new `IOSystem` interface which provides common I/O methods, such as `print` , `println` and `readLine` . I then made all game output come through this interface and created two new classes that implemented it, `StandardIO` as a fallback to stdout and the `TerminalEmulator` which creates a new window.

To render the window I have a class extending `JFrame` which shows my `JTerminalView` . To render the text I override into `paintComponent` and draw my own content using the provided `Graphics` context.[19] (example: Figure 6)

The font used is VT323 Regular.[12]

4.4.1 Ansi Escape Codes
One of the main reasons I built this was for greater flexibility, so naturally I added support for Ansi escape codes. I matched the escape codes in the `TextBuffer` and adjusted the buffer accordingly. (Figure 17 for the implementation)

4.4.2 Emoji / Image Support
I also added a way to load and display small images like emojis (see Figure 4), although I had some difficulties with Unicode being mangled when working with BlueJ.[11]

4.4.3 EventDraw and the Map
Since I already had a way to load images, I also provided a way for anything to draw to the terminal. I added an `EventDraw` (Figure 14) which is emitted whenever the terminal view paints and used it to draw a map (see Figure 7).

5 Code Quality
5.1 Coupling
An example of loose coupling in my project is the `EventSystem` , it is completely independent from the rest of the project and can be used with anything (see Figure 12), as such I am using it with both the `World` and `TerminalEmulator` for different purposes to provide different sets of events. Another example of loose coupling is the `Dialogue<T>` system, it accepts any kind of key through the generic `T` so it can be used with practically anything and if needed, the type can be constricted (example: Figure 13). The only dependency it has is the `IOSystem` interface.

5.2 Cohesion
One example of high cohesion is the `IOSystem` interface (see Figure 9) and the classes which implement it. It is used throughtout the project to refer to an arbitrary input or output, as such, the first implementation I made was StandardIO (see Figure 10) which passes these methods through to `System.out/in` . One benefit is that I can easily slot in additional processing layers, such as `LocalisedIO` (see Figure 11) which can take in data, process it and pass it through to an arbitrary `IOSystem` .

5.3 Responsibility-driven design
I considered responsibility-driven design when first building the structure of my project, for example, the `World` class is responsible entirely for keeping track of rooms and entities and nothing else, to add functionality to it you must extend it as I have done with the `CampaignWorld` . As another example, each `Entity` just stores any information directly relating to the entity such as a name and its location, to implement additional actions such as 'using the entity', we must extend `Entity` and then implement `IUseable` .

5.4 Maintainability
One example of where I considered maintainability is the `Inventory` system, the logic is entirely self-contained and quite simple to understand. Fields such as `items` , `maxWeight` are both private and inaccessible outside of the inventory class, as such I can guarantee that the inventory always stays consistent and hence easily maintainable, these values may only be modified by methods with distinct tasks and appropriate validity checks (examples: Figure 16).

6 Walkthrough
This guide will skip over some parts of the plot, it serves as a way to just get to the end, I would recommend playing through first and then falling back on this if you are stuck.

You wake up in your home, *go down* , then *east* then *north* . Here, you must *talk to Marie* , go through the prompts $(1 \rightarrow 1 \rightarrow 2 \rightarrow 1)$ to progress the story.

Now, *go south* , *north west* , *north* and *talk with shopkeeper* , buy the communicator device and leave. Now *go south* , *east* , *north* and *talk to Marie* again. Continue by going *south* , *north west* , *west* . Sit down by *use couch* then *use comms* and go through the prompts, afterwards *go down* . Here, you must pick up documents 2, 4 and 5 by using *take doc2* , ...

Now, *go up* , *east* , *south* , *up* to get back to your home and *use the laptop* . Go through the prompts $(1 \rightarrow 4 \rightarrow 2)$. At this point, you can use the command *win* . If you want to do the side-quest, you'll have to figure it out yourself :-)

6.1 Access to teleporter room
To get to the teleporter room, you must go to the shopkeeper and buy the 'speed boat key', from there you can go to the south of the city and use the boat to get to the mainland. Going south to the forest then east to get to the teleporter room.

7 Known Issues
- Anything starting with 'b' will show inventory.

- NPC dialogue does not vary in conclusion.

- Can't include Unicode in Java source.

References

[1] Brand New Animal Wiki. Beastman
https://brand-new-animal.fandom.com/wiki/Beastman

[2] Brand New Animal Wiki. Animacity
https://brand-new-animal.fandom.com/wiki/Anima_City

[3] Brand New Animal Wiki. Marie Itami
https://brand-new-animal.fandom.com/wiki/Marie_Itami

[4] IMDb. BNA (TV Mini Series 2020)
https://www.imdb.com/title/tt12013558/

[5] IMDb. Utopia (TV Series 2013-2014)
https://www.imdb.com/title/tt2384811/

[6] Excalidraw.
https://excalidraw.com/

[7] GitHub. maven-bluej / BlueJ.java
https://github.com/KCLOSS/maven-bluej/blob/master/BlueJ.java

[8] Google Maps. Jezioro Świerklaniec, Poland
https://www.google.com/maps/@50.4293559,18.9742453,16.12z

[9] PDF. Set of average weights for furniture, appliances and other items https://democracy.york.gov.uk/documents/s2116/Annex%20C%20REcycling%20Report%20frnweights2005.pdf

[10] YouTube. The best (and worst) show you haven't seen
https://youtu.be/PFx2QM0Z8Qo

[11] GitHub. BlueJ Bug Reproductions (wip)
https://github.com/insertish/bluej-bug-demo

[12] Fontsource. VT323
https://fontsource.org/fonts/vt323

References in code.

[13] `Ansi.java` StackOverflow. How to print color in console using System.out.println?
https://stackoverflow.com/a/5762502

[14] `LocalisedIO.java` StackOverflow. What is the equivalent of Regex-replace-with-function-evaluation in Java 7? https://stackoverflow.com/a/27359491

[15] `JTerminalFrame.java` How to detect a key press in Java
https://stackoverflow.com/a/21970006

[16] `JTerminalView.java` How to "do something" on Swing component resizing?
https://stackoverflow.com/a/8917978

[17] `JTerminalView.java` Java Documentation. Font Concepts
https://docs.oracle.com/javase/tutorial/2d/text/fontconcepts.html

[18] `JTerminalView.java` Java Documentation. Java Thread Primitive Deprecation https://docs.oracle.com/javase/1.5.0/docs/guide/misc/threadPrimitiveDeprecation.html

[19] `JTerminalView.java` StackOverflow. Drawing Canvas on JFrame
https://stackoverflow.com/a/17922749

[20] `TerminalEmulator.java` StackOverflow. Passing values between 2 threads without intrrrupting each other https://stackoverflow.com/a/23413506

Libraries used:

[21] *com.moandjiezana.toml* https://github.com/mwanji/toml4j

[22] *commons-io* `https://commons.apache.org/proper/commons-io`

[23] *kuusisto.tinysound* (fork by DrogoniEntity) `https://github.com/DrogoniEntity/TinySound`

[24] *com.projectdarkstar.ext.jorbis*
`https://search.maven.org/artifact/com.projectdarkstar.ext.jorbis/jorbis`

[25] *com.googlecode.soundlibs.tritonus-share*
`https://search.maven.org/artifact/com.googlecode.soundlibs/tritonus-share/0.3.7.4/bundle`

[26] *com.googlecode.soundlibs.vorbisspi*
`https://search.maven.org/artifact/com.googlecode.soundlibs/vorbisspi/1.0.3.3/bundle`

Sounds used:

[27] Freesound. "Money Bag" by PhilSavlem (licensed under CC0)
`https://freesound.org/people/PhilSavlem/sounds/338260/`

[28] Freesound. "bay of fundy 01.flac" by tim.kahn (licensed CC BY 3.0)
`https://freesound.org/people/tim.kahn/sounds/127569/`

[29] Freesound. "bay of fundy 02.flac" by tim.kahn (licensed CC BY 3.0)
`https://freesound.org/people/tim.kahn/sounds/127568/`

[30] Freesound. "2020-03-17 Lofi Trip Hop" by Doctor_Dreamchip (licensed CC BY 3.0)
`https://freesound.org/people/Doctor_Dreamchip/sounds/511279/`

[31] Freesound. "Remix of GioMilko Freesound #417960.flac" by Timbre (licensed CC BY-NC 3.0)
`https://freesound.org/people/Timbre/sounds/418692/`

[32] Freesound. "loopable Lo-fied remix of Tenshi_Mixer freesound #585947.flac" by Timbre (licensed CC BY-NC 3.0) `https://freesound.org/people/Timbre/sounds/586988/`

[33] Freesound. "Birds In The Forest" by BurghRecords (licensed under CC0)
`https://freesound.org/people/BurghRecords/sounds/456123/`

[34] Freesound. "Bed of entanglement" by CosmicD (licensed CC BY 3.0)
`https://freesound.org/people/CosmicD/sounds/133007/`

[35] Freesound. "VOC_150325-0973-1_HK_citywalk.wav" by kevp888 (licensed CC BY 3.0)
`https://freesound.org/people/kevp888/sounds/440973/`

[36] Freesound. "VOC_150325-0972_HK_citywalk.wav" by kevp888 (licensed CC BY 3.0)
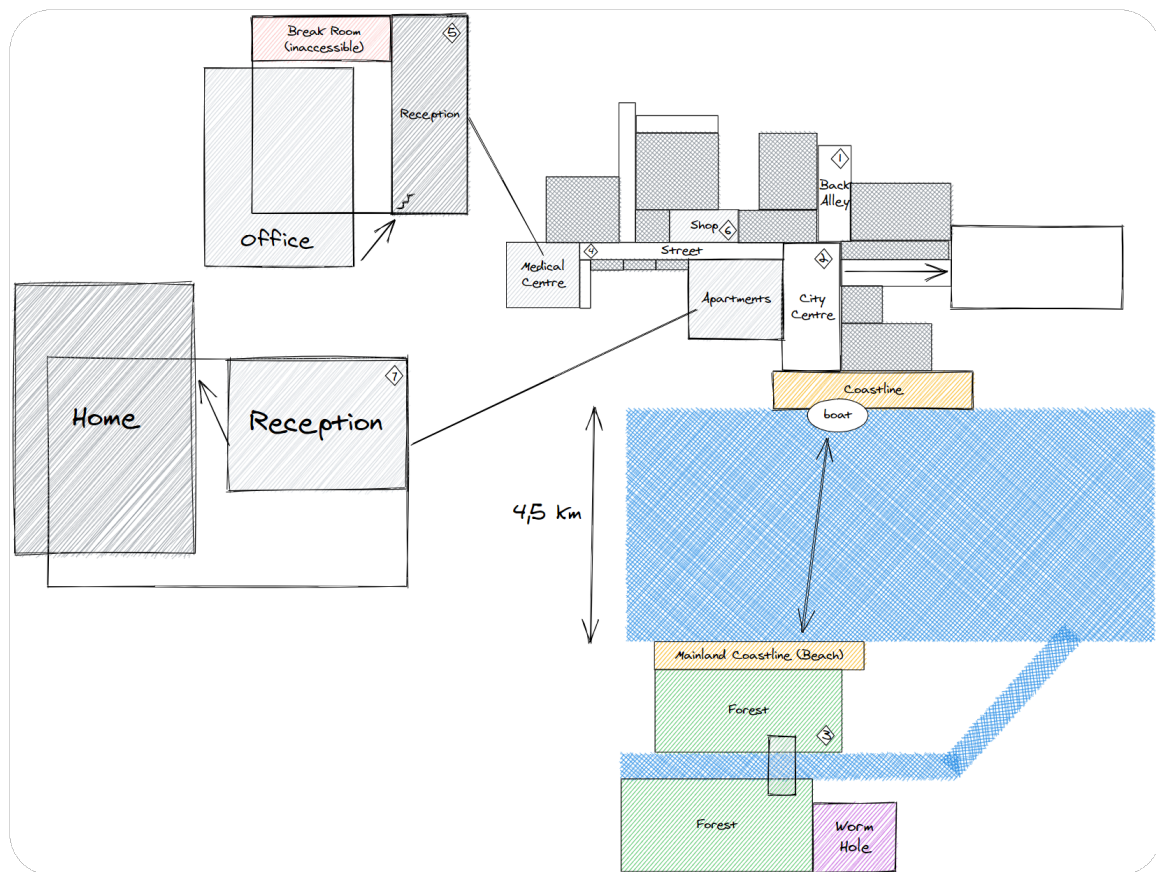`https://freesound.org/people/kevp888/sounds/440974/`

Figure 1: World Map

Figure 2: Shot of Animacity as seen in Episode 1 at 3:02 of
Brand New Animal[4]

Figure 3: Michiru Kagemori and Marie Itami pictured left
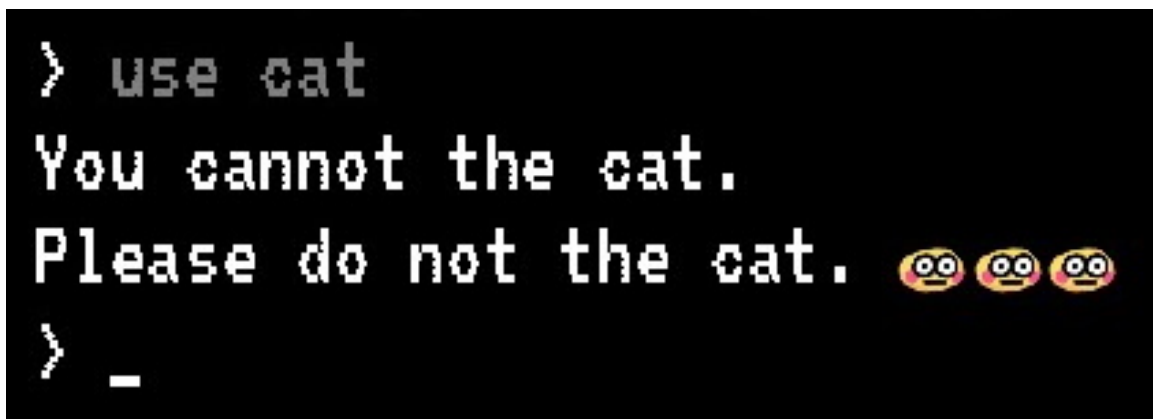to right in Episode 1 at 12:51 of Brand New Animal[4]
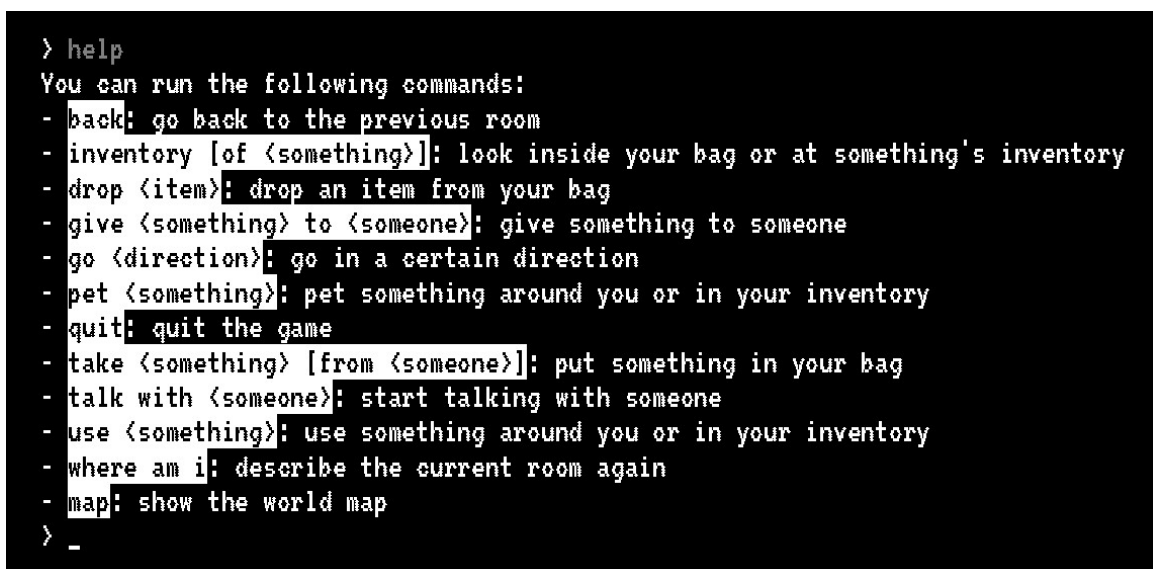

Figure 4: Sample emoji output
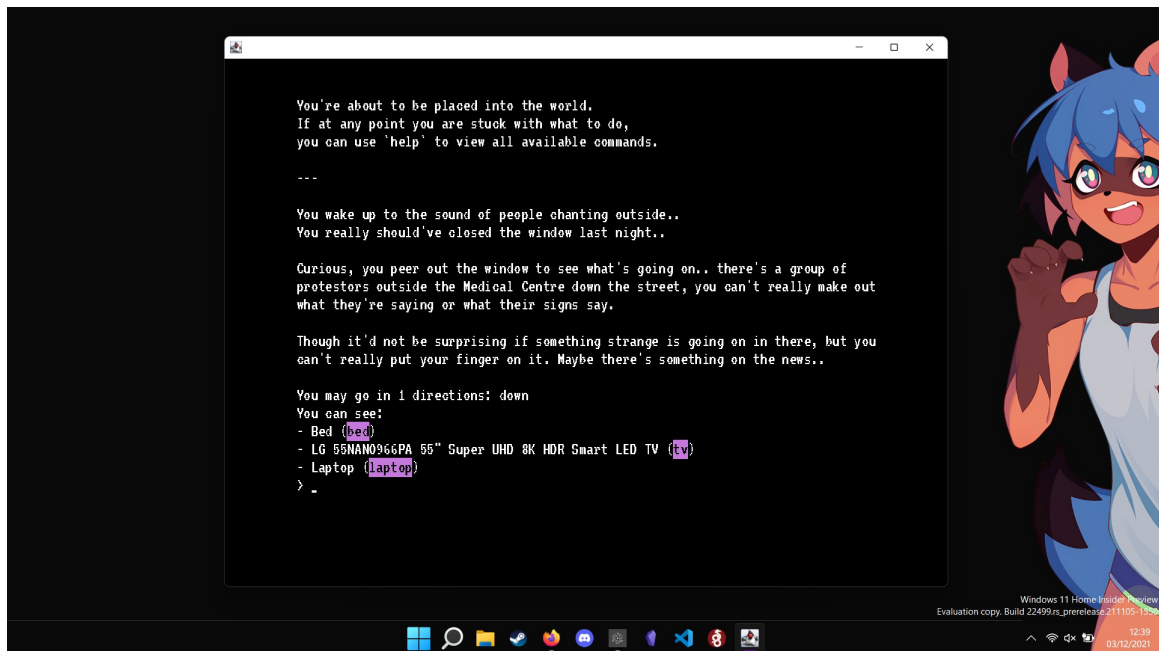

Figure 5: Output from the help command

Figure 6: The terminal emulator

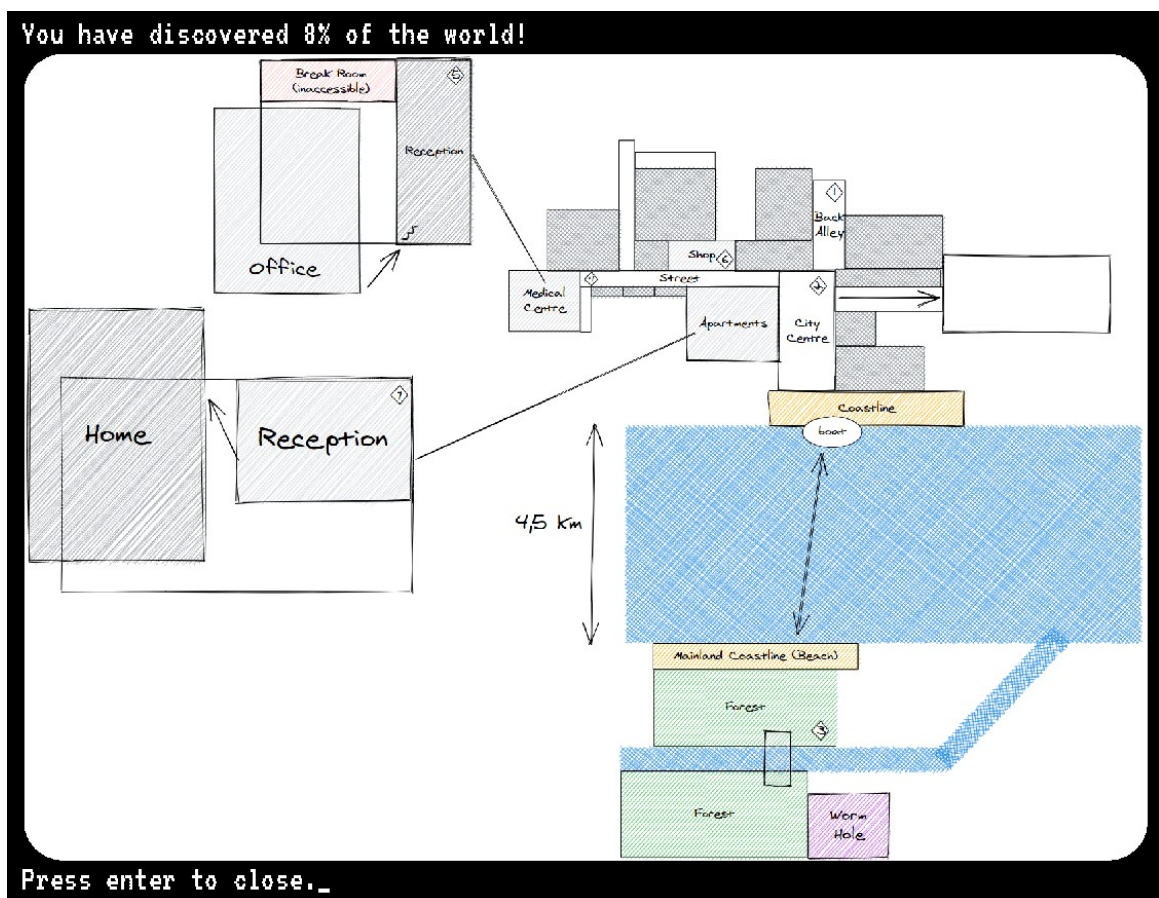Figure 7: Output from the map command

```java
package uk.insrt.coursework.zuul.events;

/**
 * Represents a single event fired from
 * any source to be consumed by anything.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class Event {
    private boolean propagating = true;

    /**
     * Whether this event can continue running.
     * @return Whether propogation of this event was stopped
     */
    public boolean canRun() {
        return this.propagating;
    }

    /**
     * Stop further propagation of this event.
     */
    public void stopPropagation() {
        this.propagating = false;
    }
}
```

Figure 8: Event.java
(uk.insrt.coursework.zuul.events.Event)

```java
package uk.insrt.coursework.zuul.io;

/**
 * Interface representing an arbitrary IO system.
 * This can be implemented to input or output from various interfaces.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public interface IOSystem {
    /**
     * Print a string out through an arbitrary output channel.
     * @param out String to print
     */
    public void print(String out);

    /**
     * Print a string out through an arbitrary output channel and append {@code \n}.
     * @param out String to print
     */
    public void println(String out);

    /**
     * Read a String up until the first encountered {@code \n} from an arbitrary
     input channel.
     * @return String of line read in
     */
    public String readLine();

    /**
     * Dispose of the arbitrary input and output channels.
     */
    public void dispose();
}
```

Figure 9: IOSystem.java
(uk.insrt.coursework.zuul.io.IOSystem)

```java
package uk.insrt.coursework.zuul.io;

import java.util.Scanner;

/**
 * A simple IO system implementation which feeds
 * into System.out and takes data from System.in
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class StandardIO implements IOSystem {
    private Scanner reader;

    /**
     * Construct a new StandardIO.
     */
    public StandardIO() {
        this.reader = new Scanner(System.in);
    }

    @Override
    public void print(String out) {
        System.out.print(out);
    }

    @Override
    public void println(String out) {
        System.out.println(out);
    }

    @Override
    public String readLine() {
        return this.reader.nextLine();
    }

    @Override
    public void dispose() {}
}
```

Figure 10: StandardIO.java
(uk.insrt.coursework.zuul.io.StandardIO)

```java
package uk.insrt.coursework.zuul.io;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

import uk.insrt.coursework.zuul.util.Localisation;

/**
 * Translate and localise any incoming output.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class LocalisedIO implements IOSystem {
    private final Pattern pattern = Pattern.compile("<([\\w\\.]+?)>");

    private IOSystem io;
    private Localisation locale;

    /**
     * Construct a new LocalisedIO.
     * @param io Provided IO system we should feed into
     * @param locale Locale to apply to any i18n strings
     */
    public LocalisedIO(IOSystem io, Localisation locale) {
        this.io = io;
        this.locale = locale;
    }

    /**
     * Replace i18n strings in any given String with their actual localised values.
     * Using replacement code from https://stackoverflow.com/a/27359491.
     * @param input String to process
     * @return Final processed string
     */
    private String replace(String input) {
        StringBuffer result = new StringBuffer();
        Matcher matcher = this.pattern.matcher(input);

        while (matcher.find()) {
            matcher.appendReplacement(result, this.locale.get(matcher.group(1)));
        }

        matcher.appendTail(result);
        return result.toString();
    }

    @Override
    public void print(String out) {
        this.io.print(this.replace(out));
    }

    @Override
    public void println(String out) {
        this.io.println(this.replace(out));
    }

    @Override
    public String readLine() {
        return this.io.readLine();
    }

    @Override
    public void dispose() {
        this.io.dispose();
    }
}
```

Figure 11: LocalisedIO.java
(uk.insrt.coursework.zuul.io.LocalisedIO)

```java
package uk.insrt.coursework.zuul.events;

import java.util.HashMap;
import java.util.HashSet;
import java.util.LinkedHashSet;

/**
 * Event system which manages taking in events
 * from different sources and handles them
 * by firing callbacks on event listeners.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class EventSystem {
    private HashMap<Class<? extends Event>, LinkedHashSet<IEventListener<? extends
    Event>>> listeners = new HashMap<>();

    /**
     * Get existing Event listener list or create a new one if not exists.
     * @param event Event
     * @return Set of event listeners
     */
    private HashSet<IEventListener<? extends Event>> getList(Class<? extends Event>
    event) {
        var list = this.listeners.get(event);
        if (list == null) {
            list = new LinkedHashSet<>();
            this.listeners.put(event, list);
        }

        return list;
    }

    /**
     * Add a new event listener to this system.
     * @param <E> Generic Event type
     * @param event Event to remove from
     * @param listener Event listener callback
     */
    public<E extends Event> void addListener(Class<E> event, IEventListener<E>
    listener) {
        this.getList(event).add(listener);
    }

    /**
     * Remove an new event listener from this system.
     * @param <E> Generic Event type
     * @param event Event to remove from
     * @param listener Event listener callback
     */
    public<E extends Event> void removeListener(Class<E> event, IEventListener<E>
    listener) {
        this.getList(event).remove(listener);
    }

    /**
     * Emit an Event.
     * @param <E> Generic Event type
     * @param event Event to emit
     */
    @SuppressWarnings("unchecked")
    public <E extends Event> void emit(E event) {
        var listeners = this.listeners.get(event.getClass());
        if (listeners == null) return;

        for (@SuppressWarnings("rawtypes") IEventListener listener : listeners) {
            listener.onEvent(event);
            // Previously, there was a try catch ClassCastException
            // but I've since constricted the types on `addListener`
            // and `removeListener` so this should never happen.

            if (!event.canRun())
                break;
        }
    }
}
```

13
Figure 12: EventSystem.java
(uk.insrt.coursework.zuul.events.EventSystem)

```java
package uk.insrt.coursework.zuul.dialogue;

import uk.insrt.coursework.zuul.io.IOSystem;

/**
 * An option which branches off a {@link DialogueNode} into another node.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class DialogueOption<T> {
    private IDialogueHandler<T> handler;

    private String description;
    private boolean shouldExit;
    private T target;

    /**
     * Construct a new simple DialogueOption with a description and destination.
     * @param description Description of this option
     * @param target Target node to jump to
     */
    public DialogueOption(String description, T target) {
        this.target = target;
        this.description = description;
    }

    /**
     * Construct a complex DialogueOption with a description and select handler.
     * @param description Description of this option
     * @param handler Method called when this option is selected
     */
    public DialogueOption(String description, IDialogueHandler<T> handler) {
        this.handler = handler;
        this.description = description;
    }

    /**
     * Tell the Dialogue system to exit if this option is selected.
     * @return This dialogue option so method calls can be chained
     */
    public DialogueOption<T> mustExit() {
        this.shouldExit = true;
        return this;
    }

    /**
     * Get the description of this option.
     * @return Description string
     */
    public String getDescription() {
        return this.description;
    }

    /**
     * Get the destination of this option.
     * @return Destination if it exists
     */
    public T getTarget() {
        return this.target;
    }

    /**
     * Handle the player selecting this dialogue option.
     * @param io Provided IO system
     * @return The new node or null if we should exit and stay put.
     */
    public T handle(IOSystem io) {
        if (this.handler != null) {
            return this.handler.onAction(io);
        } else if (!this.shouldExit) {
            return this.target;
        }

        return null;
    }
}
```

14

Figure 13: DialogueOption.java

(uk.insrt.coursework.zuul.dialogue.DialogueOption)

```java
package uk.insrt.coursework.zuul.ui;

import java.awt.Graphics;

import uk.insrt.coursework.zuul.events.Event;

/**
 * Event fired when the terminal emulator draws a new frame.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class EventDraw extends Event {
    private Graphics g;
    private float ox;
    private float oy;
    private float fw;
    private float fh;

    /**
     * Construct a new EventDraw Event.
     * @param g Graphics context
     * @param ox Origin X position
     * @param oy Origin Y position
     * @param fw Font character width
     * @param fh Font character height
     */
    public EventDraw(Graphics g, float ox, float oy, float fw, float fh) {
        this.g = g;
        this.ox = ox;
        this.oy = oy;
        this.fw = fw;
        this.fh = fh;
    }

    /**
     * Get the Graphics relating to this event.
     * @return Graphics
     */
    public Graphics getGraphics() {
        return this.g;
    }

    /**
     * Get the origin X position of the contents of the terminal.
     * @return X position
     */
    public float getOriginX() {
        return this.ox;
    }

    /**
     * Get the origin Y position of the contents of the terminal.
     * @return Y position
     */
    public float getOriginY() {
        return this.oy;
    }

    /**
     * Get the character width.
     * @return Character width
     */
    public float getCharWidth() {
        return this.fw;
    }

    /**
     * Get the character height.
     * @return Character height
     */
    public float getCharHeight() {
        return this.fh;
    }
}
```

Figure 14: EventDraw.java
(uk.insrt.coursework.zuul.ui.EventDraw)

```java
public<E extends Event> void addListener(Class<E> event,
    IEventListener<E> listener) {
    this.getList(event).add(listener);
}
```

Figure 15: Excerpt from EventSystem.java
(uk.insrt.coursework.zuul.events.EventSystem)

```java
/**
 * Get the current weight of this inventory.
 * @return Weight (in kg)
 */
public double getWeight() {
    return this
        .items
        .stream()
        .mapToDouble(Entity::getWeight)
        .sum();
}

/**
 * Check if the inventory is full.
 * @return True if the weight is greater than the max weight
 */
public boolean isFull() {
    return this.getWeight() >= this.getMaxWeight();
}

/**
 * Add an entity to this inventory.
 *
 * There must be sufficient space for the entity.
 * @param entity Target Entity
 * @return Whether we successfully added the new entity.
 */
public boolean add(Entity entity) {
    if (this.getWeight() + entity.getWeight() > this.maxWeight) {
        return false;
    }

    this.items.add(entity);
    return true;
}
```

Figure 16: Excerpt from Inventory.java
(uk.insrt.coursework.zuul.entities.Inventory)

```java
/**
 * Write a string value to the text buffer.
 * @param value String value to write
 */
public void write(String value) {
    // Write each character sequentially.
    for (int i=0;i<value.length();i++) {
        char c = value.charAt(i);

        // If we encounter an Ansi escape character, then take the
        // substring from this point on and determine if it is a valid
        // escape code. If it is, apply any changes before continuing.
        if (c == '\u001B') {
            Matcher matcher = Ansi.AnsiPattern.matcher(value.substring(i));
            if (matcher.find()) {
                int v = Integer.parseInt(matcher.group(1));
                i += 3 + (v > 9 ? 1 : 0);

                if (v == 0) {
                    this.bg = Color.BLACK;
                    this.fg = Color.WHITE;
                } else if (v >= 30 && v < 38) {
                    this.fg = Ansi.fromEscapeCode(v);
                } else if (v >= 40 && v < 48) {
                    this.bg = Ansi.fromEscapeCode(v);
                }

                continue;
            }
        }

        this.write(c);
    }
}
```

Figure 17: Excerpt from TextBuffer.java
(uk.insrt.coursework.zuul.ui.TextBuffer)

```java
package uk.insrt.coursework.zuul.commands;

import java.util.regex.Matcher;

import uk.insrt.coursework.zuul.world.Direction;

/**
 * Wrapper around regex Matcher for deriving the values of given arguments in commands.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class Arguments {
    private Matcher matcher;

    /**
     * Construct a new Arguments wrapper.
     * @param matcher Regex Matcher
     */
    public Arguments(Matcher matcher) {
        this.matcher = matcher;
    }

    /**
     * Take a named group from the Matcher.
     * We assume that the provided Regex doesn't match the group if it's empty.
     * @param group Named group
     * @return String value of named group or null if it doesn't exist
     */
    public String group(String group) {
        // We ignore and return null on error as a bit of convenience.
        // This may not be best practice but it is justified by the fact
        // that it avoids an incredible amount of boilerplate further up
        // the chain, and in my opinion that's worth this design decision.
        try {
            return this.matcher.group(group);
        } catch (Exception e) {
            return null;
        }
    }
}
```

```java
42        /**
43         * Check whether this named group was matched.
44         * @param group Named group
45         * @return Whether this named group was matched
46         */
47        public boolean has(String group) {
48            return this.group(group) != null;
49        }
50
51        /**
52         * Get the provided Direction.
53         * @return Parsed Direction value
54         */
55        public Direction direction() {
56            return Direction.fromString(this.group("direction"));
57        }
58    }
```

```java
package uk.insrt.coursework.zuul.commands;

import java.util.List;
import java.util.regex.Pattern;

import uk.insrt.coursework.zuul.entities.Entity;
import uk.insrt.coursework.zuul.entities.Inventory;
import uk.insrt.coursework.zuul.util.Search;
import uk.insrt.coursework.zuul.world.World;

/**
 * Representation of an action which can be performed by the user.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public abstract class Command {
    private Pattern[] patterns;
    private String syntax;
    private String usage;

    /**
     * Construct a new Command.
     * @param syntax Information about how to use the command
     * @param usage Information about what the command does
     * @param patterns Patterns to execute this command on
     */
    public Command(String syntax, String usage, Pattern[] patterns) {
        this.patterns = patterns;
        this.syntax = syntax;
        this.usage = usage;
    }

    /**
     * Get information about how to use the command.
     * @return String Information about how to use the command
     */
    public String getSyntax() {
        return this.syntax;
    }
```

```java
    /**
     * Get information about what the command does.
     * @return String Information about what the command does
     */
    public String getUsage() {
        return this.usage;
    }

    /**
     * Get all applicable patterns to match to execute this command.
     * @return Regex Pattern array
     */
    public Pattern[] getPatterns() {
        return this.patterns;
    }

    /**
     * Check whether this command is visible to the player in the help menu.
     * @return True if visible
     */
    public boolean isVisible() {
        return true;
    }

    /**
     * Run this command within the scope of a world and with any parsed arguments.
     * @param world Current World object
     * @param args Arguments passed into command
     * @return Boolean indicating whether the game loop should exit.
     */
    public abstract boolean run(World world, Arguments args);

    /**
     * Filter entities by those that are in the current room.
     */
    public static final int FILTER_ROOM = 1;

    /**
     * Filter entities by those that are in the player's inventory.
     */
    public static final int FILTER_INVENTORY = 2;
```

```java
 83
 84        /**
 85         * Don't filter entities and instead search through both the current room and the player's inventory.
 86         */
 87        public static final int FILTER_ALL = FILTER_ROOM + FILTER_INVENTORY;
 88
 89        /**
 90         * Given a World and filter, use the provided Arguments and the relevant group to find an entity.
 91         * If an Entity is not found or not provided, the appropriate error is displayed to the player.
 92         * @param world World to look for the Entity within
 93         * @param filter Integer value which represents the filtering, specify one of: {@link #FILTER_ROOM}, {@link
#FILTER_INVENTORY}, {@link #FILTER_ALL}
 94         * @param args Arguments object to pull information out of
 95         * @param group Group we should pull the Entity query out of
 96         * @param failure Failure message if an Entity is not specified
 97         * @return An Entity if one is found, or null if one isn't.
 98         */
 99        public Entity findEntity(World world, int filter, Arguments args, String group, String failure) {
100            String name = args.group(group);
101            if (name == null) {
102                world.getIO().println(failure);
103                return null;
104            }
105
106            // Search the inventory first.
107            Entity player = world.getPlayer();
108            Entity entity = null;
109            if ((filter & FILTER_INVENTORY) == FILTER_INVENTORY) {
110                Inventory inventory = player.getInventory();
111                entity = Search.findEntity(inventory.getItems(), name, true);
112            }
113
114            // If we haven't found an entity yet, search the room.
115            if (entity == null
116             && (filter & FILTER_ROOM) == FILTER_ROOM) {
117                List<Entity> entities = world.getEntitiesInRoom(player.getRoom());
118                entity = Search.findEntity(entities, name, true);
119            }
120
121            if (entity == null) {
122                world.getIO().println("<selectors.cant_find.1> " + name + " <selectors.cant_find.2>.");
```

```java
123              }
124
125              return entity;
126          }
127
128          /**
129           * Given a World and filter, use the provided Arguments and using the group "entity" to find an entity.
130           * If an Entity is not found or not provided, the appropriate error is displayed to the player.
131           * @param world World to look for the Entity within
132           * @param filter Integer value which represents the filtering, specify one of: {@link #FILTER_ROOM}, {@link
#FILTER_INVENTORY}, {@link #FILTER_ALL}
133           * @param args Arguments object to pull information out of
134           * @param failure Failure message if an Entity is not specified
135           * @return An Entity if one is found, or null if one isn't.
136           */
137          public Entity findEntity(World world, int filter, Arguments args, String failure) {
138              return this.findEntity(world, filter, args, "entity", failure);
139          }
140
141          /**
142           * Given a World and using the {@link #FILTER_ROOM} filter, use the provided Arguments and using the group "entity"
to find an entity.
143           * If an Entity is not found or not provided, the appropriate error is displayed to the player.
144           * @param world World to look for the Entity within
145           * @param args Arguments object to pull information out of
146           * @param failure Failure message if an Entity is not specified
147           * @return An Entity if one is found, or null if one isn't.
148           */
149          public Entity findEntity(World world, Arguments args, String failure) {
150              return this.findEntity(world, FILTER_ROOM, args, failure);
151          }
152
153          /**
154           * Given a World and using the {@link #FILTER_ROOM} filter, use the provided Arguments and the relevant group to
find an entity.
155           * If an Entity is not found or not provided, the appropriate error is displayed to the player.
156           * @param world World to look for the Entity within
157           * @param args Arguments object to pull information out of
158           * @param group Group we should pull the Entity query out of
159           * @param failure Failure message if an Entity is not specified
160           * @return An Entity if one is found, or null if one isn't.
```

```java
161            */
162        public Entity findEntity(World world, Arguments args, String group, String failure) {
163            return this.findEntity(world, FILTER_ROOM, args, group, failure);
164        }
165    }
```

```java
package uk.insrt.coursework.zuul.commands;

import java.util.ArrayList;
import java.util.List;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import uk.insrt.coursework.zuul.commands.core.CommandBack;
import uk.insrt.coursework.zuul.commands.core.CommandBag;
import uk.insrt.coursework.zuul.commands.core.CommandDrop;
import uk.insrt.coursework.zuul.commands.core.CommandGive;
import uk.insrt.coursework.zuul.commands.core.CommandGo;
import uk.insrt.coursework.zuul.commands.core.CommandHelp;
import uk.insrt.coursework.zuul.commands.core.CommandPet;
import uk.insrt.coursework.zuul.commands.core.CommandQuit;
import uk.insrt.coursework.zuul.commands.core.CommandTake;
import uk.insrt.coursework.zuul.commands.core.CommandTalk;
import uk.insrt.coursework.zuul.commands.core.CommandUse;
import uk.insrt.coursework.zuul.commands.core.CommandWhereAmI;
import uk.insrt.coursework.zuul.world.World;

/**
 * Command handler which constructs, then resolves
 * and executes commands from an arbitrary input.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class CommandManager {
    private ArrayList<Command> commands = new ArrayList<>();

    /**
     * Construct a new CommandManager.
     *
     * You should only need one present at any given time.
     */
    public CommandManager() {
        this.initialiseCommands();
    }

    /**
```

```java
     * Register a new Command.
     * @param command Command
     */
    public void registerCommand(Command command) {
        this.commands.add(command);
    }

    /**
     * Register multiple commands.
     * @param commands Command array
     */
    public void registerCommands(Command[] commands) {
        for (Command command : commands) {
            this.registerCommand(command);
        }
    }

    /**
     * Get Commands provided by this Command manager.
     * @return List of Commands
     */
    public List<Command> getCommands() {
        return this.commands;
    }

    /**
     * Initialise all the commands a player can execute.
     */
    private void initialiseCommands() {
        final Command[] DEFAULT_COMMANDS = {
            new CommandBack(),
            new CommandBag(),
            new CommandDrop(),
            new CommandGive(),
            new CommandGo(),
            new CommandHelp(this),
            new CommandPet(),
            new CommandQuit(),
            new CommandTake(),
            new CommandTalk(),
            new CommandUse(),
```

```java
 83                new CommandWhereAmI(),
 84            };
 85
 86            this.registerCommands(DEFAULT_COMMANDS);
 87        }
 88
 89        /**
 90         * Interpret a given command and execute it within the scope of a given world.
 91         * @param world Current World object
 92         * @param cmd Arbitrary input to match against
 93         * @return Boolean indicating whether the game loop should exit.
 94         */
 95        public boolean runCommand(World world, String cmd) {
 96            for (Command command : this.commands) {
 97                for (Pattern pattern : command.getPatterns()) {
 98                    Matcher matcher = pattern.matcher(cmd);
 99                    if (matcher.find()) {
100                        Arguments arguments = new Arguments(matcher);
101                        return command.run(world, arguments);
102                    }
103                }
104            }
105
106            world.getIO().println("<commands.unknown>");
107            return false;
108        }
109    }
```

```java
package uk.insrt.coursework.zuul.commands.core;

import java.util.regex.Pattern;

import uk.insrt.coursework.zuul.commands.Arguments;
import uk.insrt.coursework.zuul.commands.Command;
import uk.insrt.coursework.zuul.world.World;

/**
 * Command which allows the Player to walk back through the previous Rooms.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class CommandBack extends Command {
    public CommandBack() {
        super("back", "<commands.back>",
            new Pattern[] {
                Pattern.compile("^(?:(?:go|walk)\\s+)*back(?!\\w)"),
                // back, go back, walk back
            });
    }

    @Override
    public boolean run(World world, Arguments arguments) {
        // We call a specialised method on the player as we keep
        // track of visited rooms within the Player class itself.
        world.getPlayer().back();
        return false;
    }
}
```

```java
package uk.insrt.coursework.zuul.commands.core;

import java.text.DecimalFormat;
import java.util.regex.Pattern;

import uk.insrt.coursework.zuul.commands.Arguments;
import uk.insrt.coursework.zuul.commands.Command;
import uk.insrt.coursework.zuul.entities.Entity;
import uk.insrt.coursework.zuul.entities.Inventory;
import uk.insrt.coursework.zuul.io.Ansi;
import uk.insrt.coursework.zuul.io.IOSystem;
import uk.insrt.coursework.zuul.world.World;

/**
 * Command which allows the Player to look at their or another Entity's inventory.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class CommandBag extends Command {
    private final DecimalFormat format = new DecimalFormat("0.00");

    public CommandBag() {
        super("inventory [of <selectors.something>]", "<commands.bag.usage>",
            new Pattern[] {
                Pattern.compile("^(?:b(?:ag)*|inv(?:entory)*)(?:\\s+(?<entity>[\\w\\s]+))*"),
                // b, bag, inv, inventory, bag of <entity>, inventory of <entity>, (+2)
            });
    }

    @Override
    public boolean run(World world, Arguments arguments) {
        IOSystem io = world.getIO();

        // Figure out if we're checking our own inventory or another entity's inventory.
        Entity entity;
        boolean ours;
        if (arguments.has("entity")) {
            ours = false;
            entity = this.findEntity(world, arguments, "<commands.bag.cant_find>");
            if (entity == null) return false;
```

```java
        } else {
            ours = true;
            entity = world.getPlayer();
        }

        // Get the selected entity's inventory and provide output if empty.
        Inventory inv = entity.getInventory();
        if (inv.getWeight() == 0) {
            if (ours) {
                io.println("<commands.bag.empty> <commands.bag.can_carry_kg> "
                    + inv.getMaxWeight() + " kg.");
            } else {
                io.println(entity.getHighlightedName() + " <commands.bag.entity_empty>.");
            }

            return false;
        }

        // Otherwise describe some statistics about the inventory.
        if (ours) {
            io.println("<commands.bag.are_carrying_kg> " + this.format.format(inv.getWeight())
                + " / " + inv.getMaxWeight() + " kg.\n<commands.bag.look_in_bag>:");
        } else {
            io.println(entity.getHighlightedName() + " <commands.bag.entity_appears_to_have>:");
        }

        // Describe all the items in this inventory we are currently looking at.
        for (Entity item : inv.getItems()) {
            io.println("- " + Ansi.Yellow + item.getWeight() + " kg"
                + Ansi.Reset + " " + item.describe()
                + " (" + item.getHighlightedName() + ")");
        }

        return false;
    }
}
```

```java
package uk.insrt.coursework.zuul.commands.core;

import java.util.regex.Pattern;

import uk.insrt.coursework.zuul.commands.Arguments;
import uk.insrt.coursework.zuul.commands.Command;
import uk.insrt.coursework.zuul.entities.Entity;
import uk.insrt.coursework.zuul.io.Ansi;
import uk.insrt.coursework.zuul.world.World;

/**
 * Command which allows the Player to drop any item in their inventory.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class CommandDrop extends Command {
    public CommandDrop() {
        super("drop <selectors.item>", "<commands.drop.usage>",
            new Pattern[] {
                Pattern.compile("^(?:drop|place|put down)(?:\\s+(?<entity>[\\w\\s]+))*")
                // drop, place, put down, drop <item>, place <item>, put down <item>
            });
    }

    @Override
    public boolean run(World world, Arguments args) {
        // Find the given entity within our inventory and drop it if it's found.
        Entity entity = this.findEntity(world, Command.FILTER_INVENTORY, args, "<commands.drop.nothing_specified>");
        if (entity != null) {
            world.getIO().println("<commands.drop.dropped.1> " + Ansi.BackgroundWhite + Ansi.Black
                + entity.getName() + Ansi.Reset + " <commands.drop.dropped.2>!");
            entity.setLocation(world.getPlayer().getRoom());
        }

        return false;
    }
}
```

```java
package uk.insrt.coursework.zuul.commands.core;

import java.util.regex.Pattern;

import uk.insrt.coursework.zuul.commands.Arguments;
import uk.insrt.coursework.zuul.commands.Command;
import uk.insrt.coursework.zuul.entities.Entity;
import uk.insrt.coursework.zuul.entities.EntityPlayer;
import uk.insrt.coursework.zuul.entities.actions.IGiveable;
import uk.insrt.coursework.zuul.world.World;

/**
 * Command which allows the player to give something to someone.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class CommandGive extends Command {
    public CommandGive() {
        super("give <selectors.something> to <selectors.someone>", "<commands.give.usage>",
            new Pattern[] {
                Pattern.compile("^(?:give|put)(?:\\s+(?<item>[\\w\\s]+)\\s+(?:to|in)\\s+(?<entity>[\\w\\s]+))*")
                // give, put, give <something> to <someone>, put <something> in <something>, (+2)
            });
    }

    @Override
    public boolean run(World world, Arguments args) {
        // Find the entity we want to give in the room or our inventory.
        Entity item = this.findEntity(world, Command.FILTER_ALL, args, "item", "<commands.give.nothing_specified>");
        if (item == null) return false;

        // Explicitly deny the player being given to anything, otherwise we will end up in an inventory.
        var io = world.getIO();
        if (item instanceof EntityPlayer) {
            io.println("<commands.give.denied_player>");
            return false;
        }

        // Find the target to give to.
        Entity target = this.findEntity(world, args, "<commands.give.no_target>");
```

```java
42          if (target == null) return false;
43
44          // If the target entity is an IGiveable, check if they accept this item.
45          if (target instanceof IGiveable) {
46              ((IGiveable) target).give(item);
47          } else {
48              io.println("<commands.give.denied.1> " + item.getHighlightedName()
49                  + " <commands.give.denied.2> " + target.getHighlightedName() + "!");
50          }
51
52          return false;
53      }
54  }
```

```java
package uk.insrt.coursework.zuul.commands.core;

import java.util.regex.Pattern;

import uk.insrt.coursework.zuul.commands.Arguments;
import uk.insrt.coursework.zuul.commands.Command;
import uk.insrt.coursework.zuul.world.Direction;
import uk.insrt.coursework.zuul.world.World;

/**
 * Command which allows the Player to walk in a particular Direction.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class CommandGo extends Command {
    public CommandGo() {
        super("go <selectors.direction>", "<commands.go.usage>",
            new Pattern[] {
                Pattern.compile("^(?:go|walk)(?:\\s+(?<direction>[\\w\\s]+))*")
                // go, walk, go <direction>, walk <direction>
            });
    }

    @Override
    public boolean run(World world, Arguments arguments) {
        Direction direction = arguments.direction();
        if (direction == null) {
            world.getIO().println("<commands.go.nothing_specified>");
            return false;
        }

        world.getPlayer().go(direction);
        return false;
    }
}
```

```java
package uk.insrt.coursework.zuul.commands.core;

import java.util.regex.Pattern;
import java.util.stream.Collectors;

import uk.insrt.coursework.zuul.commands.Arguments;
import uk.insrt.coursework.zuul.commands.Command;
import uk.insrt.coursework.zuul.commands.CommandManager;
import uk.insrt.coursework.zuul.io.Ansi;
import uk.insrt.coursework.zuul.world.World;

/**
 * Command which allows the player to list all the available commands.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class CommandHelp extends Command {
    private CommandManager commandManager;

    public CommandHelp(CommandManager commandManager) {
        super("help", "<commands.help.usage>",
            new Pattern[] {
                Pattern.compile("^(?:h(?:elp)*)(?!\\w)")
                // h, help
            });

        this.commandManager = commandManager;
    }

    @Override
    public boolean run(World world, Arguments arguments) {
        // Describe all the commands the player can run.
        world.getIO()
            .println(
            "<commands.help.can_run>\n" +
            this.commandManager
                .getCommands()
                .stream()
                .filter(Command::isVisible)
                .map(c -> "- " + Ansi.BackgroundWhite + Ansi.Black
```

```
42                        + c.getSyntax() + Ansi.Reset + ": " + c.getUsage())
43                    .collect(Collectors.joining("\n"))
44            );
45
46            return false;
47        }
48
49        @Override
50        public boolean isVisible() {
51            return false;
52        }
53    }
```

```java
package uk.insrt.coursework.zuul.commands.core;

import java.util.regex.Pattern;

import uk.insrt.coursework.zuul.commands.Arguments;
import uk.insrt.coursework.zuul.commands.Command;
import uk.insrt.coursework.zuul.entities.Entity;
import uk.insrt.coursework.zuul.entities.actions.IPettable;
import uk.insrt.coursework.zuul.world.World;

/**
 * Command which allows the player to pet another entity.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class CommandPet extends Command {
    public CommandPet() {
        super("pet <selectors.something>", "<commands.pet.usage>",
            new Pattern[] {
                Pattern.compile("^pet(?:\\s+(?<entity>[\\w\\s]+))*")
                // pet, pet <something>
            });
    }

    @Override
    public boolean run(World world, Arguments args) {
        // Scan the room for entities that have IPettable.
        Entity entity = this.findEntity(world, Command.FILTER_ALL, args, "<commands.pet.nothing_specified>");
        if (entity != null) {
            if (entity instanceof IPettable) {
                ((IPettable) entity).pet();
            } else {
                world.getIO().println("<commands.pet.denied> " + entity.getHighlightedName() + ".");
            }
        }

        return false;
    }
}
```

```java
package uk.insrt.coursework.zuul.commands.core;

import java.util.regex.Pattern;

import uk.insrt.coursework.zuul.commands.Arguments;
import uk.insrt.coursework.zuul.commands.Command;
import uk.insrt.coursework.zuul.world.World;

/**
 * Command which allows the player to quit the game.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class CommandQuit extends Command {
    public CommandQuit() {
        super("quit", "<commands.quit>",
            new Pattern[] {
                Pattern.compile("^quit|exit(?!\\w)"),
                // quit
            });
    }

    @Override
    public boolean run(World world, Arguments arguments) {
        // We return true from run() in order to tell the game loop to exit.
        return true;
    }
}
```

```java
package uk.insrt.coursework.zuul.commands.core;

import java.util.regex.Pattern;

import uk.insrt.coursework.zuul.commands.Arguments;
import uk.insrt.coursework.zuul.commands.Command;
import uk.insrt.coursework.zuul.entities.Entity;
import uk.insrt.coursework.zuul.entities.Inventory;
import uk.insrt.coursework.zuul.io.IOSystem;
import uk.insrt.coursework.zuul.util.Search;
import uk.insrt.coursework.zuul.world.World;

/**
 * Command which allows the Player to take an Entity and put it in their Inventory.
 * They may also take these Entities from other Entity Inventories.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class CommandTake extends Command {
    public CommandTake() {
        super("take <selectors.something> [from <selectors.someone>]", "<commands.take.usage>",
            new Pattern[] {
                Pattern.compile("^take\\s+(?<entity>[\\w\\s]+)\\s+from\\s+(?<other>[\\w\\s]+)"),
                Pattern.compile("^take(?:\\s+(?<entity>[\\w\\s]+))*")
                // take, take <item>, take <item> from <entity>
            });
    }

    @Override
    public boolean run(World world, Arguments args) {
        IOSystem io = world.getIO();
        Entity player = world.getPlayer();
        Inventory target = player.getInventory();

        // Detect if we are taking from another entity, in that case run different logic.
        if (args.has("other")) {
            String name = args.group("entity");
            Entity other = this.findEntity(world, args, "other", "<commands.take.nothing_specified>");
            if (other == null) return false;
```

```java
42          Entity item = Search.findEntity(other.getInventory().getItems(), name, true);
43          if (item == null) {
44              io.println(other.getHighlightedName() + " <commands.take.entity_does_not_have_entity> " + name + "!");
45              return false;
46          }
47
48          if (item.setLocation(target)) {
49              io.println("<commands.take.took.1> " + item.getHighlightedName()
50                  + " <commands.take.took.2> " + other.getHighlightedName()
51                  + " <commands.take.took.3>.");
52          } else {
53              io.println("<commands.take.denied.1> " + item.getName()
54                  + ", <commands.take.denied.2>.");
55          }
56
57          return false;
58      }
59
60      // Otherwise, look around the room and find something we can take.
61      Entity entity = this.findEntity(world, args, "<commands.take.item_not_specified>");
62      if (entity != null) {
63          if (entity == player) {
64              io.println("\u1F633");
65              return false;
66          }
67
68          if (entity.setLocation(target)) {
69              io.println("<commands.take.took.1> " + entity.getHighlightedName()
70                  + " <commands.take.took.3>.");
71          } else {
72              io.println("<commands.take.denied.1> " + entity.getHighlightedName()
73                  + ", <commands.take.denied.2>.");
74          }
75      }
76
77      return false;
78  }
79 }
```

```java
package uk.insrt.coursework.zuul.commands.core;

import java.util.regex.Pattern;

import uk.insrt.coursework.zuul.commands.Arguments;
import uk.insrt.coursework.zuul.commands.Command;
import uk.insrt.coursework.zuul.entities.Entity;
import uk.insrt.coursework.zuul.entities.actions.ITalkwith;
import uk.insrt.coursework.zuul.world.World;

/**
 * Command which allows the Player to talk with other Entities.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class CommandTalk extends Command {
    public CommandTalk() {
        super("talk with <selectors.someone>", "<commands.talk.usage>",
            new Pattern[] {
                Pattern.compile("^talk(?:(?:\\s*with|to)*(?:\\s+(?<entity>[\\w\\s]+))*)*")
                // talk, talk with <entity>, talk to <entity>
            });
    }

    @Override
    public boolean run(World world, Arguments args) {
        Entity entity = this.findEntity(world, args, "<commands.talk.nothing_specified>");
        if (entity != null) {
            if (entity instanceof ITalkwith) {
                ((ITalkwith) entity).talk();
            } else {
                world.getIO().println("<commands.talk.denied> " + entity.getHighlightedName() + ".");
            }
        }

        return false;
    }
}
```

```java
package uk.insrt.coursework.zuul.commands.core;

import java.util.regex.Pattern;

import uk.insrt.coursework.zuul.commands.Arguments;
import uk.insrt.coursework.zuul.commands.Command;
import uk.insrt.coursework.zuul.entities.Entity;
import uk.insrt.coursework.zuul.entities.actions.IUseable;
import uk.insrt.coursework.zuul.world.World;

/**
 * Command which allows the Player to use an Entity.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class CommandUse extends Command {
    public CommandUse() {
        super("use <selectors.something>", "<commands.use.usage>",
            new Pattern[] {
                Pattern.compile("^use(?:\\s+(?<entity>[\\w\\s]+))*")
                // use, use <entity>
            });
    }

    @Override
    public boolean run(World world, Arguments args) {
        Entity entity = this.findEntity(world, Command.FILTER_ALL, args, "<commands.use.nothing_specified>");
        if (entity != null) {
            if (entity instanceof IUseable) {
                ((IUseable) entity).use(world.getPlayer());
            } else {
                world.getIO().println("<commands.use.denied> " + entity.getHighlightedName() + ".");
            }
        }

        return false;
    }
}
```

```java
package uk.insrt.coursework.zuul.commands.core;

import java.util.regex.Pattern;

import uk.insrt.coursework.zuul.commands.Arguments;
import uk.insrt.coursework.zuul.commands.Command;
import uk.insrt.coursework.zuul.events.world.EventEntityEnteredRoom;
import uk.insrt.coursework.zuul.world.World;

/**
 * Command which allows the player to reorient themselves in the world.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class CommandWhereAmI extends Command {
    public CommandWhereAmI() {
        super("where am i", "<commands.where_am_i>",
            new Pattern[] {
                Pattern.compile("^where(\\s+am\\s+(i(?!\\w))*)*"),
                // where am i
            });
    }

    @Override
    public boolean run(World world, Arguments arguments) {
        // We can just re-emit the enter room event to
        // trigger the room description logic to run again.
        world.emit(new EventEntityEnteredRoom(world.getPlayer()));
        return false;
    }
}
```

```java
package uk.insrt.coursework.zuul.content.campaign;

import java.io.IOException;
import java.util.HashSet;
import java.util.stream.Collectors;

import uk.insrt.coursework.zuul.content.campaign.StoryFlags.Stage;
import uk.insrt.coursework.zuul.content.campaign.entities.EntityCat;
import uk.insrt.coursework.zuul.content.campaign.entities.EntityWithDialogue;
import uk.insrt.coursework.zuul.content.campaign.events.EventGameStageChanged;
import uk.insrt.coursework.zuul.content.campaign.rooms.RoomApartmentsHome;
import uk.insrt.coursework.zuul.content.campaign.rooms.RoomApartmentsReception;
import uk.insrt.coursework.zuul.content.campaign.rooms.RoomBackAlley;
import uk.insrt.coursework.zuul.content.campaign.rooms.RoomCityCentre;
import uk.insrt.coursework.zuul.content.campaign.rooms.RoomCoastline;
import uk.insrt.coursework.zuul.content.campaign.rooms.RoomForest;
import uk.insrt.coursework.zuul.content.campaign.rooms.RoomMainlandCoastline;
import uk.insrt.coursework.zuul.content.campaign.rooms.RoomMedicalCentreOffice;
import uk.insrt.coursework.zuul.content.campaign.rooms.RoomMedicalCentreReception;
import uk.insrt.coursework.zuul.content.campaign.rooms.RoomShop;
import uk.insrt.coursework.zuul.content.campaign.rooms.RoomStreet;
import uk.insrt.coursework.zuul.content.campaign.rooms.RoomWormHole;
import uk.insrt.coursework.zuul.dialogue.DialogueLoader;
import uk.insrt.coursework.zuul.entities.Entity;
import uk.insrt.coursework.zuul.entities.EntityPlayer;
import uk.insrt.coursework.zuul.events.IEventListener;
import uk.insrt.coursework.zuul.events.world.EventEntityEnteredRoom;
import uk.insrt.coursework.zuul.events.world.EventEntityLeftRoom;
import uk.insrt.coursework.zuul.io.Ansi;
import uk.insrt.coursework.zuul.io.IOSystem;
import uk.insrt.coursework.zuul.sound.EventMusic;
import uk.insrt.coursework.zuul.sound.MusicType;
import uk.insrt.coursework.zuul.world.Room;
import uk.insrt.coursework.zuul.world.World;

/**
 * The main campaign World.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.1-SNAPSHOT
 */
```

```java
public class CampaignWorld extends World {
    private StoryFlags flags;
    private HashSet<Room> visitedRooms;
    private DialogueLoader dialogueLoader;

    /**
     * Construct a new Campaign World
     * @param io Provided IO system
     */
    public CampaignWorld(IOSystem io) {
        super(io);

        this.visitedRooms = new HashSet<>();
        this.dialogueLoader = new DialogueLoader();
        this.flags = new StoryFlags(this.getEventSystem());

        try {
            this.dialogueLoader.load("/dialogue.toml");
        } catch (IOException e) {
            System.err.println("Failed to load resources for campaign world!");
            e.printStackTrace();
        }

        this.buildWorld();
        this.spawnEntities();
        this.registerEvents();
    }

    /**
     * Get this World's Dialogue Loader
     * @return Dialogue Loader
     */
    public DialogueLoader getDialogueLoader() {
        return this.dialogueLoader;
    }

    /**
     * Get the global story flags.
     * @return Story flags instance
     */
    public StoryFlags getStoryFlags() {
```

```java
 83            return this.flags;
 84        }
 85
 86        /**
 87         * Check whether the Player has visited a certain Room yet
 88         * @param room Room to check
 89         * @return True if the Player has visited the given Room
 90         */
 91        public boolean hasVisited(Room room) {
 92            return this.visitedRooms.contains(room);
 93        }
 94
 95        /**
 96         * Get a rounded whole number percentage of how much the World has been explored.
 97         * @return Integer representing percentage of World explored
 98         */
 99        public int percentVisited() {
100            return Math.round((float) this.visitedRooms.size() / this.rooms.size() * 100.0f);
101        }
102
103        /**
104         * Create all the Worlds and link adjacent Rooms together.
105         */
106        private void buildWorld() {
107            final Room[] rooms = {
108                new RoomCityCentre(this),
109                new RoomStreet(this),
110                new RoomShop(this),
111                new RoomBackAlley(this),
112                new RoomApartmentsReception(this),
113                new RoomApartmentsHome(this),
114                new RoomMedicalCentreReception(this),
115                new RoomMedicalCentreOffice(this),
116                new RoomCoastline(this),
117                new RoomMainlandCoastline(this),
118                new RoomForest(this),
119                new RoomWormHole(this)
120            };
121
122            for (Room room : rooms) {
123                this.addRoom(room);
```

```java
124            }
125
126            this.linkRooms();
127        }
128
129        /**
130         * Spawn and setup any Entities within this World.
131         */
132        private void spawnEntities() {
133            for (Room room : this.rooms.values()) {
134                room.spawnEntities();
135            }
136
137            // Entangle boat inventories.
138            Entity boat1 = this.entities.get("boat1");
139            Entity boat2 = this.entities.get("boat2");
140            boat1.entangleInventory(boat2.getInventory());
141        }
142
143        /**
144         * Register all the game logic.
145         */
146        private void registerEvents() {
147            // Capture all Events for Entities entering Rooms.
148            this.eventSystem.addListener(EventEntityEnteredRoom.class,
149                event -> {
150                    Entity entity = event.getEntity();
151                    if (entity instanceof EntityPlayer) {
152                        Room room = entity.getRoom();
153
154                        // Whenever the Player enters a Room, we should print the
155                        // description of the Room and list things found in the Room.
156                        this.io.println(
157                            room.describe()
158                                + "\n<global.can_go_in_x_directions.1> "
159                                + room.getDirections().size()
160                                + " <global.can_go_in_x_directions.2>: "
161                                + room.getDirections()
162                                    .stream()
163                                    .map(x ->
164                                        x.toString()
```

```java
                                    .toLowerCase()
                                    .replaceAll("_", " ")
                            )
                            .collect(Collectors.joining(", "))
                    );

                    // Mark current room as previously visited.
                    this.visitedRooms.add(room);

                    // When we enter a new room, list what we can see.
                    String entities = this.getEntitiesInRoom(entity.getRoom())
                        .stream()
                        .filter(e -> !(e instanceof EntityPlayer))
                        .map(e -> "- " + e.describe() + " ("
                            + Ansi.BackgroundPurple + Ansi.Black
                            + e.getName() + Ansi.Reset + ")")
                        .collect(Collectors.joining("\n"));

                    if (entities.length() > 0) {
                        this.io.println("<global.sight>\n" + entities);
                    }
                } else {
                    // If another entity enters the room,
                    // conditionally mention this to the player.
                    EntityPlayer player = this.getPlayer();
                    if (entity.getRoom() == player.getRoom()) {
                        if (entity instanceof EntityCat) {
                            this.io.println("\n<entities.cat.enter>");
                        }
                    }
                }
            });

        // Capture all Events for Entities leaving Rooms.
        this.eventSystem.addListener(EventEntityLeftRoom.class,
            event -> {
                Entity entity = event.getEntity();
                if (entity instanceof EntityPlayer) return;

                Room room = event.getRoom();
                if (room != this.player.getRoom()) return;
```

```java
206
207                    // If another entity leaves the room,
208                    // conditionally mention this to the player.
209                    if (entity instanceof EntityCat) {
210                        this.io.println("\n<entities.cat.leave>");
211                    }
212                });
213
214        // Register event for game stage changing.
215        this.eventSystem.addListener(EventGameStageChanged.class,
216            event -> {
217                Stage stage = event.getStage();
218                switch (stage) {
219                    case Recon: {
220                        for (Entity entity : this.entities.values()) {
221                            if (entity instanceof EntityWithDialogue) {
222                                ((EntityWithDialogue<?>) entity).setDialogueNodeIfPresent("recon");
223                            }
224                        }
225                        break;
226                    }
227                    case End: {
228                        io.println("<stage.reached_conclusion>");
229                        break;
230                    }
231                    default: break;
232                }
233            });
234
235        // Register required Events for Worm Hole room to function.
236        @SuppressWarnings("unchecked")
237        var wh = (IEventListener<EventEntityEnteredRoom>) this.getRoom("Worm Hole");
238        this.eventSystem.addListener(EventEntityEnteredRoom.class, wh);
239
240        // Register required Events for the protestors to disappear.
241        @SuppressWarnings("unchecked")
242        var st = (IEventListener<EventGameStageChanged>) this.getRoom("Street");
243        this.eventSystem.addListener(EventGameStageChanged.class, st);
244
245        // Play BGM when player enters room.
246        this.eventSystem.addListener(EventEntityEnteredRoom.class,
```

```java
                    event -> {
                        Entity entity = event.getEntity();
                        if (entity == this.getPlayer()) {
                            Room room = entity.getRoom();
                            MusicType type = null;
                            if (room instanceof RoomCoastline) {
                                type = MusicType.Bay1;
                            } else if (room instanceof RoomMainlandCoastline) {
                                type = MusicType.Bay2;
                            } else if (room instanceof RoomForest) {
                                type = MusicType.Nature;
                            } else if (room instanceof RoomCityCentre) {
                                type = MusicType.City1;
                            } else if (room instanceof RoomStreet) {
                                type = MusicType.City2;
                            }

                            if (type != null) {
                                this.getEventSystem()
                                    .emit(new EventMusic(type, true));
                            }
                        }
                    });

            // Stop prevous BGM when player leaves room.
            this.eventSystem.addListener(EventEntityLeftRoom.class,
                event -> {
                    if (event.getEntity() == this.getPlayer()) {
                        Room room = event.getRoom();
                        MusicType type = null;
                        if (room instanceof RoomCoastline) {
                            type = MusicType.Bay1;
                        } else if (room instanceof RoomMainlandCoastline) {
                            type = MusicType.Bay2;
                        } else if (room instanceof RoomForest) {
                            type = MusicType.Nature;
                        } else if (room instanceof RoomCityCentre) {
                            type = MusicType.City1;
                        } else if (room instanceof RoomStreet) {
                            type = MusicType.City2;
                        }
```

```java
288
289                    if (type != null) {
290                        this.eventSystem
291                            .emit(new EventMusic(type, false));
292                    }
293                }
294            });
295
296        // Adaptive BGM depending on the part of the story.
297        this.eventSystem.addListener(EventGameStageChanged.class,
298            event -> {
299                switch (event.getStage()) {
300                    case Stealth: {
301                        this.eventSystem
302                            .emit(new EventMusic(MusicType.BgmExplore, false));
303                        this.eventSystem
304                            .emit(new EventMusic(MusicType.BgmMission, true));
305                        break;
306                    }
307                    case End: {
308                        this.eventSystem
309                            .emit(new EventMusic(MusicType.BgmMission, false));
310                        this.eventSystem
311                            .emit(new EventMusic(MusicType.BgmConclusion, true));
312                        break;
313                    }
314                    default: break;
315                }
316            });
317    }
318
319    @Override
320    public void spawnPlayer() {
321        this.player.setLocation(this.rooms.get("Apartments: Home"));
322    }
323 }
```

```java
package uk.insrt.coursework.zuul.content.campaign.commands;

import java.awt.Image;
import java.io.InputStream;
import java.util.regex.Pattern;

import javax.imageio.ImageIO;

import uk.insrt.coursework.zuul.commands.Arguments;
import uk.insrt.coursework.zuul.commands.Command;
import uk.insrt.coursework.zuul.content.campaign.CampaignWorld;
import uk.insrt.coursework.zuul.events.IEventListener;
import uk.insrt.coursework.zuul.io.IOSystem;
import uk.insrt.coursework.zuul.ui.EventDraw;
import uk.insrt.coursework.zuul.world.World;

/**
 * Command available for the terminal emulator which displays a graphical map.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class CommandMap extends Command implements IEventListener<EventDraw> {
    private boolean visible;
    private Image image;

    public CommandMap() {
        super("map", "<commands.map.usage>",
            new Pattern[] {
                Pattern.compile("^(?:m(?:ap)*)(?!\\w)")
                // m, map
            });

        this.visible = false;

        try {
            InputStream stream = this.getClass().getResourceAsStream("/map/base.png");
            this.image = ImageIO.read(stream);
        } catch (Exception e) {}
    }
```

```java
42      @Override
43      public boolean run(World world, Arguments arguments) {
44          CampaignWorld campaignWorld = (CampaignWorld) world;
45          IOSystem io = world.getIO();
46          io.print("<commands.map.discovered.1> " + campaignWorld.percentVisited()
47              + "% <commands.map.discovered.2>!" + "\n".repeat(24) + "<commands.map.close>");
48
49          // We make the map visible and block on user input,
50          // Once the user interacts, the map is hidden again.
51          this.visible = true;
52          io.readLine();
53          this.visible = false;
54          return false;
55      }
56
57      @Override
58      public void onEvent(EventDraw event) {
59          if (!this.visible) return;
60
61          // We are drawing the map from [0,1] to [80,24].
62          float fw = event.getCharWidth();
63          float fh = event.getCharHeight();
64
65          var g = event.getGraphics();
66          g.drawImage(
67              this.image,
68              Math.round(event.getOriginX()),
69              Math.round(event.getOriginY() + fh),
70              Math.round(fw * 80),
71              Math.round(fh * 23),
72              null
73          );
74      }
75  }
```

```java
package uk.insrt.coursework.zuul.content.campaign.commands;

import java.util.regex.Pattern;

import uk.insrt.coursework.zuul.commands.Arguments;
import uk.insrt.coursework.zuul.commands.Command;
import uk.insrt.coursework.zuul.content.campaign.CampaignWorld;
import uk.insrt.coursework.zuul.content.campaign.StoryFlags.Stage;
import uk.insrt.coursework.zuul.world.World;

/**
 * This command is unlocked after the player completes the final mission.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class CommandWin extends Command {
    public CommandWin() {
        super("win", "<commands.win.usage>",
            new Pattern[] {
                Pattern.compile("^win(?!\\w)"),
                // win
            });
    }

    @Override
    public boolean run(World world, Arguments args) {
        var io = world.getIO();
        var w = (CampaignWorld) world;
        var flags = w.getStoryFlags();
        if (flags.getStage() != Stage.End) return false;

        io.println("<commands.win.conclusion>\n<commands.win.stats>\n"
            + "<commands.win.total_ticks>" + flags.getTicks() + "\n"
            + "<commands.win.sidequests_complete>"
            + flags.getCompletedQuests() + " / " + flags.getTotalQuests()
            + "\n\n<commands.win.press_enter_key>");

        io.readLine();
        return true;
    }
```

```java
42
43      @Override
44      public boolean isVisible() {
45          return false;
46      }
47  }
```

```java
package uk.insrt.coursework.zuul.content.campaign.entities;

import uk.insrt.coursework.zuul.entities.Entity;
import uk.insrt.coursework.zuul.entities.EntityObject;
import uk.insrt.coursework.zuul.entities.actions.IUseable;
import uk.insrt.coursework.zuul.events.world.EventTick;
import uk.insrt.coursework.zuul.world.Location;
import uk.insrt.coursework.zuul.world.World;

/**
 * Bed entity which lets the player tick the World forwards.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class EntityBed extends EntityObject implements IUseable {
    public EntityBed(World world, Location location) {
        super(world, location, 80, new String[] { "bed" }, "<entities.bed.description>");
    }

    public void use(Entity target) {
        // We emit EventTick an arbitrary amount of times to
        // in-effect push the time forwards. This will trigger
        // all random events which listen to this event.
        for (int i=0;i<20;i++) {
            world.emit(new EventTick());
        }

        this.world.getIO().println("<entities.bed.use>");
    }
}
```

```java
package uk.insrt.coursework.zuul.content.campaign.entities;

import uk.insrt.coursework.zuul.content.campaign.CampaignWorld;
import uk.insrt.coursework.zuul.content.campaign.StoryFlags.Stage;
import uk.insrt.coursework.zuul.entities.Entity;
import uk.insrt.coursework.zuul.entities.Inventory;
import uk.insrt.coursework.zuul.entities.actions.IGiveable;
import uk.insrt.coursework.zuul.entities.actions.IUseable;
import uk.insrt.coursework.zuul.io.IOSystem;
import uk.insrt.coursework.zuul.world.Location;
import uk.insrt.coursework.zuul.world.Room;
import uk.insrt.coursework.zuul.world.World;

/**
 * Boat entity which ferries the Player to an arbitrary location.
 * There is no restriction on location but they should be place as
 * appropriate and where it would be realistic to put a boat.
 *
 * Boats may not be operated by the player while they are carrying
 * anything so instead they must use the boat's storage.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class EntityBoat extends Entity implements IUseable, IGiveable {
    private Room destination;

    public EntityBoat(World world, Location location, Room destination) {
        super(world, location, 200);
        this.destination = destination;
        this.inventory.setMaxWeight(100);
    }

    @Override
    public String[] getAliases() {
        return new String[] { "boat" };
    }

    @Override
    public String describe() {
        return "<entities.boat.description>";
```

```java
42         }
43
44         @Override
45         public void use(Entity target) {
46             CampaignWorld world = (CampaignWorld) this.getWorld();
47             IOSystem io = world.getIO();
48             Inventory inventory = target.getInventory();
49
50             // Check if the player has the key to this boat.
51             boolean hasKey = false;
52             for (Entity item : inventory.getItems()) {
53                 if (item instanceof EntityBoatKey) {
54                     hasKey = true;
55                 }
56             }
57
58             if (!hasKey) {
59                 if (world.getStoryFlags().getStage() == Stage.Exposition) {
60                     io.println("<entities.boat.locked>");
61                 } else {
62                     io.println("<entities.boat.locked_for_sale>");
63                 }
64
65                 return;
66             }
67
68             // Check whether the player is carrying too much.
69             if (inventory.getWeight() > 1) {
70                 io.println("<entities.boat.denied>");
71                 return;
72             }
73
74             // If we're good to go, travel to the other side.
75             io.println("<entities.boat.travel>\n");
76             target.setLocation(this.destination);
77         }
78
79         @Override
80         public void give(Entity item) {
81             var io = this.getWorld().getIO();
82             if (item.setLocation(this.getInventory())) {
```

```
83                io.println("<entities.boat.give.1> " + item.getHighlightedName() + " <entities.boat.give.2>.");
84            } else {
85                io.println("<entities.boat.too_heavy>");
86            }
87        }
88    }
```

```java
package uk.insrt.coursework.zuul.content.campaign.entities;

import uk.insrt.coursework.zuul.entities.EntityObject;
import uk.insrt.coursework.zuul.world.Location;
import uk.insrt.coursework.zuul.world.World;

/**
 * Boat key object which is used to unlock and start the
 * speed boat present on the coast.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class EntityBoatKey extends EntityObject {
    public EntityBoatKey(World world, Location location) {
        super(world, location, 0.01d,
            new String[] { "key" },
            "<entities.boat_key>");
    }
}
```

```java
package uk.insrt.coursework.zuul.content.campaign.entities;

import uk.insrt.coursework.zuul.entities.Entity;
import uk.insrt.coursework.zuul.entities.actions.IPettable;
import uk.insrt.coursework.zuul.entities.actions.IUseable;
import uk.insrt.coursework.zuul.events.world.EventTick;
import uk.insrt.coursework.zuul.events.world.behaviours.SimpleWanderAI;
import uk.insrt.coursework.zuul.world.Location;
import uk.insrt.coursework.zuul.world.Room;
import uk.insrt.coursework.zuul.world.World;

/**
 * Cat entity which wanders around the map.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class EntityCat extends Entity implements IPettable, IUseable {
    public static final int WEIGHT = 5;

    public EntityCat(World world, Location startingLocation) {
        super(world, startingLocation, WEIGHT);
    }

    @Override
    public String[] getAliases() {
        return new String[] {
            "cat",
            "the cat"
        };
    }

    @Override
    public String describe() {
        return "<entities.cat.description>";
    }

    @Override
    public void pet() {
        this.getWorld().getIO().println("<entities.cat.pet>");
    }
```

```java
42
43      @Override
44      public void use(Entity target) {
45          this.getWorld().getIO().println("<entities.cat.use>");
46      }
47
48      /**
49       * Enable a simple wander behaviour for entity within given bounds.
50       * @param rooms Path that this Entity should follow
51       * @param chance The chance x that this entity moves, where x gives a 1/x fractional chance of moving.
52       */
53      public void useWanderAI(Room[] rooms, int chance) {
54          this.getWorld()
55              .getEventSystem()
56              .addListener(EventTick.class, new SimpleWanderAI(this, rooms, chance));
57      }
58  }
```

```java
package uk.insrt.coursework.zuul.content.campaign.entities;

import uk.insrt.coursework.zuul.content.campaign.CampaignWorld;
import uk.insrt.coursework.zuul.content.campaign.StoryFlags.Stage;
import uk.insrt.coursework.zuul.content.campaign.rooms.RoomMedicalCentreOffice;
import uk.insrt.coursework.zuul.content.campaign.rooms.RoomMedicalCentreReception;
import uk.insrt.coursework.zuul.dialogue.Dialogue;
import uk.insrt.coursework.zuul.dialogue.DialogueOption;
import uk.insrt.coursework.zuul.entities.Entity;
import uk.insrt.coursework.zuul.entities.actions.IUseable;
import uk.insrt.coursework.zuul.world.Location;
import uk.insrt.coursework.zuul.world.Room;
import uk.insrt.coursework.zuul.world.World;

/**
 * Comms entity which is used to communicate between
 * Marie and the player during the mission.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class EntityComms extends EntityWithDialogue<String> implements IUseable {
    public EntityComms(World world, Location location) {
        super(world, location, 0.3d);
        this.setupDialogue();
    }

    @Override
    public void use(Entity target) {
        var w = (CampaignWorld) this.world;
        var io = w.getIO();

        if (w.getStoryFlags().getStage() == Stage.Stealth) {
            Room room = w.getPlayer().getRoom();
            if (room instanceof RoomMedicalCentreOffice) {
                this.dialogue.setNodeIfPresent("office");
            } else if (room instanceof RoomMedicalCentreReception) {
                if (((RoomMedicalCentreReception) room).getCouch().isSitting()) {
                    this.dialogue.setNodeIfPresent("in_position");
                } else {
                    this.dialogue.setNodeIfPresent("complaint");
```

```java
42                  }
43              } else {
44                  this.dialogue.setNodeIfPresent("orientation");
45              }
46
47              this.dialogue.run(io);
48          } else {
49              io.println("<entities.comms.off>");
50          }
51      }
52
53      @Override
54      public void setupDialogue(Dialogue<String> dialogue) {
55          this.setupDialogueFromId(dialogue, "comms_marie");
56
57          // Make the guards disappear at this point.
58          dialogue.getPart("distraction")
59              .addOption(new DialogueOption<>(
60                  "<marie.comms.distraction.option_1>",
61                  io -> {
62                      ((RoomMedicalCentreReception) this.world.getRoom("Medical Centre: Reception"))
63                          .getGuard()
64                          .consume(false);
65
66                      return "coast_is_clear";
67                  }
68          ));
69      }
70
71      @Override
72      public String[] getAliases() {
73          return new String[] { "comms" };
74      }
75
76      @Override
77      public String describe() {
78          return "<entities.comms.description>";
79      }
80  }
```

```java
package uk.insrt.coursework.zuul.content.campaign.entities;

import uk.insrt.coursework.zuul.entities.Entity;
import uk.insrt.coursework.zuul.entities.EntityObject;
import uk.insrt.coursework.zuul.entities.actions.IUseable;
import uk.insrt.coursework.zuul.events.IEventListener;
import uk.insrt.coursework.zuul.events.world.EventEntityLeftRoom;
import uk.insrt.coursework.zuul.world.Location;
import uk.insrt.coursework.zuul.world.World;

/**
 * Couch present in the Medical Centre reception area.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class EntityCouch extends EntityObject implements IUseable, IEventListener<EventEntityLeftRoom> {
    private boolean isSitting;

    /**
     * Construct a new EntityCouch.
     * @param world World
     * @param location Location
     */
    public EntityCouch(World world, Location location) {
        super(world, location, Double.MAX_VALUE, "couch", "<entities.couch.description>");
    }

    @Override
    public void use(Entity target) {
        var io = this.getWorld().getIO();
        if (this.isSitting) {
            io.println("<entities.couch.sitting>");
        } else {
            io.println("<entities.couch.sit>");
            this.isSitting = true;
        }
    }

    @Override
    public void onEvent(EventEntityLeftRoom event) {
```

```java
42              this.isSitting = false;
43          }
44
45          /**
46           * Whether the player is sitting on the couch.
47           * @return True if the player is sat down
48           */
49          public boolean isSitting() {
50              return this.isSitting;
51          }
52      }
```

```java
package uk.insrt.coursework.zuul.content.campaign.entities;

import uk.insrt.coursework.zuul.entities.Entity;
import uk.insrt.coursework.zuul.entities.EntityObject;
import uk.insrt.coursework.zuul.entities.actions.IUseable;
import uk.insrt.coursework.zuul.world.Location;
import uk.insrt.coursework.zuul.world.World;

/**
 * Documents which the player needs to find and take.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class EntityDocument extends EntityObject implements IUseable {
    private int count;

    /**
     * Construct a new EntityDocument
     * @param world World
     * @param location Location
     * @param count Document Id
     */
    public EntityDocument(World world, Location location, int count) {
        super(world, location, 10, "doc" + count, "<medical_centre_office.books." + count + ".title>");
        this.count = count;
    }

    @Override
    public void use(Entity target) {
        this.getWorld()
            .getIO()
            .println("<medical_centre_office.books." + count + ".contents>");
    }

    /**
     * Check whether these are the documents we want.
     * @return Whether we want this document
     */
    public boolean getIsValid() {
        switch (this.count) {
```

```java
                case 2:
                case 4:
                case 5: return true;
        }

        return false;
    }
}
```

```java
package uk.insrt.coursework.zuul.content.campaign.entities;

import java.awt.Desktop;
import java.net.URI;

import uk.insrt.coursework.zuul.content.campaign.CampaignWorld;
import uk.insrt.coursework.zuul.content.campaign.StoryFlags.Stage;
import uk.insrt.coursework.zuul.dialogue.Dialogue;
import uk.insrt.coursework.zuul.dialogue.DialogueNode;
import uk.insrt.coursework.zuul.dialogue.DialogueOption;
import uk.insrt.coursework.zuul.entities.Entity;
import uk.insrt.coursework.zuul.entities.Inventory;
import uk.insrt.coursework.zuul.entities.actions.IUseable;
import uk.insrt.coursework.zuul.world.Location;
import uk.insrt.coursework.zuul.world.World;

/**
 * This is the player's laptop which resides in their home.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class EntityLaptop extends EntityWithDialogue<String> implements IUseable {
    /**
     * Construct a new EntityLaptop.
     * @param world World
     * @param location Location
     */
    public EntityLaptop(World world, Location location) {
        super(world, location, 2);
        this.setupDialogue();
    }

    @Override
    public void use(Entity target) {
        this.dialogue.run(this.getWorld().getIO());
    }

    @Override
    public void setupDialogue(Dialogue<String> dialogue) {
        this.setupDialogueFromId(dialogue, "entity_laptop");
```

```java
42
43            // Add funny cat videos.
44            this.dialogue.addPart("funny_cat_videos",
45                new DialogueNode<String>("<entities.laptop.cat_videos.dialog>")
46                    .addOption(new DialogueOption<String>("<entities.laptop.cat_videos.option_q>",
47                        io -> {
48                            try {
49                                Desktop.getDesktop().browse(new URI("https://youtu.be/k35aiQPgNI4"));
50                            } catch (Exception e) { /* If we fail, just ignore this ever happen. */ }
51
52                            return "home";
53                        })));
54
55            // Story handler for sending documents to Marie.
56            this.dialogue.getPart("document")
57                .addOption(new DialogueOption<String>("<entities.laptop.document.option_1>",
58                    io -> {
59                        var w = (CampaignWorld) this.getWorld();
60                        var flags = w.getStoryFlags();
61                        if (flags.getStage() != Stage.Stealth) {
62                            io.println("<marie.comms.no_access>");
63                            return "document";
64                        }
65
66                        Inventory inv = w.getPlayer().getInventory();
67                        int validPieces = 0;
68                        for (Entity item : inv.getItems()) {
69                            if (item instanceof EntityDocument) {
70                                if (((EntityDocument) item).getIsValid()) {
71                                    validPieces++;
72                                }
73                            }
74                        }
75
76                        if (validPieces == 3) {
77                            io.println("<marie.comms.received>");
78                            flags.setStage(Stage.End);
79                            return null;
80                        }
81
82                        io.println("<marie.comms.bad_documents>");
```

```java
                    return "document";
                }));
    }

    @Override
    public String[] getAliases() {
        return new String[] { "laptop" };
    }

    @Override
    public String describe() {
        return "<entities.laptop.description>";
    }
}
```

```java
package uk.insrt.coursework.zuul.content.campaign.entities;

import uk.insrt.coursework.zuul.content.campaign.CampaignWorld;
import uk.insrt.coursework.zuul.content.campaign.StoryFlags.Stage;
import uk.insrt.coursework.zuul.dialogue.Dialogue;
import uk.insrt.coursework.zuul.dialogue.DialogueOption;
import uk.insrt.coursework.zuul.entities.Entity;
import uk.insrt.coursework.zuul.entities.Inventory;
import uk.insrt.coursework.zuul.world.Location;
import uk.insrt.coursework.zuul.world.World;

/**
 * Marie Itami
 * https://brand-new-animal.fandom.com/wiki/Marie_Itami
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class EntityMarie extends EntityNpc {
    /**
     * Construct a new EntityMarie.
     * @param world World
     * @param location Location
     */
    public EntityMarie(World world, Location location) {
        super(world, location,
            "npc_marie",
            "<marie.description>",
            new String[] { "marie", "itami", "mink" });
    }

    @Override
    public void setupDialogue(Dialogue<String> dialogue) {
        super.setupDialogue(dialogue);
        var w = (CampaignWorld) this.world;

        // Progress story if player accepts mission.
        dialogue.getPart("confirm")
            .addOption(new DialogueOption<>("<marie.alley.confirm.option_1>",
                io -> {
                    w.getStoryFlags()
```

```java
                    .setStage(Stage.Recon);

                    return "recon";
                }));
    }

    @Override
    public void talk() {
        if (this.dialogue.getCurrentNode().equals("waiting")) {
            var w = ((CampaignWorld) this.getWorld());
            Inventory inv = w.getPlayer().getInventory();
            for (Entity item : inv.getItems()) {
                if (item instanceof EntityComms) {
                    this.dialogue.setNodeIfPresent("mission_brief");
                    w.getStoryFlags().setStage(Stage.Stealth);
                }
            }
        }

        super.talk();
    }
}
```

```java
package uk.insrt.coursework.zuul.content.campaign.entities;

import uk.insrt.coursework.zuul.dialogue.Dialogue;
import uk.insrt.coursework.zuul.entities.actions.ITalkwith;
import uk.insrt.coursework.zuul.world.Location;
import uk.insrt.coursework.zuul.world.World;

/**
 * NPC entity which provides dialog and can be talked with by the Player.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class EntityNpc extends EntityWithDialogue<String> implements ITalkwith {
    private String description;
    private String alias[];
    private String id;

    public EntityNpc(World world, Location location, String id, String description, String alias[]) {
        super(world, location, 75, null);

        this.description = description;
        this.alias = alias;
        this.id = id;

        this.setupDialogue();
    }

    public void talk() {
        this.dialogue.run(this.getWorld().getIO());
    }

    @Override
    public String[] getAliases() {
        return this.alias;
    }

    @Override
    public String describe() {
        return this.description;
    }
```

```java
42
43        @Override
44        public void setupDialogue(Dialogue<String> dialogue) {
45            this.setupDialogueFromId(dialogue, id);
46        }
47    }
```

```java
package uk.insrt.coursework.zuul.content.campaign.entities;

import uk.insrt.coursework.zuul.content.campaign.CampaignWorld;
import uk.insrt.coursework.zuul.content.campaign.StoryFlags.Quest;
import uk.insrt.coursework.zuul.entities.Entity;
import uk.insrt.coursework.zuul.entities.actions.IGiveable;
import uk.insrt.coursework.zuul.world.Location;
import uk.insrt.coursework.zuul.world.World;

/**
 * The old man who is in the forest.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class EntityOldMan extends EntityNpc implements IGiveable {
    /**
     * Construct a new EntityOldMan.
     * @param world World
     * @param location Location
     */
    public EntityOldMan(World world, Location location) {
        super(world, location, "npc_old_man",
            "<forest.old_man.description>",
            new String[] { "oldman", "man" });

        this.inventory.setMaxWeight(EntityCat.WEIGHT);
    }

    @Override
    public void give(Entity item) {
        var io = this.getWorld().getIO();
        if (this.inventory.isFull()) {
            io.println("<forest.old_man.full>");
            return;
        }

        if (item instanceof EntityCat) {
            item.setLocation(this.inventory);
            this.dialogue.setNodeIfPresent("praise");
            ((CampaignWorld) this.getWorld())
```

```
42                    .getStoryFlags()
43                    .completeSideQuest(Quest.Cat);
44
45            io.println("<forest.old_man.accept>");
46        } else {
47            io.println("<forest.old_man.deny> " + item.getHighlightedName());
48        }
49    }
50 }
```

```java
package uk.insrt.coursework.zuul.content.campaign.entities;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

import uk.insrt.coursework.zuul.content.campaign.CampaignWorld;
import uk.insrt.coursework.zuul.content.campaign.StoryFlags.Stage;
import uk.insrt.coursework.zuul.dialogue.Dialogue;
import uk.insrt.coursework.zuul.dialogue.DialogueNode;
import uk.insrt.coursework.zuul.dialogue.DialogueOption;
import uk.insrt.coursework.zuul.entities.Entity;
import uk.insrt.coursework.zuul.entities.EntityObject;
import uk.insrt.coursework.zuul.entities.Inventory;
import uk.insrt.coursework.zuul.io.Ansi;
import uk.insrt.coursework.zuul.sound.EventSound;
import uk.insrt.coursework.zuul.sound.SoundType;
import uk.insrt.coursework.zuul.world.Location;
import uk.insrt.coursework.zuul.world.World;

/**
 * Shop keeper which the player can buy items from in the town.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class EntityShopkeeper extends EntityNpc {
    private HashMap<Stage, Entity[]> items;
    private HashMap<Entity, Integer> stock;
    private HashMap<Entity, Integer> price;
    private HashMap<Entity, IEntityFactory> entityFactory;

    /**
     * Construct a new EntityShopkeeper.
     * @param world World
     * @param location Location
     */
    public EntityShopkeeper(World world, Location location) {
        super(world, location,
            "npc_shopkeeper",
            "<shop.npc.description>",
```

```java
42                  new String[] { "shopkeeper", "shop", "keeper" });
43
44          this.items = new HashMap<>();
45          this.stock = new HashMap<>();
46          this.price = new HashMap<>();
47          this.entityFactory = new HashMap<>();
48
49          this.createItems(world);
50      }
51
52      /**
53       * Interface implemented by entity factories for producing entities.
54       */
55      private interface IEntityFactory {
56          /**
57           * Produce a new Entity of a certain type.
58           * @return New entity
59           */
60          public Entity produce();
61      }
62
63      /**
64       * Generate all the items and populate the data.
65       * @param world World to place items in
66       */
67      private void createItems(World world) {
68          EntityObject itemBoatKey = new EntityBoatKey(world, new Location());
69          this.stock.put(itemBoatKey, 1);
70          this.price.put(itemBoatKey, 39_260);
71          this.entityFactory.put(itemBoatKey, () -> new EntityBoatKey(world, new Location()));
72
73          EntityComms itemComms = new EntityComms(world, new Location());
74          this.stock.put(itemComms, 3);
75          this.price.put(itemComms, 3_100);
76          this.entityFactory.put(itemComms, () -> new EntityComms(world, new Location()));
77
78          EntityObject itemCat = new EntityObject(world, new Location(), 5, "", "<shop.npc.fake_item.cat>");
79          this.stock.put(itemCat, 0);
80          this.price.put(itemCat, 21_300);
81
82          this.items.put(Stage.Exposition, new Entity[] {  });
```

```java
 83            this.items.put(Stage.Recon, new Entity[] { itemBoatKey, itemComms });
 84            this.items.put(Stage.Stealth, new Entity[] { itemBoatKey, itemComms, itemCat });
 85            this.items.put(Stage.End, new Entity[] { itemBoatKey, itemCat });
 86        }
 87
 88        /**
 89         * Index dialogue node, displays things that the player can buy.
 90         * Implicitly takes the context of the EntityShopkeeper.
 91         */
 92        private class IndexNode extends DialogueNode<String> {
 93            /**
 94             * Construct a new IndexNode.
 95             * We may ignore the description since we override {@code #getDescription}.
 96             */
 97            public IndexNode() {
 98                super(null);
 99            }
100
101            @Override
102            public String getDescription() {
103                var w = (CampaignWorld) world;
104                return "<shop.npc.greeting."
105                    + w.getStoryFlags().getStage().toString()
106                    + ">\n"
107                    + "<shop.npc.currently_have_amount_of_money> ¥ "
108                    + w.getStoryFlags().getBalance()
109                    + "\n";
110            }
111
112            @Override
113            protected List<DialogueOption<String>> getOptions() {
114                ArrayList<DialogueOption<String>> options = new ArrayList<>();
115
116                var w = (CampaignWorld) world;
117                var flags = w.getStoryFlags();
118                var player = w.getPlayer();
119
120                // Get all the items we can access at this story stage.
121                Entity[] list = items.get(flags.getStage());
122                for (Entity item : list) {
123                    int count = stock.get(item);
```

```java
124                    int cost = price.get(item);
125                    IEntityFactory factory = entityFactory.get(item);
126
127                    // Add the option for this item.
128                    options.add(new DialogueOption<String>(
129                        item.describe()
130                            + " ["
131                            + Ansi.Yellow
132                            + item.getWeight()
133                            + " kg"
134                            + Ansi.Reset
135                            + "] (¥ "
136                            + Ansi.Green
137                            + cost
138                            + Ansi.Reset
139                            + ") - "
140                            + (count == 0 ?
141                                "<shop.npc.out_of_stock>!"
142                              : count + " <shop.npc.x_left>"),
143                        io -> {
144                            // Check that this item is in stock.
145                            if (count == 0) {
146                                io.println("\n\n<shop.npc.item_out_of_stock.1> "
147                                    + Ansi.Red
148                                    + "<shop.npc.out_of_stock>"
149                                    + Ansi.Reset
150                                    + ", <shop.npc.item_out_of_stock.2>!");
151                            } else {
152                                // Make sure the player can hold this item without
153                                // going over their inventory weight limit.
154                                Inventory inv = player.getInventory();
155                                if (inv.getWeight() + item.getWeight() > inv.getMaxWeight()) {
156                                    io.println("\n\n<shop.npc.too_heavy>");
157                                } else {
158                                    // Try to deduct money from the player's money.
159                                    if (flags.deductFromBalance(cost)) {
160                                        io.println("\n\n<shop.npc.bought.1> "
161                                            + item.getHighlightedName()
162                                            + " <shop.npc.bought.2>!");
163
164                                        stock.put(item, count - 1);
```

```java
                                Entity entity = factory.produce();
                                entity.setLocation(inv);

                                world.emit(new EventSound(SoundType.MoneyBag));
                            } else {
                                io.println("\n\n<shop.npc.not_enough> "
                                    + item.getHighlightedName() + "!");
                            }
                        }
                    }

                    return "index";
                }
            ));
        }

        options.add(new DialogueOption<String>("<shop.npc.leave>", "index").mustExit());
        return options;
    }
}

    @Override
    public void setupDialogue(Dialogue<String> dialogue) {
        dialogue.addPart("index", new IndexNode());
        dialogue.setNodeIfPresent("index");
    }
}
```

```java
package uk.insrt.coursework.zuul.content.campaign.entities;

import uk.insrt.coursework.zuul.dialogue.Dialogue;
import uk.insrt.coursework.zuul.entities.Entity;
import uk.insrt.coursework.zuul.entities.actions.IUseable;
import uk.insrt.coursework.zuul.world.Location;
import uk.insrt.coursework.zuul.world.World;

/**
 * TV entity present in the player's room which they interact
 * with at the start of the game to learn more about the world.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class EntityTV extends EntityWithDialogue<String> implements IUseable {
    public EntityTV(World world, Location location) {
        super(world, location, 40);
        this.setupDialogue();
    }

    @Override
    public void use(Entity target) {
        this.dialogue.run(this.getWorld().getIO());
    }

    @Override
    public void setupDialogue(Dialogue<String> dialogue) {
        this.setupDialogueFromId(dialogue, "home_tv");
    }

    @Override
    public String[] getAliases() {
        return new String[] { "tv", "television" };
    }

    @Override
    public String describe() {
        return "<home.tv.description>";
    }
}
```

```java
package uk.insrt.coursework.zuul.content.campaign.entities;

import uk.insrt.coursework.zuul.content.campaign.CampaignWorld;
import uk.insrt.coursework.zuul.dialogue.Dialogue;
import uk.insrt.coursework.zuul.entities.Entity;
import uk.insrt.coursework.zuul.world.Location;
import uk.insrt.coursework.zuul.world.World;

/**
 * Abstract implementation of an entity which provides some sort of dialogue.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public abstract class EntityWithDialogue<T> extends Entity {
    protected Dialogue<T> dialogue;

    /**
     * Construct a new EntityWithDialogue with a starting dialogue node
     * @param world Current World object
     * @param location Initial Location of this Entity
     * @param weight The weight (in kg) of this Entity
     * @param startNode The starting dialogue node
     */
    public EntityWithDialogue(World world, Location location, double weight, T startNode) {
        super(world, location, weight);

        Dialogue<T> dialogue = new Dialogue<T>(startNode);
        this.dialogue = dialogue;
    }

    /**
     * Construct a new EntityWithDialogue without a starting dialogue node
     * @param world Current World object
     * @param location Initial Location of this Entity
     * @param weight The weight (in kg) of this Entity
     */
    public EntityWithDialogue(World world, Location location, double weight) {
        this(world, location, weight, null);
    }
```

```java
42        /**
43         * Configure this Entity's dialogue,
44         * create nodes and options to add to this Entity.
45         * @param dialogue Entity Dialogue
46         */
47        public abstract void setupDialogue(Dialogue<T> dialogue);
48
49        /**
50         * Configure dialogue.
51         */
52        public void setupDialogue() {
53            this.setupDialogue(this.dialogue);
54        }
55
56        /**
57         * Use the CampaignWorld's DialogueLoader to populate this Entity's Dialogue
58         * @param dialogue Entity Dialogue
59         * @param id Target dialogue ID in file
60         */
61        public void setupDialogueFromId(Dialogue<String> dialogue, String id) {
62            var world = (CampaignWorld) this.getWorld();
63            world.getDialogueLoader().populate(dialogue, id);
64        }
65
66        /**
67         * Set the current dialogue node if the given node is present.
68         * @param node Target node
69         */
70        public void setDialogueNodeIfPresent(Object node) {
71            try {
72                @SuppressWarnings("unchecked")
73                T n = (T) node;
74
75                this.dialogue.setNodeIfPresent(n);
76            } catch (ClassCastException ex) {
77                // Ignore the error since if we can't cast it
78                // to whatever type this is, then obviously this
79                // node is not present within this Dialogue.
80            }
81        }
82    }
```

```java
package uk.insrt.coursework.zuul.content.campaign.events;

import uk.insrt.coursework.zuul.content.campaign.StoryFlags.Stage;
import uk.insrt.coursework.zuul.events.Event;

/**
 * Event fired when the story stage (chapter) changes.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class EventGameStageChanged extends Event {
    private Stage stage;

    /**
     * Construct a new GameStageChanged event.
     * @param stage New stage
     */
    public EventGameStageChanged(Stage stage) {
        this.stage = stage;
    }

    /**
     * Get the new game stage.
     * @return Stage
     */
    public Stage getStage() {
        return this.stage;
    }
}
```

```java
package uk.insrt.coursework.zuul.content.campaign.rooms;

import uk.insrt.coursework.zuul.content.campaign.CampaignWorld;
import uk.insrt.coursework.zuul.world.Room;
import uk.insrt.coursework.zuul.world.World;

/**
 * Class which overrides getWorld to instead provide the CampaignWorld.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public abstract class CampaignRoom extends Room {
    /**
     * Construct a new CampaignRoom.
     * @param world World
     * @param name Internal name used to refer to this Room
     */
    public CampaignRoom(World world, String name) {
        super(world, name);
    }

    @Override
    public CampaignWorld getWorld() {
        return (CampaignWorld) super.getWorld();
    }
}
```

```java
package uk.insrt.coursework.zuul.content.campaign.rooms;

import uk.insrt.coursework.zuul.content.campaign.entities.EntityBed;
import uk.insrt.coursework.zuul.content.campaign.entities.EntityLaptop;
import uk.insrt.coursework.zuul.content.campaign.entities.EntityTV;
import uk.insrt.coursework.zuul.world.Direction;
import uk.insrt.coursework.zuul.world.World;

/**
 * The player's home in the apartments complex.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class RoomApartmentsHome extends CampaignRoom {
    public RoomApartmentsHome(World world) {
        super(world, "Apartments: Home");
    }

    @Override
    public String describe() {
        var world = this.getWorld();
        if (!world.hasVisited(this)) {
            return "<home.first_load>";
        }

        return "<home.enter>";
    }

    @Override
    protected void setupDirections() {
        this.setAdjacent(Direction.DOWN, this.getWorld().getRoom("Apartments: Reception"));
    }

    @Override
    public void spawnEntities() {
        World world = this.getWorld();

        world.spawnEntity("tv", new EntityTV(world, this.toLocation()));
        world.spawnEntity("bed", new EntityBed(world, this.toLocation()));
        world.spawnEntity("laptop", new EntityLaptop(world, this.toLocation()));
```

```
42        }
43    }
```

```java
package uk.insrt.coursework.zuul.content.campaign.rooms;

import uk.insrt.coursework.zuul.content.campaign.entities.EntityNpc;
import uk.insrt.coursework.zuul.world.Direction;
import uk.insrt.coursework.zuul.world.World;

/**
 * The reception of the apartments complex.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class RoomApartmentsReception extends CampaignRoom {
    public RoomApartmentsReception(World world) {
        super(world, "Apartments: Reception");
    }

    @Override
    public String describe() {
        return "<apartments.enter>";
    }

    @Override
    protected void setupDirections() {
        World world = this.getWorld();
        this.setAdjacent(Direction.NORTH, world.getRoom("Street"));
        this.setAdjacent(Direction.EAST, world.getRoom("City Centre"));
        this.setAdjacent(Direction.UP, world.getRoom("Apartments: Home"));
    }

    @Override
    public void spawnEntities() {
        World world = this.getWorld();
        world.spawnEntity("receptionist",
            new EntityNpc(
                world,
                this.toLocation(),
                "npc_receptionist",
                "<apartments.receptionist.description>",
                new String[] { "receptionist" }
            ));
```

```
42        }
43    }
```

```java
package uk.insrt.coursework.zuul.content.campaign.rooms;

import uk.insrt.coursework.zuul.content.campaign.entities.EntityMarie;
import uk.insrt.coursework.zuul.world.Direction;
import uk.insrt.coursework.zuul.world.World;

/**
 * The back alley in the North East side of the map.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class RoomBackAlley extends CampaignRoom {
    public RoomBackAlley(World world) {
        super(world, "Back Alley");
    }

    @Override
    public String describe() {
        var world = this.getWorld();
        if (!world.hasVisited(this)) {
            return "<back_alley.first_load>";
        }

        return "<back_alley.enter>";
    }

    @Override
    protected void setupDirections() {
        this.setAdjacent(Direction.SOUTH, this.getWorld().getRoom("City Centre"));
    }

    @Override
    public void spawnEntities() {
        World world = this.getWorld();
        world.spawnEntity("npc_marie", new EntityMarie(world, this.toLocation()));
    }
}
```

```java
package uk.insrt.coursework.zuul.content.campaign.rooms;

import uk.insrt.coursework.zuul.content.campaign.entities.EntityCat;
import uk.insrt.coursework.zuul.content.campaign.entities.EntityNpc;
import uk.insrt.coursework.zuul.world.Direction;
import uk.insrt.coursework.zuul.world.Room;
import uk.insrt.coursework.zuul.world.World;

/**
 * The city centre connecting most major locations.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class RoomCityCentre extends CampaignRoom {
    public RoomCityCentre(World world) {
        super(world, "City Centre");
    }

    @Override
    public String describe() {
        var world = this.getWorld();
        if (!world.hasVisited(this)) {
            return "<city_centre.first_load>";
        }

        return "<city_centre.enter>";
    }

    @Override
    protected void setupDirections() {
        World world = this.getWorld();
        this.setAdjacent(Direction.NORTH, world.getRoom("Back Alley"));
        this.setAdjacent(Direction.NORTH_WEST, world.getRoom("Street"));
        this.setAdjacent(Direction.WEST, world.getRoom("Apartments: Reception"));
        this.setAdjacent(Direction.SOUTH, world.getRoom("Coastline"));
    }

    @Override
    public void spawnEntities() {
        World world = this.getWorld();
```

```java
        EntityCat cat = new EntityCat(world, this.toLocation());
        world.spawnEntity("cat", cat);
        cat.useWanderAI(
            new Room[] {
                world.getRoom("City Centre"),
                world.getRoom("Street"),
                world.getRoom("Shop"),
                world.getRoom("Street"),
                world.getRoom("City Centre"),
                world.getRoom("Back Alley"),
                world.getRoom("City Centre")
            },
            8
        );

        world.spawnEntity("city_npc",
            new EntityNpc(
                world,
                this.toLocation(),
                "npc_city_centre",
                "<city_centre.npc.description>",
                new String[] { "stranger", "person", "people" }
            ));
    }
}
```

```java
package uk.insrt.coursework.zuul.content.campaign.rooms;

import uk.insrt.coursework.zuul.content.campaign.entities.EntityBoat;
import uk.insrt.coursework.zuul.world.Direction;
import uk.insrt.coursework.zuul.world.World;

/**
 * The coast which connects the main city to the mainland.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class RoomCoastline extends CampaignRoom {
    public RoomCoastline(World world) {
        super(world, "Coastline");
    }

    @Override
    public String describe() {
        return "<coastline.enter>";
    }

    @Override
    protected void setupDirections() {
        this.setAdjacent(Direction.NORTH, this.getWorld().getRoom("City Centre"));
    }

    @Override
    public void spawnEntities() {
        World world = this.getWorld();
        world.spawnEntity("boat1",
            new EntityBoat(world, this.toLocation(),
                world.getRoom("Mainland: Coastline")));
    }
}
```

```java
package uk.insrt.coursework.zuul.content.campaign.rooms;

import uk.insrt.coursework.zuul.content.campaign.entities.EntityOldMan;
import uk.insrt.coursework.zuul.world.Direction;
import uk.insrt.coursework.zuul.world.World;

/**
 * A forest on the mainland side.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class RoomForest extends CampaignRoom {
    public RoomForest(World world) {
        super(world, "Forest");
    }

    @Override
    public String describe() {
        return "<forest.enter>";
    }

    @Override
    protected void setupDirections() {
        World world = this.getWorld();
        this.setAdjacent(Direction.NORTH, world.getRoom("Mainland: Coastline"));
        this.setAdjacent(Direction.EAST, world.getRoom("Worm Hole"));
    }

    @Override
    public void spawnEntities() {
        World world = this.getWorld();
        world.spawnEntity("npc_old_man", new EntityOldMan(world, this.toLocation()));
    }
}
```

```java
package uk.insrt.coursework.zuul.content.campaign.rooms;

import uk.insrt.coursework.zuul.content.campaign.entities.EntityBoat;
import uk.insrt.coursework.zuul.world.Direction;
import uk.insrt.coursework.zuul.world.World;

/**
 * The coast which connects the mainland to the main city.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class RoomMainlandCoastline extends CampaignRoom {
    public RoomMainlandCoastline(World world) {
        super(world, "Mainland: Coastline");
    }

    @Override
    public String describe() {
        return "<mainland_coastline.enter>";
    }

    @Override
    protected void setupDirections() {
        this.setAdjacent(Direction.SOUTH, this.getWorld().getRoom("Forest"));
    }

    @Override
    public void spawnEntities() {
        World world = this.getWorld();
        world.spawnEntity("boat2",
            new EntityBoat(world, this.toLocation(),
                world.getRoom("Coastline")));
    }
}
```

```java
package uk.insrt.coursework.zuul.content.campaign.rooms;

import uk.insrt.coursework.zuul.content.campaign.entities.EntityDocument;
import uk.insrt.coursework.zuul.world.Direction;
import uk.insrt.coursework.zuul.world.World;

/**
 * Private, usually inaccessible room within the Medical Centre complex.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class RoomMedicalCentreOffice extends CampaignRoom {
    public RoomMedicalCentreOffice(World world) {
        super(world, "Medical Centre: Office");
    }

    @Override
    public String describe() {
        return "<medical_centre_office.enter>";
    }

    @Override
    protected void setupDirections() {
        this.setAdjacent(Direction.UP, this.getWorld().getRoom("Medical Centre: Reception"));
    }

    @Override
    public void spawnEntities() {
        World world = this.getWorld();
        for (int i=1;i<=6;i++) {
            world.spawnEntity("doc" + i, new EntityDocument(world, this.toLocation(), i));
        }
    }
}
```

```java
package uk.insrt.coursework.zuul.content.campaign.rooms;

import uk.insrt.coursework.zuul.content.campaign.entities.EntityCouch;
import uk.insrt.coursework.zuul.content.campaign.entities.EntityNpc;
import uk.insrt.coursework.zuul.entities.Entity;
import uk.insrt.coursework.zuul.events.world.EventEntityLeftRoom;
import uk.insrt.coursework.zuul.world.Direction;
import uk.insrt.coursework.zuul.world.World;

/**
 * Reception of the Medical Centre complex.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class RoomMedicalCentreReception extends CampaignRoom {
    private Entity guardEntity;
    private EntityCouch couchEntity;

    public RoomMedicalCentreReception(World world) {
        super(world, "Medical Centre: Reception");
    }

    @Override
    public String describe() {
        return "<medical_centre.enter>";
    }

    @Override
    protected void setupDirections() {
        World world = this.getWorld();
        this.setAdjacent(Direction.EAST, world.getRoom("Street"));
        this.setAdjacent(Direction.DOWN, world.getRoom("Medical Centre: Office"));
    }

    @Override
    public boolean canLeave(Direction direction) {
        if (direction == Direction.DOWN) {
            if (this.guardEntity.getRoom() == this) {
                this.getWorld()
                    .getIO()
```

```java
42                     .println("<medical_centre.guard.blocking>");
43
44                 return false;
45             }
46         }
47
48         return true;
49     }
50
51     @Override
52     public void spawnEntities() {
53         World world = this.getWorld();
54
55         this.guardEntity = new EntityNpc(
56             world,
57             this.toLocation(),
58             "npc_security_guard",
59             "<medical_centre.guard.description>",
60             new String[] { "guard", "security" }
61         );
62         world.spawnEntity("npc_guard", this.guardEntity);
63
64         this.couchEntity = new EntityCouch(world, this.toLocation());
65         world.spawnEntity("couch", this.couchEntity);
66         world.getEventSystem().addListener(EventEntityLeftRoom.class, this.couchEntity);
67     }
68
69     public EntityCouch getCouch() {
70         return this.couchEntity;
71     }
72
73     public Entity getGuard() {
74         return this.guardEntity;
75     }
76 }
```

```java
package uk.insrt.coursework.zuul.content.campaign.rooms;

import uk.insrt.coursework.zuul.content.campaign.entities.EntityShopkeeper;
import uk.insrt.coursework.zuul.world.Direction;
import uk.insrt.coursework.zuul.world.World;

/**
 * A shop within the city, the only one the player can interact with.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class RoomShop extends CampaignRoom {
    public RoomShop(World world) {
        super(world, "Shop");
    }

    @Override
    public String describe() {
        return "<shop.enter>";
    }

    @Override
    protected void setupDirections() {
        this.setAdjacent(Direction.SOUTH, this.getWorld().getRoom("Street"));
    }

    @Override
    public void spawnEntities() {
        World world = this.getWorld();
        world.spawnEntity("npc_shopkeeper",
            new EntityShopkeeper(world, this.toLocation()));
    }
}
```

```java
package uk.insrt.coursework.zuul.content.campaign.rooms;

import uk.insrt.coursework.zuul.content.campaign.StoryFlags.Stage;
import uk.insrt.coursework.zuul.content.campaign.entities.EntityNpc;
import uk.insrt.coursework.zuul.content.campaign.events.EventGameStageChanged;
import uk.insrt.coursework.zuul.entities.Entity;
import uk.insrt.coursework.zuul.events.IEventListener;
import uk.insrt.coursework.zuul.world.Direction;
import uk.insrt.coursework.zuul.world.World;

/**
 * One of the major connecting points between locations in the city.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class RoomStreet extends CampaignRoom implements IEventListener<EventGameStageChanged> {
    private Entity protestorsEntity;

    public RoomStreet(World world) {
        super(world, "Street");
    }

    @Override
    public String describe() {
        var world = this.getWorld();
        if (!world.hasVisited(this)) {
            return "<street.first_load>";
        }

        return "<street.enter>";
    }

    @Override
    protected void setupDirections() {
        World world = this.getWorld();
        this.setAdjacent(Direction.SOUTH, world.getRoom("Apartments: Reception"));
        this.setAdjacent(Direction.EAST, world.getRoom("City Centre"));
        this.setAdjacent(Direction.NORTH, world.getRoom("Shop"));
        this.setAdjacent(Direction.WEST, world.getRoom("Medical Centre: Reception"));
    }
```

```java
42
43        @Override
44        public boolean canLeave(Direction direction) {
45            if (direction == Direction.WEST) {
46                if (this.protestorsEntity.getRoom() == this) {
47                    this.getWorld()
48                            .getIO()
49                            .println("<street.protestors.blocking>");
50
51                    return false;
52                }
53            }
54
55            return true;
56        }
57
58        @Override
59        public void spawnEntities() {
60            World world = this.getWorld();
61            this.protestorsEntity = new EntityNpc(
62                world,
63                this.toLocation(),
64                "npc_protestors",
65                "<street.protestors.description>",
66                new String[] { "protestors", "protestor" }
67            );
68            world.spawnEntity("npc_protestors", this.protestorsEntity);
69        }
70
71        @Override
72        public void onEvent(EventGameStageChanged event) {
73            // Remove the protestors if we are in Stealth chapter.
74            if (event.getStage() == Stage.Stealth) {
75                this.protestorsEntity.consume(false);
76            } else {
77                this.protestorsEntity.setLocation(this);
78            }
79        }
80    }
```

```java
package uk.insrt.coursework.zuul.content.campaign.rooms;

import java.util.Random;

import uk.insrt.coursework.zuul.entities.Entity;
import uk.insrt.coursework.zuul.entities.EntityPlayer;
import uk.insrt.coursework.zuul.events.IEventListener;
import uk.insrt.coursework.zuul.events.world.EventEntityEnteredRoom;
import uk.insrt.coursework.zuul.sound.EventSound;
import uk.insrt.coursework.zuul.sound.SoundType;
import uk.insrt.coursework.zuul.world.Room;
import uk.insrt.coursework.zuul.world.World;

/**
 * Teleporter room implemented as required by the challenge tasks.
 * Any Entity that walks into the worm hole is transported into a random public location.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class RoomWormHole extends CampaignRoom implements IEventListener<EventEntityEnteredRoom> {
    public RoomWormHole(World world) {
        super(world, "Worm Hole");
    }

    @Override
    public String describe() {
        return "";
    }

    @Override
    protected void setupDirections() {}

    @Override
    public void onEvent(EventEntityEnteredRoom event) {
        Entity entity = event.getEntity();
        Room room = entity.getRoom();
        if (room != this) return;
        event.stopPropagation();

        // This is a restricted set of locations as to not break
```

```java
            // the game's plot, say if we were transported to the medical
            // centre complex office when we're not meant to go there yet.
            final Random random = new Random();
            final String[] locations = {
                "City Centre",
                "Coastline",
                "Mainland: Coastline",
                "Forest",
                "Street",
                "Back Alley"
            };

            var world = this.getWorld();
            var io = world.getIO();
            io.println("\n<worm_hole.enter>");

            try {
                Thread.sleep(1000);

                final int WIDTH = 79;

                for (int i=0;i<5;i++) {
                    io.println("*".repeat(i*3) + "\\"
                        + " ".repeat(WIDTH - i * 6 - 2) + "/" + "*".repeat(i*3));

                    Thread.sleep(60);
                }

                // Play worm hole sound while we are falling through time and space.
                world.emit(new EventSound(SoundType.WormHole));

                for (int i=0;i<25*16;i++) {
                    var out = "";
                    for (int j=0;j<WIDTH;j++) {
                        out += random.nextInt(8) == 0 ? "*" : " ";
                    }

                    io.println(out);
                    Thread.sleep(40);
                }
```

```java
 83                 for (int i=5;i>0;i--) {
 84                     io.println("*".repeat(i*3) + "/"
 85                         + " ".repeat(WIDTH - i * 6 - 2) + "\\" + "*".repeat(i*3));
 86
 87                     Thread.sleep(60);
 88                 }
 89             } catch (InterruptedException e) {
 90                 e.printStackTrace();
 91                 io.println("There was a disruption when travelling.");
 92             }
 93
 94             io.print("\n");
 95
 96             // Pick a random location and put the entering entity in it.
 97             String location = locations[random.nextInt(locations.length)];
 98             Room target = this.getWorld().getRoom(location);
 99             entity.setLocation(target);
100
101             // If it was the player, clear their walk history.
102             if (entity instanceof EntityPlayer) {
103                 ((EntityPlayer) entity).clearHistory();
104             }
105         }
106 }
```

```java
package uk.insrt.coursework.zuul.content.campaign;

import java.util.HashSet;

import uk.insrt.coursework.zuul.content.campaign.events.EventGameStageChanged;
import uk.insrt.coursework.zuul.events.EventSystem;
import uk.insrt.coursework.zuul.events.world.EventTick;

/**
 * Class which controls story progression within the Campaign World.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class StoryFlags {
    /**
     * The current story chapter.
     */
    public enum Stage {
        Exposition, // Ch 1.
        Recon, // Ch 2.
        Stealth, // Ch 3.
        End, // Current Ending

        Twist, // Ch 4. Skipped
        Conclusion, // Ch 5. Skipped
    }

    /**
     * Side-quests available in the game.
     */
    public enum Quest {
        Cat
    }

    private EventSystem eventSystem;
    private HashSet<Quest> quests;
    private int balance;
    private Stage stage;
    private int ticks;
```

```java
    /**
     * Construct a new instance of StoryFlags
     * @param eventSystem World event system
     */
    public StoryFlags(EventSystem eventSystem) {
        this.eventSystem = eventSystem;
        this.stage = Stage.Exposition;
        this.quests = new HashSet<>();
        this.balance = 100_000;
        this.ticks = 0;

        this.eventSystem.addListener(EventTick.class, e -> this.ticks++);
    }

    /**
     * Get the current stage (chapter) of the story.
     * @return Current stage
     */
    public Stage getStage() {
        return this.stage;
    }

    /**
     * Set the current stage (chapter) of the story.
     * @param stage New stage
     */
    public void setStage(Stage stage) {
        this.stage = stage;
        this.eventSystem.emit(new EventGameStageChanged(stage));
    }

    /**
     * Get the player's balance
     * @return Player's balance
     */
    public int getBalance() {
        return this.balance;
    }

    /**
     * Set player's new balance.
```

```java
 83         * @param balance New balance
 84         */
 85        public void setBalance(int balance) {
 86            this.balance = balance;
 87        }
 88
 89        /**
 90         * Deduct money from the player's balance.
 91         * @param value Amount to deduct
 92         * @return Whether we could deduct the balance without going below zero
 93         */
 94        public boolean deductFromBalance(int value) {
 95            if (value > this.balance) {
 96                return false;
 97            }
 98
 99            this.balance -= value;
100            return true;
101        }
102
103        /**
104         * Get ticks since start of the World.
105         * @return Number of ticks since start
106         */
107        public int getTicks() {
108            return this.ticks;
109        }
110
111        /**
112         * Mark a side-quest as complete
113         */
114        public void completeSideQuest(Quest quest) {
115            this.quests.add(quest);
116        }
117
118        /**
119         * Get completed side-quests.
120         * @return Number of completed side-quests
121         */
122        public int getCompletedQuests() {
123            return this.quests.size();
```

```
124        }
125
126        /**
127         * Get total number of side-quests.
128         * @return Total number of side-quests
129         */
130        public int getTotalQuests() {
131            return Quest.values().length;
132        }
133    }
```

```java
package uk.insrt.coursework.zuul.dialogue;

import java.util.HashMap;

import uk.insrt.coursework.zuul.io.IOSystem;

/**
 * Simple dialogue engine which navigates between {@link DialogueNode}(s).
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class Dialogue<T> {
    private HashMap<T, DialogueNode<T>> parts;
    private T currentNode;

    /**
     * Contruct a new Dialogue engine.
     */
    public Dialogue() {
        this.parts = new HashMap<>();
    }

    /**
     * Construct a new Dialogue engine and initialise us at a starting node.
     * @param start Starting node
     */
    public Dialogue(T start) {
        this();
        this.currentNode = start;
    }

    /**
     * Get the current node
     */
    public T getCurrentNode() {
        return this.currentNode;
    }

    /**
     * Set the current node
```

```java
42          * @param node New node
43          */
44         public void getCurrentNode(T node) {
45             this.currentNode = node;
46         }
47
48         /**
49          * Change the current node to a different one if it exists
50          * @param node New node
51          */
52         public void setNodeIfPresent(T node) {
53             if (this.parts.containsKey(node)) {
54                 this.currentNode = node;
55             }
56         }
57
58         /**
59          * Add a new part to the dialogue
60          * @param part What this node is identified by
61          * @param node The new node
62          */
63         public void addPart(T part, DialogueNode<T> node) {
64             this.parts.put(part, node);
65         }
66
67         /**
68          * Get an existing part from the dialogue
69          * @param part What the node is identified by
70          * @return The node if it exists, otherwise null
71          */
72         public DialogueNode<T> getPart(T part) {
73             return this.parts.get(part);
74         }
75
76         /**
77          * Run the Dialogue engine until one of the options exits us out
78          * @param io Provided IO system
79          */
80         public void run(IOSystem io) {
81             var part = this.parts.get(this.currentNode);
82             io.println("\n" + part.getDescription());
```

```java
83          DialogueOption<T> option = part.pickOption(io);
84
85          T target = option.handle(io);
86          if (target == null) {
87              T newTarget = option.getTarget();
88              if (newTarget != null) {
89                  this.currentNode = newTarget;
90              }
91
92
93              return;
94          }
95
96          this.currentNode = target;
97          this.run(io);
98      }
99  }
```

```java
1    package uk.insrt.coursework.zuul.dialogue;
2
3    import java.io.IOException;
4    import java.io.InputStream;
5    import java.util.HashMap;
6    import java.util.List;
7    import java.util.Map;
8    import java.util.Map.Entry;
9
10   import com.moandjiezana.toml.Toml;
11
12   /**
13    * This is a helper class for loading and populating {@link Dialogue}.
14    * This DialogueLoader assumes that i18n is being used in the dialogue data.
15    *
16    * @author Pawel Makles (K21002534)
17    * @version 1.0-SNAPSHOT
18    */
19   public class DialogueLoader {
20       private Map<String, Object> data;
21
22       /**
23        * Construct a new DialogueLoader
24        */
25       public DialogueLoader() {
26           this.data = new HashMap<>();
27       }
28
29       /**
30        * Load all necessary data for populating Dialogue.
31        * @param path Path to the dialogue resource file
32        * @throws IOException if we can't read the dialogue file
33        */
34       public void load(String path) throws IOException {
35           InputStream stream = DialogueLoader.class.getResourceAsStream(path);
36           this.data = new Toml().read(stream).toMap();
37       }
38
39       /**
40        * Populate a Dialogue system using a specific dialog definition represented by a given key.
41        *
```

```java
42          * This method is unchecked as we expect a valid data structure to have
43          * been loaded from the resource, this should be verified by the developer.
44          * @param dialogue Dialogue system
45          * @param key Key to lookup
46          */
47         @SuppressWarnings("unchecked")
48         public void populate(Dialogue<String> dialogue, String key) {
49             Map<String, Object> nodes = (Map<String, Object>) this.data.get(key);
50
51             // Process any special keys first before we continue.
52             String prefix = "";
53             for (String nodeKey : nodes.keySet()) {
54                 if (nodeKey.equals("_prefix")) {
55                     prefix = (String) nodes.get(nodeKey);
56                 } else if (nodeKey.equals("_start")) {
57                     dialogue.getCurrentNode((String) nodes.get(nodeKey));
58                 }
59             }
60
61             for (Entry<String, Object> node : nodes.entrySet()) {
62                 // Ignore any keys starting with _, as they are used above.
63                 String nodeKey = node.getKey();
64                 if (nodeKey.startsWith("_")) continue;
65
66                 // Read each node's values and find the description and options.
67                 Map<String, Object> values = (Map<String, Object>) node.getValue();
68
69                 // Description strings are assumed to be i18n paths.
70                 String description = "<" + prefix + (String) values.get("description") + ">";
71                 List<Map<String, Object>> options = (List<Map<String, Object>>) values.get("options");
72
73                 // Construct a new Dialogue Node with the given data.
74                 DialogueNode<String> dialogueNode = new DialogueNode<>(description);
75                 for (Map<String, Object> object : options) {
76                     String desc = "<" + prefix + (String) object.get("description") + ">";
77                     String to = (String) object.get("to");
78                     Boolean mustExit = (Boolean) object.get("mustExit");
79
80                     if (mustExit == null) {
81                         dialogueNode.addOption(desc, to);
82                     } else {
```

```
83                        dialogueNode.addOption(desc, to, mustExit);
84                    }
85                }
86
87            dialogue.addPart(nodeKey, dialogueNode);
88        }
89    }
90 }
```

```java
package uk.insrt.coursework.zuul.dialogue;

import java.util.ArrayList;
import java.util.List;

import uk.insrt.coursework.zuul.io.IOSystem;

/**
 * A node in a {@link Dialogue} system.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class DialogueNode<T> {
    private String description;
    private ArrayList<DialogueOption<T>> options;

    /**
     * Construct a new node.
     * @param description Description of this node
     */
    public DialogueNode(String description) {
        this.description = description;
        this.options = new ArrayList<>();
    }

    /**
     * Get this node's description
     * @return Description string
     */
    public String getDescription() {
        return this.description;
    }

    /**
     * Add a new option which branches off this node.
     * @param option Dialogue Option
     * @return This Dialogue Node so other method calls can be chained
     */
    public DialogueNode<T> addOption(DialogueOption<T> option) {
        this.options.add(option);
```

```java
42              return this;
43          }
44
45          /**
46           * Create a new option which branches off this node.
47           * @param description Description of this option
48           * @param stage The next stage of the dialogue this should jump to
49           * @param mustExit Whether we must exit from the dialogue after selecting this option
50           * @return This Dialogue Node so other method calls can be chained
51           */
52          public DialogueNode<T> addOption(String description, T stage, boolean mustExit) {
53              var option = new DialogueOption<T>(description, stage);
54              if (mustExit) option.mustExit();
55              this.options.add(option);
56              return this;
57          }
58
59          /**
60           * Create a new option which branches off this node.
61           * @param description Description of this option
62           * @param stage The next stage of the dialogue this should jump to
63           * @return This Dialogue Node so other method calls can be chained
64           */
65          public DialogueNode<T> addOption(String description, T stage) {
66              return this.addOption(description, stage, false);
67          }
68
69          /**
70           * Get the options available to this node.
71           * @return List of options
72           */
73          protected List<DialogueOption<T>> getOptions() {
74              return this.options;
75          }
76
77          /**
78           * Ask the player to pick one of the valid options branching off this node.
79           * @param io Provided IO system
80           * @return The selected option
81           */
82          public DialogueOption<T> pickOption(IOSystem io) {
```

```java
         List<DialogueOption<T>> options = this.getOptions();
         for (int i=0;i<options.size();i++) {
             io.println((i + 1) + ". " + options.get(i).getDescription());
         }

         while (true) {
             io.print("Choice: ");
             String value = io.readLine();
             try {
                 int v = Integer.parseInt(value);
                 if (v < 1 || v > options.size()) {
                     io.println("Provide a valid option!");
                     continue;
                 }

                 return options.get(v - 1);
             } catch (Exception e) {
                 io.println("Provide a valid number!");
             }
         }
     }
}
```

```java
package uk.insrt.coursework.zuul.dialogue;

import uk.insrt.coursework.zuul.io.IOSystem;

/**
 * An option which branches off a {@link DialogueNode} into another node.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class DialogueOption<T> {
    private IDialogueHandler<T> handler;

    private String description;
    private boolean shouldExit;
    private T target;

    /**
     * Construct a new simple DialogueOption with a description and destination.
     * @param description Description of this option
     * @param target Target node to jump to
     */
    public DialogueOption(String description, T target) {
        this.target = target;
        this.description = description;
    }

    /**
     * Construct a complex DialogueOption with a description and select handler.
     * @param description Description of this option
     * @param handler Method called when this option is selected
     */
    public DialogueOption(String description, IDialogueHandler<T> handler) {
        this.handler = handler;
        this.description = description;
    }

    /**
     * Tell the Dialogue system to exit if this option is selected.
     * @return This dialogue option so method calls can be chained
     */
```

```java
42      public DialogueOption<T> mustExit() {
43          this.shouldExit = true;
44          return this;
45      }
46
47      /**
48       * Get the description of this option.
49       * @return Description string
50       */
51      public String getDescription() {
52          return this.description;
53      }
54
55      /**
56       * Get the destination of this option.
57       * @return Destination if it exists
58       */
59      public T getTarget() {
60          return this.target;
61      }
62
63      /**
64       * Handle the player selecting this dialogue option.
65       * @param io Provided IO system
66       * @return The new node or null if we should exit and stay put.
67       */
68      public T handle(IOSystem io) {
69          if (this.handler != null) {
70              return this.handler.onAction(io);
71          } else if (!this.shouldExit) {
72              return this.target;
73          }
74
75          return null;
76      }
77  }
```

```java
package uk.insrt.coursework.zuul.dialogue;

import uk.insrt.coursework.zuul.io.IOSystem;

/**
 * Interface implemented to provide an onAction method.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public interface IDialogueHandler<T> {
    /**
     * Handle the selection of a dialogue option.
     * @param io Provided IO system
     * @return Destination node, may be null
     */
    public T onAction(IOSystem io);
}
```

```java
package uk.insrt.coursework.zuul.entities.actions;

import uk.insrt.coursework.zuul.entities.Entity;

/**
 * Interface implemented to provide the ability for
 * an Entity to have other Entities given to them.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public interface IGiveable {
    /**
     * Give this entity another entity.
     * @param item Item being given
     */
    public void give(Entity item);
}
```

```java
package uk.insrt.coursework.zuul.entities.actions;

/**
 * Interface implemented to provide the
 * ability for an Entity to be pet.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public interface IPettable {
    /**
     * Pet this entity.
     */
    public void pet();
}
```

```java
package uk.insrt.coursework.zuul.entities.actions;

/**
 * Interface implemented to provide the
 * ability for an Entity to talk with the player.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public interface ITalkwith {
    /**
     * Talk with this entity.
     */
    public void talk();
}
```

```java
package uk.insrt.coursework.zuul.entities.actions;

import uk.insrt.coursework.zuul.entities.Entity;

/**
 * Interface implemented to provide the
 * ability for an Entity to be used by the player.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public interface IUseable {
    /**
     * Use this entity.
     * @param target The Entity taking this entity.
     */
    public void use(Entity target);
}
```

```java
1    package uk.insrt.coursework.zuul.entities;
2
3    import uk.insrt.coursework.zuul.events.world.EventEntityEnteredRoom;
4    import uk.insrt.coursework.zuul.events.world.EventEntityLeftRoom;
5    import uk.insrt.coursework.zuul.io.Ansi;
6    import uk.insrt.coursework.zuul.world.Location;
7    import uk.insrt.coursework.zuul.world.Room;
8    import uk.insrt.coursework.zuul.world.World;
9
10   /**
11    * Representation of any Entity in the World.
12    *
13    * Any living beings, items, or otherwise things that
14    * exist in the world are considered an Entity. Each
15    * Entity also has an Inventory so things may be stored
16    * inside of it.
17    *
18    * @author Pawel Makles (K21002534)
19    * @version 1.0-SNAPSHOT
20    */
21   public abstract class Entity {
22       protected World world;
23       protected Inventory inventory;
24
25       private Location location;
26       private double weight;
27
28       /**
29        * Construct a new Entity.
30        * @param world Current World object
31        * @param location Initial Location of this Entity
32        * @param weight The weight (in kg) of this Entity
33        */
34       public Entity(World world, Location location, double weight) {
35           this.world = world;
36           this.location = location;
37           this.inventory = new Inventory();
38           this.weight = weight;
39       }
40
41       /**
```

```java
42       * Construct a new Entity.
43       *
44       * Weight value is set to {@link Integer#MAX_VALUE}.
45       * @param world Current World object
46       * @param location Initial Location of this Entity
47       */
48      public Entity(World world, Location location) {
49          this(world, location, Integer.MAX_VALUE);
50      }
51
52      /**
53       * Get this Entity's weight.
54       * @return Weight (in kg)
55       */
56      public double getWeight() {
57          return this.weight;
58      }
59
60      /**
61       * Get the name of this Entity.
62       * Shorthand for {@link #getAliases()}[0].
63       * @return First matched alias
64       */
65      public String getName() {
66          return this.getAliases()[0];
67      }
68
69      /**
70       * Get a highlighted representation of this Entity's name.
71       * @return Ansi hightlighted name
72       */
73      public String getHighlightedName() {
74          return Ansi.BackgroundWhite + Ansi.Black + this.getName() + Ansi.Reset;
75      }
76
77      /**
78       * Get the Inventory that this Entity holds.
79       * @return Inventory
80       */
81      public Inventory getInventory() {
82          return this.inventory;
```

```java
 83        }
 84
 85        /**
 86         * Get the World this Entity resides in.
 87         * @return World
 88         */
 89        public World getWorld() {
 90            return this.world;
 91        }
 92
 93        /**
 94         * Get the Room that this Entity is currently in.
 95         * @return Room
 96         */
 97        public Room getRoom() {
 98            return this.location.getRoom();
 99        }
100
101        /**
102         * Get the Inventory that this Entity is currently in.
103         * @return Inventory
104         */
105        public Inventory getInventoryWithin() {
106            return this.location.getInventory();
107        }
108
109        /**
110         * Remove this Entity from any existing place.
111         * Provides a consistent way to clean up the Entity before placing it anywhere.
112         * @param suppressEvents Whether to suppress the "Entity Left Room" Event
113         * @return Whether this Entity was removed from an Inventory
114         */
115        public boolean consume(boolean suppressEvents) {
116            boolean consumed = false;
117            Inventory inventory = this.location.getInventory();
118            if (inventory != null) consumed = inventory.remove(this);
119
120            Room previousRoom = this.getRoom();
121            if (previousRoom != null && !suppressEvents) this.world.emit(new EventEntityLeftRoom(this, previousRoom));
122
123            this.location.clear();
```

```java
124            return consumed;
125        }
126
127        /**
128         * Move the Entity into a Room.
129         * @param room Destination Room
130         */
131        public void setLocation(Room room) {
132            boolean consumed = this.consume(false);
133            this.location.setLocation(room);
134            if (!consumed) this.world.emit(new EventEntityEnteredRoom(this));
135        }
136
137        /**
138         * Move the Entity into an Inventory.
139         * @param inventory Destination Inventory
140         * @return Whether we successfully moved the entity into the inventory.
141         */
142        public boolean setLocation(Inventory inventory) {
143            if (inventory.add(this)) {
144                this.consume(true);
145                this.location.setLocation(inventory);
146                return true;
147            }
148
149            return false;
150        }
151
152        /**
153         * Link this Entity's inventory with an existing inventory.
154         * @param inventory Target inventory
155         */
156        public void entangleInventory(Inventory inventory) {
157            this.inventory = inventory;
158        }
159
160        /**
161         * Get names that this Entity can be called by.
162         * @return String array of names for this Entity
163         */
164        public abstract String[] getAliases();
```

```java
165
166      /**
167       * Get a description of this Entity.
168       * @return String describing the Entity
169       */
170      public abstract String describe();
171  }
```

```java
package uk.insrt.coursework.zuul.entities;

import uk.insrt.coursework.zuul.world.Location;
import uk.insrt.coursework.zuul.world.World;

/**
 * Generic object class which avoids some boilerplate.
 * Use this for entities which are guaranteed to never change.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class EntityObject extends Entity {
    private String description;
    private String[] aliases;

    /**
     * Construct a new EntityObject
     * @param world Current World object
     * @param location Initial Location of this Entity
     * @param weight The weight (in kg) of this Entity
     * @param aliases Aliases which this object can be referred to by
     * @param description A description of this object
     */
    public EntityObject(World world, Location location, double weight, String[] aliases, String description) {
        super(world, location, weight);
        this.description = description;
        this.aliases = aliases;
    }

    /**
     * Construct a new EntityObject
     * @param world Current World object
     * @param location Initial Location of this Entity
     * @param weight The weight (in kg) of this Entity
     * @param name Name of this object
     * @param description A description of this object
     */
    public EntityObject(World world, Location location, double weight, String alias, String description) {
        this(world, location, weight, new String[] { alias }, description);
    }
```

```java
42
43      @Override
44      public String describe() {
45          return this.description;
46      }
47
48      @Override
49      public String[] getAliases() {
50          return this.aliases;
51      }
52  }
```

```java
package uk.insrt.coursework.zuul.entities;

import java.util.ArrayList;

import uk.insrt.coursework.zuul.io.IOSystem;
import uk.insrt.coursework.zuul.world.Direction;
import uk.insrt.coursework.zuul.world.Location;
import uk.insrt.coursework.zuul.world.Room;
import uk.insrt.coursework.zuul.world.World;

/**
 * Player entity which we can control and move around.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class EntityPlayer extends Entity {
    private ArrayList<Room> previousRooms;
    private ArrayList<Direction> retreatingDirection;

    /**
     * Construct a new Player Entity.
     * @param world World to place Player in
     */
    public EntityPlayer(World world) {
        super(world, new Location(), 70);
        this.previousRooms = new ArrayList<>();
        this.retreatingDirection = new ArrayList<>();
        this.inventory.setMaxWeight(35);
    }

    @Override
    public String[] getAliases() {
        return new String[] {
            "player", "me", "myself", "self", "yourself"
        };
    }

    @Override
    public String describe() {
        // We may skip defining how the Player looks,
```

```java
42                // this is because EntityPlayer is ignored
43                // when looking around the room.
44                return "";
45            }
46
47        /**
48         * Move in a direction as instructed by command.
49         * @param direction Target Direction
50         */
51        public void go(Direction direction) {
52            var io = this.getWorld().getIO();
53
54            Room room = this.getRoom();
55            if (room == null) {
56                io.println("You appear to be trapped.");
57                return;
58            }
59
60            if (!room.canLeave(direction)) return;
61
62            Room destination = room.getAdjacent(direction);
63            if (destination == null) {
64                io.println("You cannot go this way.");
65                return;
66            }
67
68            this.retreatingDirection.add(direction.flip());
69            this.previousRooms.add(this.getRoom());
70            this.setLocation(destination);
71        }
72
73        /**
74         * Move to the previous room the player was in.
75         */
76        public void back() {
77            IOSystem io = this.getWorld().getIO();
78            int index = this.retreatingDirection.size() - 1;
79
80            if (index < 0) {
81                io.println("Nowhere to go back to!");
82                return;
```

```java
        }

        Direction lastDirection = this.retreatingDirection.get(index);
        if (this.getRoom().hasExit(lastDirection)) {
            this.retreatingDirection.remove(index);
            this.setLocation(this.previousRooms.remove(index));
        } else {
            io.println("Cannot leave the room this way.");
        }
    }

    /**
     * Clear walk history.
     */
    public void clearHistory() {
        this.retreatingDirection.clear();
        this.previousRooms.clear();
    }
}
```

```java
package uk.insrt.coursework.zuul.entities;

import java.util.ArrayList;

/**
 * Representation of an Entity's inventory
 * and what they are holding.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class Inventory {
    private ArrayList<Entity> items = new ArrayList<>();
    private double maxWeight;

    /**
     * Construct a new Inventory.
     */
    public Inventory() {
        super();
        this.maxWeight = 0;
    }

    /**
     * Set the max weight that can be carried in this inventory.
     * @param maxWeight Max weight (in kg)
     */
    public void setMaxWeight(double maxWeight) {
        this.maxWeight = maxWeight;
    }

    /**
     * Get the maximum weight that can be carried in this inventory.
     * @return Maximum weight that can be carried
     */
    public double getMaxWeight() {
        return this.maxWeight;
    }

    /**
     * Get the current weight of this inventory.
```

```java
42          * @return Weight (in kg)
43          */
44         public double getWeight() {
45             return this
46                     .items
47                     .stream()
48                     .mapToDouble(Entity::getWeight)
49                     .sum();
50         }
51
52         /**
53          * Check if the inventory is full.
54          * @return True if the weight is greater than the max weight
55          */
56         public boolean isFull() {
57             return this.getWeight() >= this.getMaxWeight();
58         }
59
60         /**
61          * Add an entity to this inventory.
62          *
63          * There must be sufficient space for the entity.
64          * @param entity Target Entity
65          * @return Whether we successfully added the new entity.
66          */
67         public boolean add(Entity entity) {
68             if (this.getWeight() + entity.getWeight() > this.maxWeight) {
69                 return false;
70             }
71
72             this.items.add(entity);
73             return true;
74         }
75
76         /**
77          * Remove an entity from this inventory.
78          * @param entity Target Entity
79          * @return Whether there was any change to the inventory.
80          */
81         public boolean remove(Entity entity) {
82             return this.items.remove(entity);
```

```java
83          }
84
85          /**
86           * Get an Iterable over the Entities within this inventory.
87           * @return Iterable over Entities
88           */
89          public Iterable<Entity> getItems() {
90              return this.items;
91          }
92      }
```

```java
package uk.insrt.coursework.zuul.events;

/**
 * Represents a single event fired from
 * any source to be consumed by anything.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class Event {
    private boolean propagating = true;

    /**
     * Whether this event can continue running.
     * @return Whether propogation of this event was stopped
     */
    public boolean canRun() {
        return this.propagating;
    }

    /**
     * Stop further propagation of this event.
     */
    public void stopPropagation() {
        this.propagating = false;
    }
}
```

```java
package uk.insrt.coursework.zuul.events;

import java.util.HashMap;
import java.util.HashSet;
import java.util.LinkedHashSet;

/**
 * Event system which manages taking in events
 * from different sources and handles them
 * by firing callbacks on event listeners.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class EventSystem {
    private HashMap<Class<? extends Event>, LinkedHashSet<IEventListener<? extends Event>>> listeners = new HashMap<>();

    /**
     * Get existing Event listener list or create a new one if not exists.
     * @param event Event
     * @return Set of event listeners
     */
    private HashSet<IEventListener<? extends Event>> getList(Class<? extends Event> event) {
        var list = this.listeners.get(event);
        if (list == null) {
            list = new LinkedHashSet<>();
            this.listeners.put(event, list);
        }

        return list;
    }

    /**
     * Add a new event listener to this system.
     * @param <E> Generic Event type
     * @param event Event to remove from
     * @param listener Event listener callback
     */
    public<E extends Event> void addListener(Class<E> event, IEventListener<E> listener) {
        this.getList(event).add(listener);
    }
```

```java
42
43        /**
44         * Remove an new event listener from this system.
45         * @param <E> Generic Event type
46         * @param event Event to remove from
47         * @param listener Event listener callback
48         */
49        public<E extends Event> void removeListener(Class<E> event, IEventListener<E> listener) {
50            this.getList(event).remove(listener);
51        }
52
53        /**
54         * Emit an Event.
55         * @param <E> Generic Event type
56         * @param event Event to emit
57         */
58        @SuppressWarnings("unchecked")
59        public <E extends Event> void emit(E event) {
60            var listeners = this.listeners.get(event.getClass());
61            if (listeners == null) return;
62
63            for (@SuppressWarnings("rawtypes") IEventListener listener : listeners) {
64                listener.onEvent(event);
65                // Previously, there was a try catch ClassCastException
66                // but I've since constricted the types on `addListener`
67                // and `removeListener` so this should never happen.
68
69                if (!event.canRun())
70                    break;
71            }
72        }
73    }
```

```java
package uk.insrt.coursework.zuul.events;

/**
 * Interface implementing an listener for an arbitrary {@link Event}.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public interface IEventListener<E extends Event> {
    /**
     * Method called when this specific Event is emitted.
     * @param event Event to handle
     */
    public void onEvent(E event);
}
```

```java
package uk.insrt.coursework.zuul.events.world.behaviours;

import java.util.Random;

import uk.insrt.coursework.zuul.entities.Entity;
import uk.insrt.coursework.zuul.events.IEventListener;
import uk.insrt.coursework.zuul.events.world.EventTick;
import uk.insrt.coursework.zuul.world.Room;

/**
 * This is a simple behaviour which just randomly decides to move an Entity
 * through a set path whenever the game ticks forwards.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class SimpleWanderAI implements IEventListener<EventTick> {
    private Entity entity;
    private Room[] path;
    private int chance;

    private int index;
    private Random random;

    /**
     * Construct a new wandering behaviour for an Entity with a given path.
     * @param entity Entity which should be moved
     * @param path Path that this Entity should follow
     * @param chance The chance x that this entity moves, where x gives a 1/x fractional chance of moving.
     */
    public SimpleWanderAI(Entity entity, Room[] path, int chance) {
        this.entity = entity;
        this.path = path;
        this.chance = chance;

        this.index = 0;
        this.random = new Random();
    }

    @Override
    public void onEvent(EventTick event) {
```

```java
        if (this.entity.getRoom() != this.path[this.index]) return;
        if (random.nextInt(this.chance) > 0) return;

        this.index = (this.index + 1) % this.path.length;
        this.entity.setLocation(this.path[this.index]);
    }
}
```

```java
package uk.insrt.coursework.zuul.events.world;

import uk.insrt.coursework.zuul.entities.Entity;
import uk.insrt.coursework.zuul.events.Event;

/**
 * Event fired when an Entity enters a room.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class EventEntityEnteredRoom extends Event {
    private Entity entity;

    /**
     * Construct a new EntityEnteredRoom Event.
     * @param entity Target Entity
     */
    public EventEntityEnteredRoom(Entity entity) {
        this.entity = entity;
    }

    /**
     * Get the Entity relating to this event.
     * @return Entity
     */
    public Entity getEntity() {
        return this.entity;
    }
}
```

```java
package uk.insrt.coursework.zuul.events.world;

import uk.insrt.coursework.zuul.entities.Entity;
import uk.insrt.coursework.zuul.events.Event;
import uk.insrt.coursework.zuul.world.Room;

/**
 * Event fired when an Entity enters a room.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class EventEntityLeftRoom extends Event {
    private Entity entity;
    private Room room;

    /**
     * Construct a new EntityLeftRoom Event.
     * @param entity Target Entity
     * @param room Room the entity left
     */
    public EventEntityLeftRoom(Entity entity, Room room) {
        this.entity = entity;
        this.room = room;
    }

    /**
     * Get the Entity relating to this event.
     * @return Entity
     */
    public Entity getEntity() {
        return this.entity;
    }

    /**
     * Get the Room relating to this event.
     * @return Room
     */
    public Room getRoom() {
        return this.room;
```

```
41        }
42    }
```

```java
package uk.insrt.coursework.zuul.events.world;

import uk.insrt.coursework.zuul.events.Event;

/**
 * Event fired when an arbitrary command is about to be run.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class EventProcessCommand extends Event {
    private String cmd;

    /**
     * Construct a new EventProcessCommand Event.
     * @param cmd Target command
     */
    public EventProcessCommand(String cmd) {
        this.cmd = cmd;
    }

    /**
     * Set command for this event.
     * @param cmd Overwrite current command
     */
    public void setCommand(String cmd) {
        this.cmd = cmd;
    }

    /**
     * Get the command relating to this event.
     * @return Arbitrary command
     */
    public String getCommand() {
        return this.cmd;
    }
}
```

```java
package uk.insrt.coursework.zuul.events.world;

import uk.insrt.coursework.zuul.events.Event;

/**
 * Event fired when the game ticks forward.
 * Such as when the player performs an action or goes to sleep.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class EventTick extends Event {}
```

```java
package uk.insrt.coursework.zuul;

import java.io.IOException;

import javax.swing.JOptionPane;

import uk.insrt.coursework.zuul.commands.CommandManager;
import uk.insrt.coursework.zuul.content.campaign.CampaignWorld;
import uk.insrt.coursework.zuul.content.campaign.commands.CommandMap;
import uk.insrt.coursework.zuul.content.campaign.commands.CommandWin;
import uk.insrt.coursework.zuul.events.world.EventProcessCommand;
import uk.insrt.coursework.zuul.events.world.EventTick;
import uk.insrt.coursework.zuul.io.IOSystem;
import uk.insrt.coursework.zuul.io.LocalisedIO;
import uk.insrt.coursework.zuul.io.SanitiseIO;
import uk.insrt.coursework.zuul.io.StandardIO;
import uk.insrt.coursework.zuul.sound.EventMusic;
import uk.insrt.coursework.zuul.sound.MusicType;
import uk.insrt.coursework.zuul.sound.SoundManager;
import uk.insrt.coursework.zuul.ui.EventDraw;
import uk.insrt.coursework.zuul.ui.TerminalEmulator;
import uk.insrt.coursework.zuul.util.BlueJ;
import uk.insrt.coursework.zuul.util.Localisation;
import uk.insrt.coursework.zuul.world.World;

/**
 * Class for managing the game loop and initialising the world.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.1-SNAPSHOT
 */
public class Game {
    public static final String GAME_NAME = "World of Deez";

    private World world;
    private IOSystem io;
    private CommandManager commands;
    private SoundManager soundManager;

    /**
     * Entrypoint to our application.
```

```java
      * @param args Arguments provided to the application
      */
    public static void main(String[] args) {
        new Game().play();
    }

    /**
     * Initialise and start the game.
     */
    public void play() {
        this.init();
        this.start();
    }

    /**
     * Initialise all required resources for the game to run.
     */
    private void init() {
        // Load audio resources.
        this.soundManager = new SoundManager();

        try {
            this.soundManager.init();
        } catch (Exception e) {
            e.printStackTrace();
            // If we can't load the sound system,
            // just play without sound.
        }

        // Determine how the game should run.
        boolean inBlueJ = BlueJ.isRunningInBlueJ();
        int selection = JOptionPane.showConfirmDialog(null, "Play full experience?\nUses custom terminal emulator.\n(recommended option)", GAME_NAME, JOptionPane.YES_NO_OPTION);
        if (selection != 1) {
            if (inBlueJ) {
                // Fullscreen minimises itself immediately
                // when running from BlueJ, not sure what's
                // going on exactly, just disabling it in general.
                selection = 1;
            } else {
                selection = JOptionPane.showConfirmDialog(null, "Immersive mode?\nRuns emulator in fullscreen.
```

```java
\n(recommended option)", GAME_NAME, JOptionPane.YES_NO_OPTION);
 82                 }
 83
 84                 this.io = new TerminalEmulator(selection == 0);
 85             } else {
 86                 this.io = new StandardIO();
 87
 88                 if (inBlueJ) {
 89                     selection = JOptionPane.showConfirmDialog(null, "Is this running from inside BlueJ?", GAME_NAME, JOption
Pane.YES_NO_OPTION);
 90                     if (selection == 0) {
 91                         this.io = new SanitiseIO(this.io);
 92                     }
 93                 }
 94             }
 95
 96             // Setup the command manager.
 97             this.commands = new CommandManager();
 98             this.commands.registerCommand(new CommandWin());
 99
100             // Register the Map command if we're in term emu mode.
101             // We draw images here so it's not available generally.
102             if (this.io instanceof TerminalEmulator) {
103                 CommandMap map = new CommandMap();
104                 this.commands.registerCommand(map);
105                 ((TerminalEmulator) this.io).getEventSystem().addListener(EventDraw.class, map);
106             }
107
108             // Load all the data we need and initialise world.
109             Localisation locale = new Localisation();
110             this.io = new LocalisedIO(this.io, locale);
111
112             // Prompt for language
113             this.io.print("Welcome! \u1F604\nBefore we start...\n\n\u1F508 Note: \u001B[35mthis game uses sound.\u001B[0m
\n\nWhat language would you like to use?\n1. :uk:Traditional English (recommended)\n2. :us:Simplified
English\n3. :de:German\nSelection: ");
114             String input = this.io.readLine();
115             String selectedLanguage = input.equals("3") ? "de_DE" : "en_GB";
116
117             try {
118                 locale.loadLocale(selectedLanguage);
```

```java
119            } catch (IOException e) {
120                System.err.println("Failed to load translations!");
121                e.printStackTrace();
122            }
123
124            // Construct the world.
125            this.world = new CampaignWorld(this.io);
126
127            // Register sound events.
128            this.soundManager.register(this.world.getEventSystem());
129            this.world.getEventSystem().emit(new EventMusic(MusicType.BgmExplore, true));
130        }
131
132        /**
133         * Start the game loop.
134         */
135        private void start() {
136            this.world.spawnPlayer();
137
138            while (true) {
139                this.io.print("> ");
140                String input = this.io.readLine().toLowerCase();
141
142                EventProcessCommand event = new EventProcessCommand(input);
143                this.world.emit(event);
144
145                if (this.commands.runCommand(this.world, event.getCommand())) {
146                    break;
147                }
148
149                this.world.emit(new EventTick());
150            }
151
152            this.io.println("Goodbye.");
153
154            try {
155                Thread.sleep(1000);
156                this.io.dispose();
157            } catch (Exception e) {}
158
159            this.soundManager.dispose();
```

```
160        }
161    }
```

```java
package uk.insrt.coursework.zuul.io;

import java.awt.Color;
import java.util.regex.Pattern;

/**
 * ANSI escape codes
 * Used https://stackoverflow.com/a/5762502 as a reference.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class Ansi {
    public static final String Reset = "\u001B[0m";
    public static final String Black = "\u001B[30m";
    public static final String Red = "\u001B[31m";
    public static final String Green = "\u001B[32m";
    public static final String Yellow = "\u001B[33m";
    public static final String Blue = "\u001B[34m";
    public static final String Purple = "\u001B[35m";
    public static final String Cyan = "\u001B[36m";
    public static final String White = "\u001B[37m";

    public static final String BackgroundBlack = "\u001B[40m";
    public static final String BackgroundRed = "\u001B[41m";
    public static final String BackgroundGreen = "\u001B[42m";
    public static final String BackgroundYellow = "\u001B[43m";
    public static final String BackgroundBlue = "\u001B[44m";
    public static final String BackgroundPurple = "\u001B[45m";
    public static final String BackgroundCyan = "\u001B[46m";
    public static final String BackgroundWhite = "\u001B[47m";

    /**
     * Regex Pattern used to match Ansi codes forwards.
     */
    public static final Pattern AnsiPattern = Pattern.compile("^\\u001B\\[(\\d{1,3})m");

    private static final Color ColorBlack = new Color(0, 0, 0);
    private static final Color ColorRed = new Color(224, 108, 117);
    private static final Color ColorGreen = new Color(152, 195, 121);
    private static final Color ColorYellow = new Color(229, 192, 123);
```

```java
42        private static final Color ColorBlue = new Color(97, 175, 239);
43        private static final Color ColorMagenta = new Color(198, 120, 221);
44        private static final Color ColorCyan = new Color(86, 182, 194);
45        private static final Color ColorWhite = new Color(255, 255, 255);
46
47        /**
48         * Convert a given escape code value, {@code (\d+?)} in {@link #AnsiPattern}, to a Color.
49         * @param value Escape code value
50         * @return Resolved Java awt Color
51         */
52        public static Color fromEscapeCode(int value) {
53            switch (value % 10) {
54                case 0: return ColorBlack;
55                case 1: return ColorRed;
56                case 2: return ColorGreen;
57                case 3: return ColorYellow;
58                case 4: return ColorBlue;
59                case 5: return ColorMagenta;
60                case 6: return ColorCyan;
61                case 7:
62                default: return ColorWhite;
63            }
64        }
65    }
```

```java
package uk.insrt.coursework.zuul.io;

/**
 * Interface representing an arbitrary IO system.
 * This can be implemented to input or output from various interfaces.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public interface IOSystem {
    /**
     * Print a string out through an arbitrary output channel.
     * @param out String to print
     */
    public void print(String out);

    /**
     * Print a string out through an arbitrary output channel and append {@code \n}.
     * @param out String to print
     */
    public void println(String out);

    /**
     * Read a String up until the first encountered {@code \n} from an arbitrary input channel.
     * @return String of line read in
     */
    public String readLine();

    /**
     * Dispose of the arbitrary input and output channels.
     */
    public void dispose();
}
```

```java
package uk.insrt.coursework.zuul.io;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

import uk.insrt.coursework.zuul.util.Localisation;

/**
 * Translate and localise any incoming output.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class LocalisedIO implements IOSystem {
    private final Pattern pattern = Pattern.compile("<([\\w\\.]+?)>");

    private IOSystem io;
    private Localisation locale;

    /**
     * Construct a new LocalisedIO.
     * @param io Provided IO system we should feed into
     * @param locale Locale to apply to any i18n strings
     */
    public LocalisedIO(IOSystem io, Localisation locale) {
        this.io = io;
        this.locale = locale;
    }

    /**
     * Replace i18n strings in any given String with their actual localised values.
     * Using replacement code from https://stackoverflow.com/a/27359491.
     * @param input String to process
     * @return Final processed string
     */
    private String replace(String input) {
        StringBuffer result = new StringBuffer();
        Matcher matcher = this.pattern.matcher(input);

        while (matcher.find()) {
            matcher.appendReplacement(result, this.locale.get(matcher.group(1)));
```

```
42              }
43
44          matcher.appendTail(result);
45          return result.toString();
46      }
47
48      @Override
49      public void print(String out) {
50          this.io.print(this.replace(out));
51      }
52
53      @Override
54      public void println(String out) {
55          this.io.println(this.replace(out));
56      }
57
58      @Override
59      public String readLine() {
60          return this.io.readLine();
61      }
62
63      @Override
64      public void dispose() {
65          this.io.dispose();
66      }
67  }
```

```java
package uk.insrt.coursework.zuul.io;

/**
 * Sanitise incoming output and remove any Ansi escape sequences.
 * This is required to print out into the BlueJ console without additional characters.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class SanitiseIO implements IOSystem {
    private final String ansiPattern = "\\u001B\\[(\\d{1,3})m";
    private IOSystem io;

    /**
     * Construct a new SanitiseIO.
     * @param io Provided IO system we should feed into
     */
    public SanitiseIO(IOSystem io) {
        this.io = io;
    }

    @Override
    public void print(String out) {
        this.io.print(out.replaceAll(this.ansiPattern, " "));
    }

    @Override
    public void println(String out) {
        this.io.println(out.replaceAll(this.ansiPattern, " "));
    }

    @Override
    public String readLine() {
        return this.io.readLine();
    }

    @Override
    public void dispose() {
        this.io.dispose();
    }
}
```

```java
1   package uk.insrt.coursework.zuul.io;
2
3   import java.util.Scanner;
4
5   /**
6    * A simple IO system implementation which feeds
7    * into System.out and takes data from System.in
8    *
9    * @author Pawel Makles (K21002534)
10   * @version 1.0-SNAPSHOT
11   */
12  public class StandardIO implements IOSystem {
13      private Scanner reader;
14
15      /**
16       * Construct a new StandardIO.
17       */
18      public StandardIO() {
19          this.reader = new Scanner(System.in);
20      }
21
22      @Override
23      public void print(String out) {
24          System.out.print(out);
25      }
26
27      @Override
28      public void println(String out) {
29          System.out.println(out);
30      }
31
32      @Override
33      public String readLine() {
34          return this.reader.nextLine();
35      }
36
37      @Override
38      public void dispose() {}
39  }
```

```java
package uk.insrt.coursework.zuul.sound;

import uk.insrt.coursework.zuul.events.Event;

/**
 * This event is used to tell the sound manager to start or stop playing music.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.1-SNAPSHOT
 */
public class EventMusic extends Event {
    private MusicType target;
    private boolean play;

    /**
     * Construct new EventMusic
     * @param target Target music type
     * @param play Whether to play or stop
     */
    public EventMusic(MusicType target, boolean play) {
        this.target = target;
        this.play = play;
    }

    /**
     * Get the music type
     * @return music type
     */
    public MusicType getTarget() {
        return this.target;
    }

    /**
     * Get whether it should play
     * @return True if it should play or false if it should stop
     */
    public boolean shouldPlay() {
        return this.play;
    }
}
```

```java
package uk.insrt.coursework.zuul.sound;

import uk.insrt.coursework.zuul.events.Event;

/**
 * Event used to tell the sound manager to play a sound.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.1-SNAPSHOT
 */
public class EventSound extends Event {
    private SoundType target;

    /**
     * Construct new EventSound
     * @param target Target sound
     */
    public EventSound(SoundType target) {
        this.target = target;
    }

    /**
     * Get sound type
     * @return Sound type
     */
    public SoundType getTarget() {
        return this.target;
    }
}
```

```java
package uk.insrt.coursework.zuul.sound;

/**
 * Music available in the game.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.1-SNAPSHOT
 */
public enum MusicType {
    Bay1,
    Bay2,
    Nature,
    City1,
    City2,

    BgmExplore,
    BgmMission,
    BgmConclusion
}
```

```java
package uk.insrt.coursework.zuul.sound;

import java.util.HashMap;

import kuusisto.tinysound.Sound;
import kuusisto.tinysound.TinySound;
import uk.insrt.coursework.zuul.events.EventSystem;

/**
 * Class used to manage background music and foreground sounds.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.1-SNAPSHOT
 */
public class SoundManager {
    private TinySound lib;
    private boolean loaded;
    private boolean initialised;

    private HashMap<SoundType, Sound> sounds;
    private HashMap<MusicType, Sound> music;

    /**
     * Construct a new SoundManager
     */
    public SoundManager() {
        this.loaded = false;
        this.initialised = false;
        this.sounds = new HashMap<>();
        this.music = new HashMap<>();
    }

    /**
     * Initialise the sound manager and load all music and sound files.
     * @throws Exception if something goes wrong.
     */
    public void init() throws Exception {
        this.lib = TinySound.init();
        this.initialised = true;

        this.sounds.put(SoundType.MoneyBag, lib.loadSound("/sounds/money-bag.ogg"));
```

```java
42          this.sounds.put(SoundType.WormHole, lib.loadSound("/sounds/worm-hole-enc.ogg"));
43
44          this.music.put(MusicType.Bay1, lib.loadSound("/sounds/bay1-enc.ogg"));
45          this.music.put(MusicType.Bay2, lib.loadSound("/sounds/bay2-enc.ogg"));
46          this.music.put(MusicType.Nature, lib.loadSound("/sounds/nature-enc.ogg"));
47          this.music.put(MusicType.City1, lib.loadSound("/sounds/city-1-quieter.ogg"));
48          this.music.put(MusicType.City2, lib.loadSound("/sounds/city-2-quieter.ogg"));
49
50          this.music.put(MusicType.BgmExplore, lib.loadSound("/sounds/bgm-explore-quieter.ogg"));
51          this.music.put(MusicType.BgmMission, lib.loadSound("/sounds/bgm-mission-quieter.ogg"));
52          this.music.put(MusicType.BgmConclusion, lib.loadSound("/sounds/bgm-conclude-quieter.ogg"));
53
54          this.loaded = true;
55      }
56
57      /**
58       * Clean up the sound library.
59       */
60      public void dispose() {
61          if (!this.initialised) return;
62          this.lib.shutdown();
63      }
64
65      /**
66       * Register sound and music events with provided event system.
67       * @param eventSystem Event system
68       */
69      public void register(EventSystem eventSystem) {
70          eventSystem.addListener(EventSound.class, event -> {
71              if (!this.loaded) return;
72              try {
73                  this.sounds.get(event.getTarget()).play();
74              } catch (Exception e) { /* ignore sound if we fail here */ }
75          });
76
77          eventSystem.addListener(EventMusic.class, event -> {
78              if (!this.loaded) return;
79              try {
80                  Sound song = this.music.get(event.getTarget());
81                  if (event.shouldPlay()) {
82                      song.play();
```

```java
            } else {
                song.stop();
            }
        } catch (Exception e) { /* ignore sound if we fail here */ }
    });
}
}
```

```java
package uk.insrt.coursework.zuul.sound;

/**
 * Sounds available in the game.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.1-SNAPSHOT
 */
public enum SoundType {
    MoneyBag,
    WormHole,
}
```

```java
package uk.insrt.coursework.zuul.ui;

import java.awt.Image;

/**
 * Representation of a single Emoji which can be rendered in the terminal emulator.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class Emoji {
    private Image image;
    private int width;
    private int height;

    /**
     * Construct a new Emoji given the image and unicode representation.
     * @param image Image to render when this Emoji is used
     * @param unicode Unicode representation of this Emoji, used to determine width
     */
    public Emoji(Image image, String unicode) {
        this.image = image;
        this.width = (int) unicode.chars().count();
        this.height = 1;
    }

    /**
     * Construct a new Emoji given the image and size constraints.
     * @param image Image to render when this Emoji is used
     * @param width Width of this Emoji
     * @param height Height of this Emoji
     */
    public Emoji(Image image, int width, int height) {
        this.image = image;
        this.width = width;
        this.height = height;
    }

    /**
     * Get the Image for this Emoji
     * @return Image
```

```java
42        */
43       public Image getImage() {
44           return this.image;
45       }
46
47       /**
48        * Get the calculated width of this Emoji
49        * @return Width
50        */
51       public int getWidth() {
52           return this.width;
53       }
54
55       /**
56        * Get the calculated height of this Emoji
57        * @return Height
58        */
59       public int getHeight() {
60           return this.height;
61       }
62   }
```

```java
package uk.insrt.coursework.zuul.ui;

import java.awt.Image;
import java.io.IOException;
import java.io.InputStream;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;

import javax.imageio.ImageIO;

import com.moandjiezana.toml.Toml;

import org.apache.commons.io.IOUtils;

import uk.insrt.coursework.zuul.util.Tree;

/**
 * Class which helps manage loading and resolving Emojis.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class EmojiManager {
    private HashMap<String, Emoji> emojis;
    private Tree<Character, String> emojiTree;
    private Tree<Character, String> currentNode;

    /**
     * Construct a new EmojiManager.
     */
    public EmojiManager() {
        this.emojis = new HashMap<>();
        this.emojiTree = new Tree<>();
        this.currentNode = this.emojiTree;
    }

    /**
     * Check whether a given Emoji is present in this manager.
```

```java
     * @param emoji Unicode representation of Emoji
     * @return True if the Emoji is available
     */
    public boolean hasEmoji(String emoji) {
        return this.emojis.containsKey(emoji);
    }

    /**
     * Get an Emoji by its Unicode representation.
     * @param emoji Unicode representation of Emoji
     * @return The Emoji or null if it doesn't exist
     */
    public Emoji getEmoji(String emoji) {
        return this.emojis.get(emoji);
    }

    /**
     * Get currently matched emoji and resets position.
     * @return Emoji if it was found, or null if not
     */
    public Emoji getEmoji() {
        String value = this.currentNode.getValue();
        Emoji emoji = this.emojis.get(value);
        this.resetState();
        return emoji;
    }

    /**
     * Reset the state of the matching mechanism.
     */
    public void resetState() {
        this.currentNode = this.emojiTree;
    }

    /**
     * The matching mechanism has not matched any characters to potential emojis.
     */
    public static final int MATCH_NONE = 0;

    /**
     * The matching mechanism has matched some characters to potential emojis.
```

```java
 83          */
 84         public static final int MATCH_SOME = 1;
 85
 86         /**
 87          * The matching mechanism has matched an emoji.
 88          */
 89         public static final int MATCH_FOUND = 2;
 90
 91         /**
 92          * Match the next character.
 93          * @param c Character to match against
 94          * @return One of {@link #MATCH_NONE}, {@link #MATCH_SOME} or {@link #MATCH_FOUND}
 95          */
 96         public int match(char c) {
 97             var child = this.currentNode.getChild(c);
 98             if (child != null) {
 99                 this.currentNode = child;
100                 if (child.getValue() == null) return MATCH_SOME;
101                 return MATCH_FOUND;
102             }
103
104             if (this.currentNode != this.emojiTree) {
105                 this.currentNode = this.emojiTree;
106                 child = this.emojiTree.getChild(c);
107                 if (child != null) {
108                     if (child.getValue() != null) return MATCH_SOME;
109                     return MATCH_FOUND;
110                 }
111             }
112
113             return MATCH_NONE;
114         }
115
116         /**
117          * Load emoji definitions and resources from a given resource directory.
118          * @param rootDir Root directory at which we expect a valid {@code definitions.toml} to exist
119          * @throws IOException if the definition file is missing or defined emojis are invalid
120          */
121         public void loadResources(String rootDir) throws IOException {
122             InputStream defnStream = this.getClass().getResourceAsStream(rootDir + "/definitions.toml");
123             // We need to force UTF-8 encoding or else unicode emojis may get mangled.
```

```java
            String defnString = IOUtils.toString(defnStream, StandardCharsets.UTF_8);
            Toml defn = new Toml().read(defnString);
            List<HashMap<String, Object>> emojis = defn.getList("emojis");

            // Load each emoji in sequence
            for (var emoji : emojis) {
                String path = (String) emoji.get("path");
                String unicode = (String) emoji.get("unicode");

                InputStream stream = this.getClass().getResourceAsStream(rootDir + "/" + path);
                Image image = ImageIO.read(stream);

                Emoji newEmoji;
                if (emoji.containsKey("width") && emoji.containsKey("height")) {
                    int width = ((Long) emoji.get("width")).intValue();
                    int height = ((Long) emoji.get("height")).intValue();
                    newEmoji = new Emoji(image, width, height);
                } else {
                    newEmoji = new Emoji(image, unicode);
                }
                this.emojis.put(unicode, newEmoji);
                this.emojiTree.addChildWithPath(
                    new ArrayList<>(Arrays.asList(
                        unicode.chars()
                            .mapToObj(c -> (char)c)
                            .toArray(Character[]::new))),
                    unicode
                );
            }
        }
    }
```

```java
package uk.insrt.coursework.zuul.ui;

import java.awt.Graphics;

import uk.insrt.coursework.zuul.events.Event;

/**
 * Event fired when the terminal emulator draws a new frame.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class EventDraw extends Event {
    private Graphics g;
    private float ox;
    private float oy;
    private float fw;
    private float fh;

    /**
     * Construct a new EventDraw Event.
     * @param g Graphics context
     * @param ox Origin X position
     * @param oy Origin Y position
     * @param fw Font character width
     * @param fh Font character height
     */
    public EventDraw(Graphics g, float ox, float oy, float fw, float fh) {
        this.g = g;
        this.ox = ox;
        this.oy = oy;
        this.fw = fw;
        this.fh = fh;
    }

    /**
     * Get the Graphics relating to this event.
     * @return Graphics
     */
    public Graphics getGraphics() {
        return this.g;
```

```java
42          }
43
44          /**
45           * Get the origin X position of the contents of the terminal.
46           * @return X position
47           */
48          public float getOriginX() {
49              return this.ox;
50          }
51
52          /**
53           * Get the origin Y position of the contents of the terminal.
54           * @return Y position
55           */
56          public float getOriginY() {
57              return this.oy;
58          }
59
60          /**
61           * Get the character width.
62           * @return Character width
63           */
64          public float getCharWidth() {
65              return this.fw;
66          }
67
68          /**
69           * Get the character height.
70           * @return Character height
71           */
72          public float getCharHeight() {
73              return this.fh;
74          }
75      }
```

```java
package uk.insrt.coursework.zuul.ui;

import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.GraphicsEnvironment;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

import javax.swing.JFrame;

/**
 * Window frame for {@link TerminalEmulator}
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class JTerminalFrame extends JFrame {
    private JTerminalView view;
    private boolean fullscreen;

    /**
     * Construct a new JTerminalFrame
     * @param emulator Terminal emulator this frame belongs to
     */
    public JTerminalFrame(TerminalEmulator emulator) {
        this.view = new JTerminalView(emulator);
        this.fullscreen = emulator.isFullscreen();
        this.makeFrame(emulator);
    }

    /**
     * Make and display all of the elements within this frame.
     * @param emulator Terminal emulator this frame belongs to
     */
    public void makeFrame(TerminalEmulator emulator) {
        this.setLayout(new BorderLayout());
        this.add(this.view);
        this.pack();
        this.setLocationRelativeTo(null);
        this.setMinimumSize(new Dimension(640, 640));
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```java
42
43          // Add a listener for any user input.
44          // https://stackoverflow.com/a/21970006
45          this.addKeyListener(new KeyListener() {
46              public void keyPressed(KeyEvent event) {
47                  int code = event.getKeyCode();
48                  switch (code) {
49                      case 8: {
50                          emulator.pop();
51                          return;
52                      }
53                      case 10: {
54                          emulator.flush();
55                          break;
56                      }
57                      default: {
58                          if ((code >= 65 && code <= 90) || (code >= 48 && code <= 57) || code == 32) {
59                              emulator.push(event.getKeyChar());
60                          }
61                      }
62                  }
63              }

65              public void keyTyped(KeyEvent e) {}
66              public void keyReleased(KeyEvent e) {}
67          });
68
69          // We are ready to display, show everything.
70          // Prerequisite to making the frame fullscreen too.
71          this.setVisible(true);
72
73          // If we are allowed to launch in fullscreen, switch to that mode now.
74          if (this.fullscreen) {
75              GraphicsEnvironment
76                  .getLocalGraphicsEnvironment()
77                  .getDefaultScreenDevice()
78                  .setFullScreenWindow(this);
79          }
80      }
81
82      @Override
```

```
83      public void dispose() {
84          super.dispose();
85          this.view.dispose();
86      }
87  }
```

```java
package uk.insrt.coursework.zuul.ui;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.FontFormatException;
import java.awt.Graphics;
import java.awt.event.ComponentAdapter;
import java.awt.event.ComponentEvent;
import java.awt.font.FontRenderContext;
import java.awt.geom.AffineTransform;
import java.io.IOException;
import java.io.InputStream;

import javax.swing.JPanel;

import com.moandjiezana.toml.Toml;

/**
 * Rendering component of {@link TerminalEmulator}
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class JTerminalView extends JPanel {
    private TerminalEmulator emulator;

    private EmojiManager emojiManager;
    private Font derivedFont;
    private Font font;

    private Thread blinkThread;
    private boolean blinkState;

    private float fw, fh, foffset, fratio;

    /**
     * Construct a TerminalView
     * @param emulator Terminal emulator this view belongs to
     */
    public JTerminalView(TerminalEmulator emulator) {
```

```java
            this.emulator = emulator;
            this.emojiManager = new EmojiManager();
            this.blinkState = false;
            this.loadResources();
            this.makeFrame();
        }

        /**
         * Prepare the terminal view for rendering.
         */
        public void makeFrame() {
            var view = this;
            this.setBackground(Color.BLACK);

            // Register a listener to repaint and adjust measurements when resizing window.
            // https://stackoverflow.com/a/8917978
            this.addComponentListener(new ComponentAdapter() {
                public void componentResized(ComponentEvent e) {
                    view.setBackground(Color.BLACK);
                    view.deriveFont();
                    view.repaint();
                }
            });

            // Start a new thread for blinking the cursor.
            this.blinkThread = new Thread("Blink Thread") {
                public void run() {
                    try {
                        while (true) {
                            view.blinkState = !view.blinkState;
                            view.repaint();

                            Thread.sleep(500);
                        }
                    } catch (InterruptedException e) {}
                }
            };

            this.blinkThread.start();
        }
```

```java
 83        /**
 84         * Load any resources required by the terminal view to render itself properly.
 85         */
 86        public void loadResources() {
 87            try {
 88                InputStream stream = this.getClass().getResourceAsStream("/emulator.toml");
 89                Toml defn = new Toml().read(stream);
 90
 91                // If a font is defined, load it.
 92                String font = defn.getString("font");
 93                if (font != null) {
 94                    this.loadFont(font, 32.0f / 12.8f);
 95                }
 96
 97                // If an emoji root directory is defined, load it.
 98                String rootDir = defn.getString("emojis");
 99                if (rootDir != null) {
100                    this.emojiManager.loadResources(rootDir);
101                }
102            } catch (Exception e) {
103                System.err.println("Failed to load any resources for terminal view!");
104                e.printStackTrace();
105            }
106        }
107
108        /**
109         * Load a specific font with a known ratio.
110         * We expect this font to be monoscape.
111         * @param source Path to the font to be loaded
112         * @param ratio Ratio of width to height for this font
113         * @throws IOException if the font cannot be loaded from a given path
114         * @throws FontFormatException if the font is of an incorrect format, we expect a TTF
115         */
116        public void loadFont(String source, float ratio) throws IOException, FontFormatException {
117            InputStream stream = this.getClass().getResourceAsStream(source);
118            this.fratio = ratio;
119            this.font = Font.createFont(Font.TRUETYPE_FONT, stream);
120        }
121
122        /**
123         * Derive the font measurements before we continue rendering.
```

```java
124          */
125         public void deriveFont() {
126             final int padding = 100;
127
128             // To find the height of the font, we take the smallest
129             // side of the window height or the proportional height
130             // found from the window width, and then we divide it
131             // by our fixed terminal height.
132             float height = Math.min(
133                 this.fratio *
134                         (float) (this.getWidth() - padding)
135                     / (float) TerminalEmulator.TERMINAL_WIDTH,
136                 (this.getHeight() - padding)
137                     / (float) TerminalEmulator.TERMINAL_HEIGHT
138             );
139
140             this.derivedFont = this.font.deriveFont(height);
141
142             // The FontRenderContext is used to determine the font dimensions
143             var frc = new FontRenderContext(new AffineTransform(), true, true);
144             var bounds = this.derivedFont.getStringBounds(" ", frc);
145
146             this.fw = (float) bounds.getWidth();
147             this.fh = (float) bounds.getHeight();
148
149             // We need to find the distance between the baseline
150             // and the ascender so we can properly align everything.
151             // https://docs.oracle.com/javase/tutorial/2d/text/fontconcepts.html
152             this.foffset = this.derivedFont.getLineMetrics(" ", frc).getAscent();
153         }
154
155         /**
156          * Dispose of this terminal view.
157          */
158         public void dispose() {
159             // We need to kill the blind thread when disposing
160             // of the UI, but that's not possible so instead I
161             // am interrupting the thread, catching that and
162             // exiting out peacefully.
163             // https://docs.oracle.com/javase/1.5.0/docs/guide/misc/threadPrimitiveDeprecation.html
164             this.blinkThread.interrupt();
```

```java
165        }
166
167        @Override
168        public Dimension getPreferredSize() {
169            return new Dimension(1280, 960);
170        }
171
172        @Override
173        protected void paintComponent(Graphics g) {
174            // https://stackoverflow.com/a/17922749
175            super.paintComponent(g);
176
177            // Setup our canvas for rendering.
178            g.setColor(Color.BLACK);
179            g.setFont(this.derivedFont);
180            g.fillRect(0, 0, this.getWidth(), this.getHeight());
181
182            // Find our topleft-most (x,y) to start rendering from.
183            TextBuffer buffer = this.emulator.getBuffer();
184            float ox = (this.getWidth() - this.fw * buffer.getWidth()) / 2;
185            float oy = (this.getHeight() - this.fh * buffer.getHeight()) / 2;
186
187            // Render each cell individually.
188            for (int y=0;y<buffer.getHeight();y++) {
189                int skipChars = 0;
190                for (int x=0;x<buffer.getWidth();x++) {
191                    // If needs be, skip chars in this line.
192                    if (skipChars > 0) {
193                        skipChars--;
194                        continue;
195                    }
196
197                    // Get the character to render.
198                    char c = buffer.getChar(x, y);
199
200                    // Match each char for emoji codepoints.
201                    // If we start to match an emoji, peek ahead.
202                    int emojiMatch = this.emojiManager.match(c);
203                    int offset = 0;
204                    while (emojiMatch == EmojiManager.MATCH_SOME) {
205                        emojiMatch = this.emojiManager.match(buffer.getChar(x + ++offset, y));
```

```java
206                }
207
208                // Get this cell's background and foreground colours.
209                Color bg = buffer.getBg(x, y);
210                Color fg = buffer.getFg(x, y);
211
212                // Find this char's offset.
213                int drawX = Math.round(ox + this.fw * x);
214                int drawY = Math.round(oy + this.fh * y);
215
216                // Draw rect if there's a background present.
217                if (bg != null && bg != Color.BLACK) {
218                    g.setColor(bg);
219                    g.fillRect(drawX, drawY, (int) Math.ceil(this.fw), (int) Math.ceil(this.fh));
220                }
221
222                // If we're drawing an emoji, get the image and skip text.
223                if (emojiMatch == EmojiManager.MATCH_FOUND) {
224                    Emoji emoji = this.emojiManager.getEmoji();
225                    g.drawImage(
226                        emoji.getImage(),
227                        drawX, drawY,
228                        Math.round(this.fw * emoji.getWidth()),
229                        Math.round(this.fh * emoji.getHeight()),
230                        this
231                    );
232
233                    skipChars = offset;
234                    continue;
235                }
236
237                // Drawing the char if it's not a space, we have to
238                // take care to add the offset we previously found or
239                // otherwise the distance between the baseline and
240                // the ascender. This is because Graphics.drawString
241                // draws text from the leftmost baseline equal to (x,y).
242                if (c != 0 && c != ' ') {
243                    g.setColor(fg);
244                    g.drawString(
245                        String.valueOf(c),
246                        drawX,
```

```java
                       Math.round(drawY + this.foffset)
                  );
               }
            }

            this.emojiManager.resetState();
        }

        // Draw a blinker to indicate the user can type here.
        // Offset it a bit to not interfere with any text on-screen.
        if (this.blinkState) {
            g.setColor(Color.WHITE);
            g.fillRect(
                (int) (ox + this.fw * buffer.getPosX() + 1),
                (int) (oy + this.fh * buffer.getPosY() + this.foffset),
                (int) this.fw - 2,
                (int) (this.fh / 8)
            );
        }

        // Fire draw event for custom rendering.
        this.emulator.getEventSystem().emit(new EventDraw(g, ox, oy, this.fw, this.fh));
    }
}
```

```java
package uk.insrt.coursework.zuul.ui;

import java.awt.Color;
import java.awt.EventQueue;
import java.util.concurrent.BlockingQueue;
import java.util.concurrent.LinkedBlockingQueue;

import uk.insrt.coursework.zuul.events.EventSystem;
import uk.insrt.coursework.zuul.io.IOSystem;

/**
 * A terminal emulator which implements an IO system to
 * be arbitrarily plugged into any existing components.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class TerminalEmulator implements IOSystem {
    public static final int TERMINAL_WIDTH = 80;
    public static final int TERMINAL_HEIGHT = 25;

    private BlockingQueue<String> queue;
    private EventSystem eventSystem;
    private JTerminalFrame frame;
    private boolean fullscreen;
    private TextBuffer buffer;
    private String input;

    /**
     * Construct and build a new TerminalEmulator
     * @param fullscreen Whether to launch the emulator in fullscreen
     */
    public TerminalEmulator(boolean fullscreen) {
        this.queue = new LinkedBlockingQueue<>();
        this.eventSystem = new EventSystem();
        this.buffer = new TextBuffer(TERMINAL_WIDTH, TERMINAL_HEIGHT);
        this.fullscreen = fullscreen;
        this.input = new String();

        this.buildFrame();
    }
```

```java
42
43      /**
44       * Construct a new TerminalEmulator and force windowed mode.
45       */
46      public TerminalEmulator() {
47          this(false);
48      }
49
50      /**
51       * Build and show the terminal emulator.
52       */
53      public void buildFrame() {
54          var emulator = this;
55          EventQueue.invokeLater(new Runnable() {
56              @Override
57              public void run() {
58                  emulator.frame = new JTerminalFrame(emulator);
59              }
60          });
61      }
62
63      /**
64       * Get the local event system for this emulator.
65       * @return The event system
66       */
67      public EventSystem getEventSystem() {
68          return this.eventSystem;
69      }
70
71      /**
72       * Get the emulator's text buffer.
73       * @return The text buffer
74       */
75      public TextBuffer getBuffer() {
76          return this.buffer;
77      }
78
79      /**
80       * Tell the terminal frame to repaint contents.
81       */
82      private void repaint() {
```

```java
 83            if (this.frame != null) {
 84                this.frame.repaint();
 85            }
 86        }
 87
 88        /**
 89         * Check whether we are in fullscreen mode.
 90         * @return True if we are fullscreen
 91         */
 92        public boolean isFullscreen() {
 93            return this.fullscreen;
 94        }
 95
 96        /**
 97         * Push a new character to the input buffer.
 98         * @param c Character to push
 99         */
100        public void push(char c) {
101            if (this.buffer.getPosX() + 1 == TERMINAL_WIDTH) return;
102
103            this.input += c;
104            this.buffer.write(new String(new char[] { c }));
105            this.buffer.setLastFg(Color.GRAY);
106            this.repaint();
107        }
108
109        /**
110         * Pop last character from the input buffer.
111         */
112        public void pop() {
113            if (this.input.length() > 0) {
114                this.input = this.input.substring(0, this.input.length() - 1);
115                this.buffer.backspace();
116                this.repaint();
117            }
118        }
119
120        /**
121         * Flush input from terminal emulator thread and send it to whatever is
122         * waiting for it on another thread. Uses a blocking queue to send data
123         * between threads, as seen here: https://stackoverflow.com/a/23413506
```

```java
124          */
125         public void flush() {
126             this.queue.add(this.input);
127             this.input = "";
128         }
129
130         @Override
131         public void print(String out) {
132             buffer.write(out);
133             this.repaint();
134         }
135
136         @Override
137         public void println(String out) {
138             buffer.write(out + "\n");
139             this.repaint();
140         }
141
142         @Override
143         public String readLine() {
144             try {
145                 String line = this.queue.take();
146                 this.print("\n");
147                 return line;
148             } catch (Exception err) {
149                 err.printStackTrace();
150                 System.exit(1);
151                 return " ";
152             }
153         }
154
155         @Override
156         public void dispose() {
157             this.frame.dispose();
158         }
159     }
```

```java
package uk.insrt.coursework.zuul.ui;

import java.awt.Color;
import java.util.regex.Matcher;

import uk.insrt.coursework.zuul.io.Ansi;

/**
 * Representation of a text and colour buffer.
 * Provides various utilities for manipulating text on screen.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class TextBuffer {
    private char[][] buffer;
    private Color[][] bufferBg;
    private Color[][] bufferFg;

    private int width;
    private int height;

    private int posX;
    private int posY;

    private Color bg;
    private Color fg;

    private boolean overflow;

    /**
     * Construct a new TextBuffer with given constraints.
     * @param width Buffer width
     * @param height Buffer height
     */
    public TextBuffer(int width, int height) {
        this.buffer = new char[height][width];
        this.bufferBg = new Color[height][width];
        this.bufferFg = new Color[height][width];

        this.width = width;
```

```java
42              this.height = height;
43
44              this.posX = 0;
45              this.posY = 0;
46
47              this.bg = Color.BLACK;
48              this.fg = Color.WHITE;
49
50              this.overflow = false;
51          }
52
53          /**
54           * Remove top-most row and move all the other rows up.
55           */
56          public void shift() {
57              for (int i=0;i<this.height-1;i++) {
58                  this.buffer[i] = this.buffer[i + 1];
59                  this.bufferBg[i] = this.bufferBg[i + 1];
60                  this.bufferFg[i] = this.bufferFg[i + 1];
61              }
62
63              this.bufferBg[this.height - 1] = new Color[this.width];
64              this.bufferFg[this.height - 1] = new Color[this.width];
65              this.buffer[this.height - 1] = new char[this.width];
66          }
67
68          /**
69           * Remove the last previous character written to the buffer.
70           */
71          public void backspace() {
72              if (this.posX == 0) return;
73              this.buffer[this.posY][--this.posX] = ' ';
74          }
75
76          /**
77           * Retroactively set the foreground for the previous character written.
78           * @param color Foreground colour
79           */
80          public void setLastFg(Color color) {
81              this.bufferFg[this.posY][this.posX - 1] = color;
82          }
```

```java
 83
 84      /**
 85       * Retroactively set the background for the previous character written.
 86       * @param color Background colour
 87       */
 88      public void setLastBg(Color color) {
 89          this.bufferBg[this.posY][this.posX - 1] = color;
 90      }
 91
 92      /**
 93       * Write a new character to the text buffer.
 94       * This will move the cursor forwards.
 95       * @param c Character to write
 96       */
 97      public void write(char c) {
 98          // If we encounter a newline, shift downwards or go on to a new line.
 99          if (c == '\n') {
100              if (this.overflow) {
101                  this.overflow = false;
102                  return;
103              }
104
105              this.posX = 0;
106
107              if (this.posY == this.height - 1) {
108                  this.shift();
109              } else {
110                  this.posY++;
111              }
112
113              return;
114          }
115
116          // Clear any overflow value.
117          this.overflow = false;
118
119          // Commit new character to buffer.
120          this.bufferBg[this.posY][this.posX] = this.bg;
121          this.bufferFg[this.posY][this.posX] = this.fg;
122          this.buffer[this.posY][this.posX++] = c;
123
```

```java
124            // If we're at the end of the line, shift downwards or move to new line.
125            if (this.posX == this.width) {
126                this.posX = 0;
127
128                if (this.posY == this.height - 1) {
129                    this.shift();
130                } else {
131                    this.posY++;
132                }
133
134                // Set a flag to say we just naturally overflowed and to ignore
135                // the next newline character that may appear.
136                this.overflow = true;
137            }
138        }
139
140        /**
141         * Write a string value to the text buffer.
142         * @param value String value to write
143         */
144        public void write(String value) {
145            // Write each character sequentially.
146            for (int i=0;i<value.length();i++) {
147                char c = value.charAt(i);
148
149                // If we encounter an Ansi escape character, then take the
150                // substring from this point on and determine if it is a valid
151                // escape code. If it is, apply any changes before continuing.
152                if (c == '\u001B') {
153                    Matcher matcher = Ansi.AnsiPattern.matcher(value.substring(i));
154                    if (matcher.find()) {
155                        int v = Integer.parseInt(matcher.group(1));
156                        i += 3 + (v > 9 ? 1 : 0);
157
158                        if (v == 0) {
159                            this.bg = Color.BLACK;
160                            this.fg = Color.WHITE;
161                        } else if (v >= 30 && v < 38) {
162                            this.fg = Ansi.fromEscapeCode(v);
163                        } else if (v >= 40 && v < 48) {
164                            this.bg = Ansi.fromEscapeCode(v);
```

```java
				}

				continue;
			}
		}

		this.write(c);
	}
}

/**
 * Get a character at a certain position
 * @param x X position
 * @param y Y position
 * @return Character at given position
 */
public char getChar(int x, int y) {
	return this.buffer[y][x];
}

/**
 * Get the background colour at a certain position
 * @param x X position
 * @param y Y position
 * @return Background colour at given position
 */
public Color getBg(int x, int y) {
	return this.bufferBg[y][x];
}

/**
 * Get the foreground colour at a certain position
 * @param x X position
 * @param y Y position
 * @return Foreground colour at given position
 */
public Color getFg(int x, int y) {
	return this.bufferFg[y][x];
}

/**
```

```java
         * Get the width of this buffer.
         * @return Width
         */
        public int getWidth() {
            return this.width;
        }

        /**
         * Get the height of this buffer.
         * @return Height
         */
        public int getHeight() {
            return this.height;
        }

        /**
         * Get the current X position of the cursor.
         * @return X position of cursor
         */
        public int getPosX() {
            return this.posX;
        }

        /**
         * Get the current Y position of the cursor.
         * @return Y position of cursor
         */
        public int getPosY() {
            return this.posY;
        }
    }
```

```java
package uk.insrt.coursework.zuul.util;

import java.lang.reflect.Field;
import java.net.URLClassLoader;
import java.util.Vector;

/**
 * Utilities for detecting we are running in BlueJ.
 *
 * @author Paul Makles <https://insrt.uk>
 * @version 2.0
 */
public class BlueJ {
    /**
     * Whether to ignore deprecation warnings.
     * Enable to allow isRunningInBlueJ() to confidently determine status.
     */
    private static boolean liveOnTheEdge = false;

    /**
     * Check whether this is being exported as BlueJ using maven-bluej
     * https://github.com/KCLOSS/maven-bluej
     * @return Whether this was exported as a BlueJ project.
     */
    public static boolean isExportedAsBlueJ() {
        return BlueJ.class.getResource("/ThisIsABlueJProject") != null;
    }

    /**
     * Detect whether we are currently running under BlueJ.
     * @return Whether we are running from BlueJ.
     */
    public static boolean isRunningInBlueJ() {
        ClassLoader classLoader = Thread.currentThread().getContextClassLoader();

        // When we load the project typically, i.e. from a JAR file, it is instead
        // loaded by jdk.internal.loader.ClassLoaders$PlatformClassLoader and then
        // $AppClassLoader, which we should also see further up the chain from the
        // java.net.URLClassLoader loader.
        if (classLoader instanceof URLClassLoader) {
            if (getJavaVersion() > 8 && !liveOnTheEdge) {
```

```java
42              // Using setAccessible() as below is deprecated in Java 9 onwards,
43              // so to avoid any errors in stderr, we can take a safe bet and
44              // assume that we are in BlueJ given the way we are being loaded.
45              return true;
46          }
47
48          // We can verify we are running under BlueJ by looping through all
49          // classes which exist on the parent class loader and to check if
50          // a BlueJ class is present.
51          try {
52              // Finding classes loaded by ClassLoader.
53              // https://stackoverflow.com/a/10261850
54              Field f = ClassLoader.class.getDeclaredField("classes");
55              f.setAccessible(true);
56
57              @SuppressWarnings("unchecked")
58              Vector<Class<?>> classes = (Vector<Class<?>>) f.get(classLoader.getParent());
59
60              for (Class<?> cls : classes) {
61                  if (cls.getName().startsWith("bluej.runtime")) {
62                      return true;
63                  }
64              }
65          } catch (NoSuchFieldException | IllegalAccessException | ClassCastException e) {}
66      }
67
68      return false;
69  }
70
71  /**
72   * Gets the current Java version as a single integer.
73   * Taken from https://stackoverflow.com/a/2591122
74   * @return Current Java major version number.
75   */
76  private static int getJavaVersion() {
77      String version = System.getProperty("java.version");
78      return Integer.parseInt(
79          version.startsWith("1.")
80              ? version.substring(2, 3)
81              : (
82                  version.indexOf(".") != -1
```

```
83                          ? version.substring(0, version.indexOf("."))
84                          : version
85                  )
86          );
87      }
88  }
```

```java
package uk.insrt.coursework.zuul.util;

import java.io.IOException;
import java.io.InputStream;
import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;
import java.util.stream.Collectors;

import com.moandjiezana.toml.Toml;

/**
 * This class provides localisation capabilities for the game.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class Localisation {
    private Map<String, Object> map;

    /**
     * Construct a new instance of Localisation.
     */
    public Localisation() {
        this.map = new HashMap<>();
    }

    /**
     * Load a certain locale by name.
     * @param locale The target locale to load
     * @throws IOException if the locale does not exist in resources
     */
    public void loadLocale(String locale) throws IOException {
        InputStream stream = this
            .getClass()
            .getResourceAsStream("/locale/" + locale + ".toml");

        this.map = new Toml().read(stream).toMap();
    }

    /**
```

```java
42        * Given a path of keys, find the value at the end of the path.
43        *
44        * Unchecked errors are supressed as they would only occur if the
45        * developer provides an incorrect data structure, in that case the
46        * error will be emitted from within this method. It is not a critical
47        * error but it should be handled immediately.
48        * @param path Path to value we want
49        * @return The value at the given path
50        */
51       @SuppressWarnings("unchecked")
52       public String from(String... path) {
53           if (path.length == 0) return "<empty string>";
54
55           try {
56               var index = 1;
57               var node = this.map.get(path[0]);
58               while (index != path.length) {
59                   Map<String, Object> map = (Map<String, Object>) node;
60                   node = map.get(path[index++]);
61               }
62
63               if (node != null) {
64                   return (String) node;
65               }
66           } catch (Exception e) {
67               // We don't want this to be a fatal error,
68               // we instead return the original template.
69           }
70
71           return "<" + Arrays.asList(path).stream().collect(Collectors.joining(".")) + ">";
72       }
73
74       /**
75        * Given a path, find the value at the end of the path.
76        * @param path Path to value we want, keys separated by period
77        * @return The value at the given path
78        */
79       public String get(String path) {
80           return this.from(path.split("\\."));
81       }
82   }
```

```java
package uk.insrt.coursework.zuul.util;

import java.util.Arrays;

import uk.insrt.coursework.zuul.entities.Entity;

/**
 * Utilities for searching through data structures related to the game
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class Search {
    /**
     * Find an Entity within an Iterable of entities given certain parameters.
     * @param entities Iterable of Entities which we search through
     * @param name Query which we are matching for
     * @param fuzzy Whether to match whether the alias contains this name in contrast to just doing exact matching
     * @return The Entity if it is found or null
     */
    public static Entity findEntity(Iterable<Entity> entities, String name, boolean fuzzy) {
        String normalised = name.toLowerCase();
        for (Entity entity : entities) {
            String[] aliases = entity.getAliases();
            for (String alias : aliases) {
                if (fuzzy) {
                    if (Arrays.asList(normalised.split("\\s"))
                        .contains(alias)) {
                        return entity;
                    }
                } else if (normalised.equals(alias)) {
                    return entity;
                }
            }
        }

        return null;
    }
}
```

```java
package uk.insrt.coursework.zuul.util;

import java.util.HashMap;
import java.util.List;

/**
 * This is an implementation of a Tree-like data structure.
 * Each node has a one or more children identified by a key
 * and each node can have more children or have a value.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class Tree<K, V> {
    private HashMap<K, Tree<K, V>> children = new HashMap<>();
    private Tree<K, V> parent;
    private V value;

    /**
     * Construct a new Empty Tree node.
     */
    public Tree() {}

    /**
     * Construct a new Tree node with a parent only.
     * @param parent Tree node which owns this node
     */
    public Tree(Tree<K, V> parent) {
        this.parent = parent;
    }

    /**
     * Construct a new Tree node with parent and value.
     * @param parent Tree node which owns this node
     * @param value The value this node should hold
     */
    public Tree(Tree<K, V> parent, V value) {
        this.parent = parent;
        this.value = value;
    }
```

```java
42          /**
43           * Get a child of this Tree node with a given key K.
44           * @param key Given key
45           * @return The child represented by this key if it exists, otherwise returns null
46           */
47          public Tree<K, V> getChild(K key) {
48              return this.children.get(key);
49          }
50
51          /**
52           * Private method used to accumulate the edges travelled up to the root node.
53           * @param acc Accumulator value
54           * @return The current accumulator value
55           */
56          private int getHeight(int acc) {
57              if (this.parent == null) return acc;
58              return this.parent.getHeight(++acc);
59          }
60
61          /**
62           * The height of the Tree from this point.
63           * This is the number of edges to get from this node to the root node.
64           * @return The height of the tree
65           */
66          public int getHeight() {
67              return this.getHeight(0);
68          }
69
70          /**
71           * Whether this Tree node has a value.
72           * @return True if this node has a value
73           */
74          public boolean hasValue() {
75              return this.value != null;
76          }
77
78          /**
79           * Get the value of this Tree node.
80           * @return Value stored if there is one, otherwise null.
81           */
82          public V getValue() {
```

```java
 83                return this.value;
 84            }
 85
 86            /**
 87             * Add a child to this Tree node represented by a key K.
 88             * @param key Key to represent this new child
 89             * @param node Child node to add
 90             */
 91            public void addChild(K key, Tree<K, V> node) {
 92                this.children.put(key, node);
 93            }
 94
 95            /**
 96             * Recurse through a given key path and add value as a node at the bottom of the path.
 97             * @param keys Keys to iterate through
 98             * @param value Value to add at the end of the path
 99             */
100            public void addChildWithPath(List<K> keys, V value) {
101                Tree<K, V> node = this;
102                while (keys.size() > 0) {
103                    K key = keys.remove(0);
104                    Tree<K, V> child = node.getChild(key);
105                    if (child == null) {
106                        child = new Tree<>(node, keys.size() == 0 ? value : null);
107                        node.addChild(key, child);
108                    }
109
110                    node = child;
111                }
112            }
113        }
```

```java
package uk.insrt.coursework.zuul.world;

import java.util.Arrays;
import java.util.List;

/**
 * Enum which represents a Cardinal direction.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public enum Direction {
    NORTH(new String[] { "N" }),
    NORTH_EAST(new String[] { "NE", "NORTH EAST" }),
    EAST(new String[] { "E" }),
    SOUTH_EAST(new String[] { "SE", "SOUTH EAST" }),
    SOUTH(new String[] { "S" }),
    SOUTH_WEST(new String[] { "SW", "SOUTH WEST" }),
    WEST(new String[] { "W" }),
    NORTH_WEST(new String[] { "NW", "NORTH WEST" }),

    UP(new String[] {}),
    DOWN(new String[] {});

    private List<String> aliases;

    /**
     * Consturct a new Direction
     * @param aliases Alternative ways to refer to this Direction
     */
    private Direction(String[] aliases) {
        this.aliases = Arrays.asList(aliases);
    }

    /**
     * Check whether this Direction matches the given aliases.
     * @param direction Direction in String format
     * @return Whether it matches.
     */
    private boolean matches(String direction) {
        return this.aliases.contains(direction);
```

```java
42          }

43

44          /**
45           * Flip a given Direction in the opposite direction.
46           * @return Direction in the opposite direction.
47           */
48          public Direction flip() {
49              switch (this) {
50                  default:
51                  case NORTH: return Direction.SOUTH;
52                  case NORTH_EAST: return Direction.SOUTH_WEST;
53                  case EAST: return Direction.WEST;
54                  case SOUTH_EAST: return Direction.NORTH_WEST;
55                  case SOUTH: return Direction.NORTH;
56                  case SOUTH_WEST: return Direction.NORTH_EAST;
57                  case WEST: return Direction.EAST;
58                  case NORTH_WEST: return Direction.SOUTH_EAST;
59                  case UP: return Direction.DOWN;
60                  case DOWN: return Direction.UP;
61              }
62          }

63

64          /**
65           * Convert an arbitrary String to a Direction.
66           * @param direction Raw string representing a Direction
67           * @return Direction or null from given string
68           */
69          public static Direction fromString(String direction) {
70              if (direction == null) return null;

71

72              String directionFormatted = direction.toUpperCase();
73              try {
74                  return Direction.valueOf(directionFormatted);
75              } catch (Exception ex) {
76                  for (Direction dir : Direction.values()) {
77                      if (dir.matches(directionFormatted)) {
78                          return dir;
79                      }
80                  }

81

82                  return null;
```

```
83            }
84        }
85    }
```

```java
package uk.insrt.coursework.zuul.world;

import uk.insrt.coursework.zuul.entities.Inventory;

/**
 * Representation of a physical location in the world,
 * whether it is a room or inventory or neither but not both.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public class Location {
    private Room room;
    private Inventory inventory;

    /**
     * Construct a new Location outside of the World.
     */
    public Location() {}

    /**
     * Construct a new Location pointing to a Room.
     * @param room Room
     */
    public Location(Room room) {
        this.room = room;
    }

    /**
     * Construct a new Location pointing to an Inventory.
     * @param inventory Inventory
     */
    public Location(Inventory inventory) {
        this.inventory = inventory;
    }

    /**
     * Change this Location to point to a Room.
     * @param room Room
     */
    public void setLocation(Room room) {
```

```java
42              this.room = room;
43              this.inventory = null;
44          }
45
46          /**
47           * Change this Location to point to an Inventory.
48           * @param inventory Inventory
49           */
50          public void setLocation(Inventory inventory) {
51              this.room = null;
52              this.inventory = inventory;
53          }
54
55          /**
56           * Reset the Location and put us outside of the World.
57           */
58          public void clear() {
59              this.room = null;
60              this.inventory = null;
61          }
62
63          /**
64           * Get the current Room this Location represents.
65           * @return Room or null if in an inventory or out of the World.
66           */
67          public Room getRoom() {
68              return this.room;
69          }
70
71          /**
72           * Get the current Inventory this Location represents.
73           * @return Inventory or null if in a room or out of this World.
74           */
75          public Inventory getInventory() {
76              return this.inventory;
77          }
78      }
```

```java
package uk.insrt.coursework.zuul.world;

import java.util.HashMap;
import java.util.Set;

/**
 * Representation of a Room within the World.
 * Handles how entities can move from this to other Rooms.
 *
 * @author Pawel Makles (K21002534)
 * @version 1.0-SNAPSHOT
 */
public abstract class Room {
    private World world;
    private String name;
    private HashMap<Direction, Room> adjacentRooms;

    /**
     * Construct a new Room in a given World with a given name.
     * @param world World
     * @param name Internal name used to refer to this Room
     */
    public Room(World world, String name) {
        this.world = world;
        this.name = name;
        this.adjacentRooms = new HashMap<>();
    }

    /**
     * Get the World that this Room is in.
     * @return World
     */
    public World getWorld() {
        return this.world;
    }

    /**
     * Get the internal name of this Room.
     * @return Internal name
     */
    public String getName() {
```

```
42              return this.name;
43          }
44
45          /**
46           * Make another Room adjacent to this Room in a particular Direction.
47           * @param direction Direction the other Room is in
48           * @param room Room which we are making adjacent
49           */
50          public void setAdjacent(Direction direction, Room room) {
51              if (room == null) System.err.println("Warning: assigned null Room to direction " + direction + " for the Room " +
this.name);
52              this.adjacentRooms.put(direction, room);
53          }
54
55          /**
56           * Get an adjacent Room in a particular Direction.
57           * @param direction Direction to look at
58           * @return The Room if one is present in that Direction, otherwise null
59           */
60          public Room getAdjacent(Direction direction) {
61              return this.adjacentRooms.get(direction);
62          }
63
64          /**
65           * Whether the player can leave in any particular direction.
66           * Should print reason if not.
67           * @param direction Direction which we are checking
68           * @return Whether the player can leave
69           */
70          public boolean canLeave(Direction direction) {
71              return true;
72          }
73
74          /**
75           * Get Directions that you can leave this Room in.
76           * @return Set of Directions we can leave in
77           */
78          public Set<Direction> getDirections() {
79              return this.adjacentRooms.keySet();
80          }
81
```

```java
 82        /**
 83         * Check whether there is an exit in a particular Direction.
 84         * @param direction Direction to check
 85         * @return True if there is an exit in a given Direction
 86         */
 87        public boolean hasExit(Direction direction) {
 88            return this.adjacentRooms.containsKey(direction);
 89        }
 90
 91        /**
 92         * Reset adjacent Rooms and reconfigure adjacent Rooms.
 93         * This should be called after all Rooms have been spawned into the World.
 94         */
 95        public void linkRooms() {
 96            this.adjacentRooms.clear();
 97            this.setupDirections();
 98        }
 99
100        /**
101         * Spawn Entities in this World.
102         * By default, nothing is done but this should be used further up to spawn
103         * the Entities for this particular Room.
104         */
105        public void spawnEntities() {}
106
107        /**
108         * Convert this Room into a Location.
109         * @return Location representation of Room
110         */
111        public Location toLocation() {
112            return new Location(this);
113        }
114
115        /**
116         * Describe what this Room looks like.
117         * @return Description of this Room
118         */
119        public abstract String describe();
120
121        /**
122         * Setup adjacent Rooms.
```

```
123          */
124     protected abstract void setupDirections();
125 }
```

```java
1    package uk.insrt.coursework.zuul.world;
2
3    import java.util.HashMap;
4    import java.util.List;
5    import java.util.Map;
6    import java.util.stream.Collectors;
7
8    import uk.insrt.coursework.zuul.entities.Entity;
9    import uk.insrt.coursework.zuul.entities.EntityPlayer;
10   import uk.insrt.coursework.zuul.events.Event;
11   import uk.insrt.coursework.zuul.events.EventSystem;
12   import uk.insrt.coursework.zuul.io.IOSystem;
13
14   /**
15    * Representation of the game World.
16    * Contains all the Rooms and Entities as well as the Player.
17    * Has its own Event system for signaling when things should happen.
18    * Also has access to the IO system which is provided to all Rooms and Entities.
19    *
20    * @author Pawel Makles (K21002534)
21    * @version 1.0-SNAPSHOT
22    */
23   public class World {
24       protected Map<String, Room> rooms = new HashMap<>();
25       protected Map<String, Entity> entities = new HashMap<>();
26       protected EntityPlayer player;
27
28       protected IOSystem io;
29       protected EventSystem eventSystem;
30
31       /**
32        * Consturct a new game World with a given IO system.
33        * @param io IO system to provide to everything
34        */
35       public World(IOSystem io) {
36           this.io = io;
37           this.eventSystem = new EventSystem();
38           this.player = new EntityPlayer(this);
39           this.entities.put("player", this.player);
40       }
41
```

```java
42          /**
43           * Find an Entity by its ID.
44           * @param id Entity ID
45           * @return Entity if it exists, otherwise null.
46           */
47          public Entity getEntity(String id) {
48              return this.entities.get(id);
49          }
50
51          /**
52           * Find an Room by its ID.
53           * @param room Room ID
54           * @return Room if it exists, otherwise null.
55           */
56          public Room getRoom(String room) {
57              return this.rooms.get(room);
58          }
59
60          /**
61           * Get the Player entity.
62           * @return The player entity
63           */
64          public EntityPlayer getPlayer() {
65              return this.player;
66          }
67
68          /**
69           * Get the IO system provided to this World.
70           * @return IO system
71           */
72          public IOSystem getIO() {
73              return this.io;
74          }
75
76          /**
77           * Get this World's event system.
78           * @return World event system
79           */
80          public EventSystem getEventSystem() {
81              return this.eventSystem;
82          }
```

```java
 83
 84        /**
 85         * Add a Room to this World.
 86         * @param room Room to add
 87         */
 88        protected void addRoom(Room room) {
 89            this.rooms.put(room.getName(), room);
 90        }
 91
 92        /**
 93         * Spawn a new Entity in the World.
 94         * @param id Unique Entity ID
 95         * @param entity Entity to spawn
 96         */
 97        public void spawnEntity(String id, Entity entity) {
 98            this.entities.put(id, entity);
 99        }
100
101        /**
102         * Get all the Entities found in a given Room.
103         * @param room Room to search for
104         * @return List of Entities in the World in a given Room
105         */
106        public List<Entity> getEntitiesInRoom(Room room) {
107            return this
108                .entities
109                .values()
110                .stream()
111                .filter(e -> e.getRoom() == room)
112                .collect(Collectors.toList());
113        }
114
115        protected void linkRooms() {
116            for (Room room : this.rooms.values()) {
117                room.linkRooms();
118            }
119        }
120
121        public void emit(Event event) {
122            this.eventSystem.emit(event);
123        }
```

```java
124
125      /**
126       * Try to spawn the player in the first available room.
127       */
128      public void spawnPlayer() {
129          this.player.setLocation(this.rooms.values().iterator().next());
130      }
131  }
```

```toml
emojis = [
    { unicode = "\u1F633", path = "flosh.png" }, #  😳
    { unicode = "\u1F601", path = "trol.png" }, #  😁
    { unicode = "\u1F438", path = "monkaStare.png" }, #  🐸
    { unicode = "\u1F642", path = "pauseChamp.png" }, #  🙂
    { unicode = "\u1F610", path = "weirdChamp.png" }, #  😐
    { unicode = "\u1F604", path = "peepoHappy.png" }, #  😄
    { unicode = "\u1F508", path = "sound.png" }, #  🔈
    { unicode = "\u1F99D", path = "they is stuck.jpg", width = 24, height = 8 }, #  🦝
    { unicode = ":uk:", path = "uk.png", width = 3, height = 1 }, #  🇬🇧
    { unicode = ":us:", path = "us.png", width = 3, height = 1 }, #  🇺🇸
    { unicode = ":de:", path = "de.png", width = 3, height = 1 }, #  🇩🇪

    # Example definitions:
    #{ unicode = "[wideChamp]", path = "weirdChamp.png" },
    #{ unicode = "[widePeepo]", path = "peepoHappy.png" },
    #{ unicode = "😄", path = "peepoHappy.png", width = 8, height = 4 },
]
```

```toml
# Translations for World of Deez
# By Infi

[global]
sight = "Du kannst Folgendes sehen:"

    [global.can_go_in_x_directions]
    1 = "Du kannst in"
    2 = "Richtungen gehen"

[selectors]
direction = "<Richtung>"
something = "<Etwas>"
someone = "<Jemand>"
item = "<Item>"

    [selectors.cant_find]
    1 = "Du schaust nach"
    2 = ", aber du findest nichts"

[commands]
unknown = "Keine Ahnung, was du gerade versuchst."

back = "Zurück zum vorherigen Raum"
quit = "Spiel beenden"
where_am_i = "Das Zimmer nochmal beschreiben"

    [commands.bag]
    usage = "Schau in deinen Beutel oder nach dem Inventar von etwas"
    cant_find = "Was du suchst kann nicht gefunden werden"
    empty = "Dein Beutel ist leer!"
    entity_empty = "hat nichts im Inventar"
    can_carry_kg = "Deine Tragekraft beträgt"
    are_carrying_kg = "Du trägst"
    look_in_bag = "Du schaust in deinen Beutel und siehst folgendes:"
    entity_appears_to_have = "hat anscheinend"

    [commands.drop]
    usage = "Lege ein Item ab"
    nothing_specified = "Was willst du ablegen?"
```

```toml
    [commands.drop.dropped]
    1 = "Du legst"
    2 = "ab"

[commands.give]
usage = "Etwas an jemanden geben"
nothing_specified = "Was willst du geben?"
no_target = "Wem oder in was willst du das geben?"
denied_player = "Du kannst dich selbst nicht selber geben \u1F438"

    [commands.give.denied]
    1 = "Du kannst"
    2 = "nicht geben an"

[commands.go]
usage = "Gehe in eine bestimmte Richtung"
nothing_specified = "Wohin gehst du?"

[commands.help]
usage = "Hilfemenü anzeigen"
can_run = "Du kannst die folgenden Befehle ausführen:"

[commands.pet]
usage = "Streichel etwas in deiner Umgebung oder deinem Inventar"
nothing_specified = "Was willst du streicheln?"
denied = "Du kannst nicht folgendes streicheln:"

[commands.take]
usage = "Nimm ein Item auf"
nothing_specified = "Von wem willst du etwas nehmen?"
entity_does_not_have_entity = "hat nicht"
item_not_specified = "Was willst du nehmen?"

    [commands.take.took]
    1 = "Du nimmst"
    2 = "von"
    3 = "und machst es in deinen Beutel"

    [commands.take.denied]
    1 = "Du kannst"
    2 = "nicht nehmen, es ist zu schwer für dich und deinen Beutel"
```

```toml
83
84      [commands.talk]
85      usage = "Fang an, mit jemandem zu reden"
86      nothing_specified = "Mit was willst du reden?"
87      denied = "Du kannst nicht reden mit:"
88
89      [commands.use]
90      usage = "Benutze ein Item in deiner Umgebung oder deinem Inventar"
91      nothing_specified = "Was willst du benutzen?"
92      denied = "Du kannst nicht folgendes benutzen:"
93
94      [commands.map]
95      usage = "Zeige die Weltkarte"
96      close = "Drück Enter, um die Weltkarte zu schließen"
97
98          [commands.map.discovered]
99          1 = "Du hast"
100         2 = "der Weltkarte erkundet"
101
102     [commands.win]
103     usage = "Gewinnen"
104     conclusion = """...
105
106  Kapitel 4.:
107 Die Dateien werden ins Internet freigegeben, wo sie jedermann lesen kann, die
108 Leute sind schnell um jedes kleine Detail zu erkennen, einige Details kommen
109 schnell ans Licht:
110 - Slyvasta hat über ethisch fragwürdige Wissenschaftsgebiete geforscht, genau-
111   genommen, wurden hunderte Tests täglich an verschiedenen Tiermensch-
112   testobjekten durchgeführt. Aber niemand konnte beweisen, dass jemand gegen
113   ihren eigenen Willen dort war.
114 - Jedoch wurde die Forschung auch sofort von ausländischen Mächten aufgenommen,
115   die schnell entdeckt haben, dass die Tiermenschgesellschaft in der Stadt eine
116   viel größere Bedrohung ist, als sie zuerst erwartet haben.
117 - Die Staft, mit dem öffentlichem Rat von Sylvasta, hat sofort die Evakuierung
118   von allen Bürgern angeordnetm, zu jedem Ort an dem sie einen potenziell töd-
119   lichen Konflikt fürchteten. Die Stadt ging in den nächsten Tagen relativ
120   schnell unters Feuer.
121 """
122     stats = "\u001B[47m\u001B[30mDeine Spielstatistiken\u001B[0m"
123     total_ticks = "Insgesamt Spielticks: "
```

```toml
124        sidequests_complete = "Sidequests abgeschlossen: "
125        press_enter_key = "Drücke Enter, um das Spiel zu schließen."
126
127    [entities]
128    boat_key = "Ein Schlüssel für ein Schnellboot"
129
130        [entities.bed]
131        description = "Bett"
132        use = "Du schläfst im Bett."
133
134        [entities.boat]
135        description = "Schnellboot an der Küste"
136        locked = "Das Boot ist abgeschlossen."
137        locked_for_sale = "Das Boot ist abgeschlossen.\nEs klebt eine Notiz darüber, dass du den\nLadenbesitzer kontaktieren
kannst, um es zu kaufen."
138        denied = "Du darfst nichts tragen, sonst darfst du nicht aufs Boot.\nDu kannst aber Dinge ins Boot legen."
139        travel = "Du springst ins Boot und reist zu der anderen Seite..."
140        too_heavy = "Im Boot ist bereits viel zu viel Zeug drin!"
141
142            [entities.boat.give]
143            1 = "" # empty
144            2 = "ins Boot legen"
145
146        [entities.cat]
147        description = "Eine verfallene schwarze Katze"
148        pet = "Du streichelst die Katze."
149        use = "Du kannst nicht die Katze.\nBitte nicht die Katze. \u1F633\u1F633\u1F633"
150        enter = "Eine Katze läuft her."
151        leave = "Du siehst, wie eine Katze wegläuft."
152
153        [entities.comms]
154        description = "Kommunikationsgerät"
155        off = "Das Gerät ist ausgeschaltet."
156
157        [entities.couch]
158        description = "Eine braune Ledercouch"
159        sitting = "Du sitzt bereits in der Couch."
160        sit = "Du setzt dich auf die Couch."
161
162        [entities.laptop]
163        description = "Laptop"
```

```toml
164
165        [entities.laptop.boot]
166        dialog = "Du schaltest den Laptop an..."
167        option_1 = "[wait]"
168
169        [entities.laptop.home]
170        dialog = "Wähle eine Option:"
171        option_q = "Herunterfahren"
172        option_1 = "/Meine Bilder"
173        option_2 = "/Lustige Katzenvideos"
174        option_3 = "/Marie's Dokumentenscanner"
175
176        [entities.laptop.pictures]
177        dialog = """Es ist ein Bild in deinem Bilderordner.
178
179 \u1F99D
180
181
182
183
184
185
186
187 """
188        option_q = "Zurück."
189
190        [entities.laptop.cat_videos]
191        dialog = "Du schaust dir lustige Katzenvideos an..."
192        option_q = "Nett hier."
193
194        [entities.laptop.document]
195        dialog = """\u001B[35mMarie\u001B[0m's Dokumentenscanner"""
196        option_q = "Beenden"
197        option_1 = "Dokumente senden"
198
199 [home]
200 first_load = '''
201 Du wirst gleich in die Welt platziert.
202 Falls du an einem Punkt ratlos feststeckst,
203 kannst du `help` nutzen um all die verfügbaren
204 Befehle zu sehen.
```

```
205
206    ---
207
208    Du wachst auf und hörst, wie Leute draußen miteinander quatschen..
209    Du solltest gestern wirklich das Fenster geschlossen haben..
210
211    Neugierig, schaust du aus dem Fenster um zu gucken, was los ist.. Es ist eine
212    Gruppe an Demonstranten außerhalb des Medizinzentrums weiter runter auf der
213    Straße, du kannst nicht wirklich erkennen, was sie sagen oder was auf den
214    Schildern steht.
215
216    Aber es wäre nicht überraschend, falls etwas komisches da drin passieren würde,
217    aber du kannst nicht wirklich sagen, was. Vielleicht ist ja etwas in den
218    Nachrichten..
219    '''
220
221    enter = '''
222    Du gehst in deine Wohnung.
223    '''
224
225        [home.tv]
226        description = "LG 55NANO966PA 55\" Super UHD 8K HDR Smart LED TV"
227        off = "Schalte den Fernseher aus."
228        keep_watching = "Schau weiter..."
229
230        # Red: \u001B[31m
231        # Green: \u001B[32m
232        # Yellow: \u001B[33m
233        # Cyan: \u001B[36m
234        [home.tv.first_on]
235        dialog = "Du schaltest den Fernseher ein.\n\nDer Nachrichtensender kommt..."
236        dialog_a="""\u001B[31mNachrichtensprecher\u001B[0m: Bürgerunruhen nehmen zu, Sylvasta steht von allen
237    Seiten in der Kritik, und viele sind über ihre Zukunft unsicher, während die
238    Spannung zwischen den Menschlichen und Tiermenschlichen Gesellschaften
239    eskalierende Konflikte an den Grenzen der Stadt verursacht.
240
241    \u001B[31mNachrichtensprecher\u001B[0m: Wir bringen ihnen jetzt Szenen aus der Umgebung des Medizinzentrums,
242    in denen eine Gruppe Demonstranten in Opposition gegenüber der Forschung bei
243    Sylvasta erschienen ist.
244
245    \u001B[36mKorrespondent\u001B[0m: Ich bin hier, stehe dreußen mit einer Gruppe Demonstranten..
```

```
246
247   \u001B[36mKorrespondent\u001B[0m zum \u001B[33mDemonstranten\u001B[0m: Was bringt sie heute hierher?
248   """
249
250           dialog_b="""\u001B[33mDemonstrant\u001B[0m:
251   Demonstrant: Sie nehmen das Geld der Stadt und nutzen es für ihren eigenen
252   Vorteil, sie sollten keine Fördergelder bekommen, geschweige denn hier ihre
253   Forschung betreiben dürfen.
254
255   \u001B[31mNachrichtensprecher\u001B[0m: Mutige Behauptungen direkt von außerhalb Sylvasta, aber ob
256   diese begründet sind ist noch nicht bekannt. Wir haben Monate und
257   Monate an Leaks von früheren Mitarbeitern und internen Fehlern, aber
258   die echte Absicht hinter den Menschen bei Sylvasta bleibt noch abzuwarten.
259
260   \u001B[31mNachrichtensprecher\u001B[0m: Sylvasta hat persönlich angekündigt, dass sie sich weigern,
261   öffentlich jegliche Informationen über ihre Forschung preiszugeben,
262   begründet mit der öffentlichen Sicherheit. Was genau das bedeutet,
263   weiß niemand außer ihre eigenen Leute.
264   """
265
266   dialog_c="""\u001B[31mNachrichtensprecher\u001B[0m: So stellt sich auch die Frage, ob die Demonstrationen
267   unbegründet sind und lediglich Probleme in der Stadt verursachen wollen.
268   Vorher haben wir auch mit lokalen Einwohnern der Stadtmitte Earlier
269   gesprochen..
270
271   \u001B[32mLadenbesitzer\u001B[0m: Ich glaube, diese Leute wollen nur Probleme, Sylvasta war
272   unglaublich wichtig darin, die Stadt aufzubauen und uns in Frieden leben zu
273   lassen, ohne dass wir uns darüber Sorgen machen müssen, attackiert zu
274   werden. Ich glaube es gibt nicht genug Gründe, zu demonstrieren.
275
276   \u001B[32mLadenbesitzer\u001B[0m: Wie sehen bereits, wie die Welt sich gegen unsere Stadt
277   wendet, und solche internen Konflifte werden ihnen einfach einen Grund geben,
278   die Stadt zu übernehmen.
279
280   \u001B[31mNachrichtensprecher\u001B[0m: Damit ist unsere Sendung für heute Morgen beendet,
281   und um eins werden wir wieder zurück sein mit der Nachrichtenstunde.
282   """
283
284   [apartments]
285   enter = '''
286   Du gehst in die Rezeption des Wohnungsbaus.
```

```
287    '''
288
289        # Cyan: \u001B[36m
290        [apartments.receptionist]
291        description = "Der Rezeptionist sitzt hinter einem Tisch"
292
293            [apartments.receptionist.first_encounter]
294            dialog = "\u001B[36mRezeptionist\u001B[0m: Guten morgen, wie geht es Ihnen heute?"
295            option_1 = "Was ist da draußen los?"
296            option_q = "Egal."
297
298            [apartments.receptionist.protestors]
299            dialog = """\u001B[36mRezeptionist\u001B[0m: Ich weiß nicht viel, aber anscheinend schreien
300    Leute außerhalb des Medizinzentrums rum..."""
301            option_1 = "Wissen Sie noch mehr?"
302            option_q = "Achso."
303
304            [apartments.receptionist.protestors2]
305            dialog = """\u001B[36mRezeptionist\u001B[0m: Sie haben vorhin Flyer ausgeteilt, vielleicht würde
306    jemand in der Nähe es wissen..."""
307            option_q = "Danke."
308
309            [apartments.receptionist.repeated]
310            dialog = "\u001B[36mRezeptionist\u001B[0m: Hallo, wie kann ich behilflich sein?"
311            option_1 = "Was ist draußen mit den Demonstranten los?"
312            option_q = "Das war's, danke."
313
314    [city_centre]
315    first_load = '''
316    Du kommst in die Stadtmitte, es ist morgens relativ voll. Viele Menschen
317    sind hier, und es ist unwahrscheinlich dass du die meisten für ein Gespräch
318    anhalten kannst.
319
320    Es ist generell sehr unruhig im Gebiet, einige Menschen sehen angespannt aus,
321    und andere, als würde die Welt heute enden...
322    '''
323
324    enter = '''
325    Du kommst in die Stadtmitte.
326    '''
327
```

```toml
328        # Yellow: \u001B[33m
329        [city_centre.npc]
330        description = "Eine Person auf einer Bank"
331
332            [city_centre.npc.small_talk]
333            dialog = "\u001B[33mFremder\u001B[0m: Hallo, kann ich ihnen helfen?"
334            option_1 = "Haben Sie zufällig Demonstranten hier gesehen?"
335            option_2 = "Was lesen Sie?"
336            option_q = "Egal."
337
338            [city_centre.npc.protestors]
339            dialog = """\u001B[33mFremder\u001B[0m: Oh ja, sie waren sehr laut..
340    Eigentlich ganz nervig, mein Morgen wurde gestört.
341    ...
342    Sie haben mir aber einen Flyer dagelassen."""
343            option_1 = "Kann ich ihn sehen?"
344
345            [city_centre.npc.enquire]
346            dialog = "\u001B[33mFremder\u001B[0m: Die Demonstranten waren hier und haben mir ein Flyer dagelassen."
347
348            [city_centre.npc.leaflet]
349            # dialog = "\u001B[33mStranger\u001B[0m: Sure, here you go, keep it.\n[\u001B[33mStranger\u001B[0m gave you an
item.]"
350            dialog = """\u001B[33mStranger\u001B[0m: Klar, hier hab ich es...
351
352    Du schaust auf den Flyer:
353
354    \u001B[41mUNETHISCHE FORSCHUNG
355    \u001B[40m\u001B[31mUnsere Stadt ist in Gefahr, Slyvasta finanziert
356    weiter ihre Forschung, aber machen nichts dafür,
357    dass die Stadt am Leben bleibt.
358
359    Sylvasta nutzt ihre Stellung aus, um
360    weiter an komplett ILLEGALER Biotech-
361    nischer Herstellung von Tiermenschen
362    zu forschen.\u001B[0m
363
364    Unterstützen Sie uns: \u001B[36mhttps://sylvasta.vercel.app\u001B[0m
365    """
366            option_1 = "Dankeschön."
367
```

```
368    [street]
369    first_load = '''
370    Du kommst in die Hauptstraße, die die hauptsächlichen Teile der Stadt vereint.
371
372    Es gibt viel Chaos und Lärm, zu deinem Westen siehst du ein paar
373    Demonstranten vor dem Medizinzentrum, sie halten Schilder hoch,
374    "UNETHISCHE FORSCHUNG", "UNSERE BEZIEHUNGEN AUF DEM SPIEL",
375    "UNVORHERGESEHENE KONSEQUENZEN".
376    '''
377
378    enter = '''
379    Du kommst in die Hauptstraße.
380    '''
381
382        # Red: \u001B[31m
383        [street.protestors]
384        description = "Eine Gruppe Demonstranten hält Schilder hoch"
385        blocking = "Eine Gruppe an Demonstranten blockiert den Eingang."
386
387            [street.protestors.small_talk]
388            dialog = "\u001B[31mDemonstrant\u001B[0m: Hast du darüber Gehört, was Sylvasta macht und warum wir hier sind?"
389            option_1 = "Nein, aber erzähl doch mal darüber."
390            option_2 = "Egal."
391
392            [street.protestors.enquire]
393            dialog = """\u001B[31mDemonstrant\u001B[0m: Sie nutzen öffentliches Geld um ihre illegale Forschungen
394    weiterzuführen, sie haben doch die Leaks gesehen, oder?"""
395            option_1 = "Nein."
396            option_2 = "Alles klar.. Ich lass Sie dann mal in Ruhe."
397
398            [street.protestors.leaks]
399            dialog = """\u001B[31mDemonstrant\u001B[0m: Naja, einige frühere Mitarbeiter haben Infos über ihre aktuellen
400    Forschungsprojekte geleakt und anscheinend experimentieren sie mit Biotechnik
401    an Tiermenschen, das sollte nicht toleriert oder gar subventioniert werden."""
402            option_1 = "Ok."
403
404            [street.protestors.confrontation]
405            dialog = "\u001B[31mProtestor\u001B[0m: You taking me for some sort of conspiracy theorist?"
406            option_1 = "Ja."
407            option_2 = "Ich gehe jetzt einfach."
408
```

```toml
409   [shop]
410   enter = '''
411   Du kommst in den Laden, die Glocke klingelt als du die Tür hinter dir schließt.
412   '''
413
414       # Green: \u001B[32m
415       [shop.npc]
416       description = "Der Ladenbesitzer"
417       leave = "Egal."
418
419       currently_have_amount_of_money = "Du hast aktuell:"
420       not_enough = "Du hast nicht genug Geld und leistest dir kein"
421       too_heavy = "Zu schwer für dich!"
422
423       out_of_stock = "ausverkauft"
424       x_left = "übrig"
425
426           [shop.npc.fake_item]
427           cat = "Eine einzige Katze"
428
429           [shop.npc.item_out_of_stock]
430           1 = "Dieses Item ist"
431           2 = ", du kannst es nicht kaufen"
432
433           [shop.npc.bought]
434           1 = "Du kaufst dir"
435           2 = "und machst es in deinen Beutel."
436
437           [shop.npc.greeting]
438           Exposition = "\u001B[32mLadenbesitzer\u001B[0m: Guten Tag, was würden Sie gerne kaufen?"
439           Recon = "\u001B[32mLadenbesitzer\u001B[0m: Hallo, was neues kaufen?"
440           Stealth = "\u001B[32mLadenbesitzer\u001B[0m: Wollen Sie etwas kaufen?"
441           End = "\u001B[32mLadenbesitzer\u001B[0m: Brauchen Sie etwas?"
442
443   [back_alley]
444   first_load = '''
445   Du kommst in eine dunkle Gasse, es ist hier ziemlich leise, ein starker
446   Kontrast zur Stadtmitte, obwohl sie direkt dazwischen liegt. Die Seiten der
447   Gasse sind voll mit Bauten und dunkler Beleuchtung.
448
449   Du siehst vor dir eine dir bekannte Figur.
```

```toml
450    '''
451
452    enter = '''
453    Du kommst in eine dunkle Gasse.
454    '''
455
456    [coastline]
457    enter = '''
458    Du kommst an die Stadtküste, das Wasser ist eher ruhig und still.
459    '''
460
461    [mainland_coastline]
462    enter = '''
463    Du kommst an die Festlandküste,
464    der See konfrontiert direkt die Steine, die gegen ihn ausgelegt sind,
465    es ist außerhalb des Beton-Dschungels außerdem viel leiser.
466    '''
467
468    [forest]
469    enter = '''
470    Du wanderst durch den Wald und hörst wie die Vögel an den Baumkronen pfeifen...
471    Durch die Mitte fließt ein Fluss und es ist eine alte Holzbrücke darüber.
472    Es würde relativ einfach sein, dich hier für eine Weile zu verlaufen.
473    '''
474
475        # Yellow: \u001B[33m
476        [forest.old_man]
477        description = "Ein alter Mann, der durch den Wald wandert"
478        full = "Du kannst ihm nichts mehr geben."
479        accept = "\u001B[33mAlter Mann\u001B[0m: Danke, dass du ihn mir zurückgebracht hast."
480        deny = "\u001B[33mAlter Mann\u001B[0m: Ich möchte aber nicht den"
481
482            [forest.old_man.small_talk]
483            dialog = "\u001B[33mAlter Mann\u001B[0m: Du hast meine schwarze Katze doch nirgendwo gesehen?"
484            option_1 = "In der Stadt habe ich so eine gesehen..."
485            option_q = "Nein."
486
487            [forest.old_man.request]
488            dialog = "\u001B[33mAlter Mann\u001B[0m: Könntest du sie mir zurückbringen?"
489            option_1 = "Klar doch."
490            option_q = "Gerade geht das leider nicht."
```

```toml
491
492        [forest.old_man.praise]
493        dialog = "\u001B[33mAlter Mann\u001B[0m: Danke für deine Hilfe!"
494        option_q = "[verlassen]"
495
496 [worm_hole]
497 enter = '''
498 Du gehst ins Wurmloch...
499 '''
500
501 [medical_centre]
502 enter = '''
503 Du bist jetzt an der Rezeption des Medizinzentrums.
504 '''
505
506     # Red: \u001B[31m
507     [medical_centre.guard]
508     description = "Sicherheitsmann an der Tür"
509     blocking = "Die Security schaut auf die Treppen, du kannst nicht an ihnen vorbei."
510
511        [medical_centre.guard.small_talk]
512        dialog = "\u001B[31mWachmann\u001B[0m: Was willst du?"
513        option_q = "Egal."
514
515 [medical_centre_office]
516 enter = '''
517 Du bist im Büro des Medizinzentrums.
518 Du solltest hier eigentlich auf jeden Fall nicht sein...
519 '''
520
521     [medical_centre_office.books]
522        [medical_centre_office.books.1]
523        title = "Alan Stern. Auf der Suche nach New Horizons"
524        contents = """
525 Am 14. Juli 2015 geschah etwas Erstaunliches. Mehr als 3 Milliarden Meilen von
526 der Erde, eine kleine NASA-Raumsonde namens New Horizons, flog an Pluto
527 mit 32.000 Meilen pro Stunde vorbei und konzentrierte seine Instrumente auf
528 die lang mysteriösen Eiswelten des Pluto-Systems und ging dann genauso schnell
529 weiter hinaus ins Jenseits.
530 """
531
```

```
532        [medical_centre_office.books.2] # yes
533        title = "Stapel von Forschungspapieren"
534        contents = """
535 Bei unseren jüngsten Testreihen haben wir festgestellt, dass die Wirkung von
536 \u001B[47m\u001B[37m[redacted]\u001B[0m erhebliche negative Auswirkungen hat.
537 Diese Tests wurden über eine Reihe von 8 Tagen durchgeführt und verursachten
538 \u001B[47m\u001B[37m[redacted]\u001B[0m und \u001B[47m\u001B[37m[redacted]\u001B[0m im Körper.
539
540 Dieses Dokument ist vertraulich und darf nicht an Dritte weitergegeben werden.
541 """
542
543        [medical_centre_office.books.3]
544        title = "Barry A. Burd. Java For Dummies"
545        contents = """
546 Java ist die Plattformunabhängige, objektorientierte Programmiersprache, die für
547 die Entwicklung von Web und Mobile-Anwendungen genutzt wird. Die neue Version
548 hat erweiterte Funkionalität, auf die Programmierer ganz aufgeregt warteten, und
549 dieses Handbuch wird alle davon vorstellen.
550
551 Sie werden lernen, wie man gut in Java programmiert, etwa so:
552
553 \u001B[36mimport \u001B[33mjava\u001B[37m.\u001B[33mio\u001B[37m.\u001B[33mBufferedReader\u001B[37m;
554 \u001B[36mimport \u001B[33mjava\u001B[37m.\u001B[33mio\u001B[37m.\u001B[33mInputStreamReader\u001B[37m;
555 \u001B[33mpublic class \u001B[31mMain  \u001B[34m{
556    \u001B[33mpublic static  void \u001B[31mmain  \u001B[34m(\u001B[33mString[] args\u001B[34m) \u001B[34m{
557        \u001B[33mtry{ String a=new \u001B[31mBufferedReader \u001B[34m(
558            \u001B[33mnew \u001B[31mInputStreamReader\u001B[34m( System\u001B[37m.\u001B[33mout\u001B[37m \u001B[34m))
\u001B[37m.
559                \u001B[31mreadLine() \u001B[37m;
560        \u001B[33mif
561            \u001B[34m(a == \u001B[32m"hi"\u001B[34m) {
562            System\u001B[37m.\u001B[33mout\u001B[37m.\u001B[31mprintln\u001B[34m(\u001B[32m"Hello!!"\u001B[34m)
\u001B[37m;
563        \u001B[34m} \u001B[33melse
564        \u001B[34m{
565            System\u001B[37m.\u001B[33mout\u001B[37m.\u001B[31mprintln\u001B[34m(\u001B[32m"you did not greet me!"\u001B[
34m) \u001B[37m;
566            \u001B[34m}
567    } \u001B[33mcatch \u001B[34m(\u001B[33mException e\u001B[34m) {\u001B[35m/*something bad happen! */ \u001B[34m} } }
568 \u001B[0m"""
569
```

```toml
        [medical_centre_office.books.4] # yes
        title = "Dr. Mike Ox. Hochstapler in die Gesellschaft integrieren"
        contents = """
[Markiert als nur für den internen Gebrauch.]

Dieses Buch beschreibt detailliert Methoden, welche genutzt werden können, um das
allgemeine Volk zu manipulieren. Ich erkläre ganz genau, wie man ge-
fährliche Personen ganz subtil entfernt und Geheimnisse geheim haltet.
"""

        [medical_centre_office.books.5] # yes
        title = "RNA-Sequenzen"
        contents = """
Vertrauliches Dokument

Figur 1.
GAGAATAAACTAGTATTCTTCTGGTCCCCACAGACTCAGAGAGAACCCGCCACCATGTTCGTGTTCCTGGTGCTGCTGCC
TCTGGTGTCCAGCCAGTGTGTGAACCTGACCACCAGAACACAGCTGCCTCCAGCCTACACCAACAGCTTTACCAGAGGCG
TGTACTACCCCGACAAGGTGTTCAGATCCAGCGTGCTGCACTCTACCCAGGACCTGTTCCTGCCTTTCTTCAGCAACGTG
ACCTGGTTCCACGCCATCCACGTGTCCGGCACCAATGGCACCAAGAGATTCGACAACCCCGTGCTGCCCTTCAACGACGG
GGTGTACTTTGCCAGCACCGAGAAGTCCAACATCATCAGAGGCTGGATCTTCGGCACCACACTGGACAGCAAGACCCAGA
GCCTGCTGATCGTGAACAACGCCACCAACGTGGTCATCAAAGTGTGCGAGTTCCAGTTCTGCAACGACCCCTTCCTGGGC
GTCTACTACCACAAGAACAACAAGAGCTGGATGGAAAGCGAGTTCCGGGTGTACAGCAGCGCCAACAACTGCACCTTCGA
GTACGTGTCCCAGCCTTTCCTGATGGACCTGGAAGGCAAGCAGGGCAACTTCAAGAACCTGCGCGAGTTCGTGTTTAAGA
ACATCGACGGCTACTTCAAGATCTACAGCAAGCACACCCCTATCAACCTCGTGCGGGATCTGCCTCAGGGCTTCTCTGCT
CTGGAACCCCTGGTGGATCTGCCCATCGGCATCAACATCACCCGGTTTCAGACACTGCTGGCCCTGCACAGAAGCTACCT
GACACCTGGCGATAGCAGCAGCGGATGGACAGCTGGTGCCGCCGCTTACTATGTGGGCTACCTGCAGCCTAGAACCTTCC
TGCTGAAGTACAACGAGAACGGCACCATCACCGACGCCGTGGATTGTGCTCTGGATCCTCTGAGCGAGACAAAGTGCACC
"""

        [medical_centre_office.books.6]
        title = "Jules Verne. Um die Welt in 80 Tagen"
        contents = """
An einem unglückseligen Abend im Reform Club wettet Phileas Fogg unvorsichtiger-
weise mit seinen um 20.000 Pfund, dass er in nur achtzig Tagen um den gesamten
Globus reisen kann. um die ganze Welt reisen kann - und er ist fest entschlossen,
nicht zu verlieren. Der zurückhaltende Engländer durchbricht die gewohnte Routine
sofort und macht sich auf den Weg nach Dover, in Begleitung seines heißblütigen
französischen Dieners Passepartout. Unterwegs mit Zug, Dampfschiff, Segelboot,
Schlitten und sogar einem Elefanten müssen sie Stürme, Entführungen, Natur-
katastrophen, Sioux-Angriffe und den verbissenen Inspektor Fix von Scotland Yard
```

```
611    - der glaubt, dass Fogg die Bank von England ausgeraubt hat - überwinden, um die
612    außergewöhnliche Wette zu gewinnen.
613    """
614
615    [marie]
616    description = "\u001B[35mMarie der Nerz\u001B[0m"
617
618        # Purple: \u001B[35m
619        [marie.alley]
620
621            [marie.alley.small_talk]
622            dialog = "\u001B[35mMarie\u001B[0m: Was bringt dich hierher?"
623            option_1 = "Hast du von den Protesten gehört?"
624            option_2 = "Sylvasta hat etwas böses vor..."
625            option_q = "Egal, hab einen schönen Tag."
626
627            [marie.alley.protests]
628            dialog = """\u001B[35mMarie\u001B[0m: Ja, die von Sylvasta haben auf jeden Fall
629    nichts Gutes auf dem Schirm, aber ich weiß auch nicht ob ich den Demonstranten
630    trauen sollte."""
631            option_1 = "Willst du mir dabei helfen, das herauszufinden?"
632            option_q = "Egal."
633
634            [marie.alley.sylvasta]
635            dialog = "\u001B[35mMarie\u001B[0m: Glaubst du, ich weiß das nicht? Willst du mir helfen, es herauszufinden?"
636            option_1 = "Klar."
637            option_q = "Nein danke."
638
639            [marie.alley.confirm]
640            dialog = "\u001B[35mMarie\u001B[0m: Wir könnten einbrechen und die Wahrheit herausfinden."
641            option_1 = "Lass machen! [bringt die Story weiter]"
642            option_q = "Auf keinen Fall!"
643
644            [marie.alley.recon]
645            dialog = """\u001B[35mMarie\u001B[0m: Okay, während ich nachdenke, wie wir reinkommen, musst du mir
646    eetwas bringen, worüber wir reden müssen, ich bin mir sicher, dass es im Laden etwas gibt, komm wieder hierher
647    wenn du zurückkommst."""
648            option_q = "Verstanden."
649
650            [marie.alley.waiting]
651            dialog = "\u001B[35mMarie\u001B[0m: Na dann? Hol es dir."
```

```
652            option_q = "Okay."
653
654        [marie.alley.mission_brief]
655            dialog = """\u001B[35mMarie\u001B[0m: Also, der Plan ist, du gehst zum Medizinzentrum,
656    bummelst lässig zur Rezeption, du darfst aber nicht auffallen... Finde dir zum Beispiel einen
657    Stuhl um darauf zu sitzen oder so was. Jedenfalls, wenn du dort bist und denkst dass die
658    Security nicht zuschaut, nutz dein Kommunikationsgerät um mir bescheid zu sagen.
659
660    \u001B[35mMarie\u001B[0m: Alsdann werde ich eine Ablenkung senden und sie in ihr
661    Pausenzimmer führen während du ins Büro schleichst und versuchst, alles was du willst, zu finden...
662    Aber wenn du dort bist, sag mir bescheid.
663
664    \u001B[35mMarie\u001B[0m: Wenn du willst, kannst du das Kommunikationsgerät jetzt testen."""
665            option_q = "Okay."
666
667        [marie.comms]
668        received = """\u001B[35mMarie\u001B[0m: Alles klar, ich hab die Dokumente gefunden,
669    Ich sende sie an die breiten Massen. Gut gemacht mit dem Einbruch."""
670        bad_documents = "\u001B[35mMarie\u001B[0m: Das sieht nicht wie die richtigen Dokumente aus..."
671        no_access = "\u001B[31mZugriff verweigert!\u001B[0m"
672
673            [marie.comms.orientation]
674            dialog = "\u001B[35mMarie\u001B[0m: Was willst du?"
675            option_1 = "Wohin muss ich nochmal?"
676            option_2 = "Kannst du mich hören?"
677            option_q = "Egal."
678
679            [marie.comms.directions]
680            dialog = """\u001B[35mMarie\u001B[0m: Du musst ins Medizinzentrum, wenn du dort bist, versuch
681    nicht aufzufallen und sage mir, wenn du in Position bist."""
682            option_q = "Verstanden."
683
684            [marie.comms.hear]
685            dialog = "\u001B[35mMarie\u001B[0m: Laut und deutlich."
686            option_q = "Nett."
687
688            [marie.comms.complaint]
689            dialog = "\u001B[35mMarie\u001B[0m: Versuche, nicht aufzufallen..."
690            option_q = "Okay."
691
692            [marie.comms.in_position]
```

```
693          dialog = "\u001B[35mMarie\u001B[0m: Okay, Ich werde eine Ablenkung schaffen, gib mir einen Moment..."
694          option_1 = "[warten]"
695
696          [marie.comms.distraction]
697          dialog = """\u001B[31mWachmann\u001B[0m: Wer ist da?
698  Komm raus, oder wir haben ein Problem.
699
700  (Du siehst, wie der Wache in den Pausenraum geht)
701  """
702          option_1 = "[warten]"
703
704          [marie.comms.coast_is_clear]
705          dialog = "\u001B[35mMarie\u001B[0m: Okay, jetzt ist deine Chance."
706          option_q = "Verstanden."
707
708          [marie.comms.gogogo]
709          dialog = "\u001B[35mMarie\u001B[0m: Auf was wartest du??"
710          option_q = "AAAAAAA"
711
712          [marie.comms.office]
713          dialog = "\u001B[35mMarie\u001B[0m: Was siehst du?"
714          option_1 = "Rumgestreute Dokumente."
715
716          [marie.comms.documents]
717          dialog = """\u001B[35mMarie\u001B[0m: Ok, schau sie dir durch und finde alles, was relevant
718  ist, achte besonders auf alles, was die Sachen erwähnt, über die demonstriert wird.
719  Wenn du das gemachst hast, solltest du die Dokumente mir hochladen,
720  geh zurück zu deiner Wohnung und wir machen es von dort, sag mir wenn du dort bist.
721
722  \u001B[35mMarie\u001B[0m: Ich schau genau darauf, dass deine Deckung nicht platzt.
723
724  \u001B[33mTipp: Du kannst die Dokumente benutzen (use), um sie zu lesen.\u001B[0m"""
725          option_q = "Okay."
726
727          [marie.comms.home]
728          dialog = """\u001B[35mMarie\u001B[0m: Warscheinlich hast du die Dokumente, schalte deinen PC an,
729  geh auf den Link den ich dir gesendet hab, er sollte dort erscheinen, und lade dann hoch."""
730
731  [stage]
732  reached_conclusion = """\n
733  Du hast das Spielende erreicht, du bist jetzt in einem Open-World-Modus,
```

```
734    erkunde weiter um deine Ergebnisse zu maximieren.
735
736    Du kannst endgültig aufhören mit dem Befehl \u001B[47m\u001B[30mwin\u001B[0m
737    """
```

```toml
1   # Translations for World of Deez
2
3   [global]
4   sight = "You can see:"
5
6       [global.can_go_in_x_directions]
7       1 = "You may go in"
8       2 = "directions"
9
10  [selectors]
11  direction = "<direction>"
12  something = "<something>"
13  someone = "<someone>"
14  item = "<item>"
15
16      [selectors.cant_find]
17      1 = "You look around for"
18      2 = "but can't find anything"
19
20  [commands]
21  unknown = "Not sure what you're trying to do."
22
23  back = "go back to the previous room"
24  quit = "quit the game"
25  where_am_i = "describe the current room again"
26
27      [commands.bag]
28      usage = "look inside your bag or at something's inventory"
29      cant_find = "Can't find what you want to look at."
30      empty = "Your bag is empty!"
31      entity_empty = "doesn't appear to have anything"
32      can_carry_kg = "You can carry"
33      are_carrying_kg = "You are carrying"
34      look_in_bag = "You look in your bag to see"
35      entity_appears_to_have = "appears to have"
36
37      [commands.drop]
38      usage = "drop an item from your bag"
39      nothing_specified = "What do you want to drop?"
40
41          [commands.drop.dropped]
```

```toml
42          1 = "You drop"
43          2 = "out of your bag"
44
45      [commands.give]
46      usage = "give something to someone"
47      nothing_specified = "What do you want to give?"
48      no_target = "What / who are you putting this in?"
49      denied_player = "You cannot give yourself to anyone or anything. \u1F438"
50
51          [commands.give.denied]
52          1 = "Cannot give"
53          2 = "to"
54
55      [commands.go]
56      usage = "go in a certain direction"
57      nothing_specified = "Where are you going?"
58
59      [commands.help]
60      usage = "show help menu"
61      can_run = "You can run the following commands:"
62
63      [commands.pet]
64      usage = "pet something around you or in your inventory"
65      nothing_specified = "What are you trying to pet?"
66      denied = "You cannot pet"
67
68      [commands.take]
69      usage = "put something in your bag"
70      nothing_specified = "From who?"
71      entity_does_not_have_entity = "does not have"
72      item_not_specified = "What do you want to take?"
73
74          [commands.take.took]
75          1 = "You take"
76          2 = "from"
77          3 = "and put it in your bag"
78
79          [commands.take.denied]
80          1 = "You cannot take"
81          2 = "it's too heavy to put in your bag"
82
```

```toml
 83        [commands.talk]
 84        usage = "start talking with someone"
 85        nothing_specified = "What do you want to talk with?"
 86        denied = "You cannot talk with"
 87
 88        [commands.use]
 89        usage = "use something around you or in your inventory"
 90        nothing_specified = "What do you want to use?"
 91        denied = "You cannot use"
 92
 93        [commands.map]
 94        usage = "show the world map"
 95        close = "Press enter to close."
 96
 97            [commands.map.discovered]
 98            1 = "You have discovered"
 99            2 = "of the world"
100
101        [commands.win]
102        usage = "win the game"
103        conclusion = """...
104
105 Chapter 4.:
106 The files are released into the internet for anyone to read, people are quick to
107 analyse through every single tiny detail, some immediate details come to light:
108 - Sylvasta was researching ethically questionable areas of science, in
109   particular, they were running hundreds of tests daily on a variety of beastman
110   test subjects. But nobody could prove anyone was there against their will.
111 - However, the research did also get immediately picked up by foreign powers who
112   quickly discovered that the beastman society living in the city is a far
113   greater threat than they initially anticipated.
114 - The city, with Sylvasta's public advice, immediately ordered evacuation of all
115   citizens to any area they could find fearing a potentially deadly conflict on
116   the horizon. The city quickly came under fire over the coming days.
117 """
118        stats = "\u001B[47m\u001B[30mYour final game stats\u001B[0m"
119        total_ticks = "Total game ticks: "
120        sidequests_complete = "Side-quests completed: "
121        press_enter_key = "Press enter to close the game."
122
123 [entities]
```

```toml
124    boat_key = "A key to the speed boat"
125
126        [entities.bed]
127        description = "Bed"
128        use = "You take a nap."
129
130        [entities.boat]
131        description = "Speedboat docked on the coast"
132        locked = "The boat is locked."
133        locked_for_sale = "The boat is locked.\nThere is a note which says to contact the shopkeeper to buy this boat."
134        denied = "You must not be carrying anything to use the boat.\nYou can however put things in the boat."
135        travel = "You hop in the boat and travel to the other side..."
136        too_heavy = "The boat is carrying too much stuff already!"
137
138            [entities.boat.give]
139            1 = "Put"
140            2 = "in the boat"
141
142        [entities.cat]
143        description = "A stray black cat"
144        pet = "You pet the cat."
145        use = "You cannot the cat.\nPlease do not the cat. \u1F633\u1F633\u1F633"
146        enter = "A cat has wandered in among us."
147        leave = "You see a cat leave."
148
149        [entities.comms]
150        description = "Communicator device"
151        off = "The device is off."
152
153        [entities.couch]
154        description = "A brown leather couch"
155        sitting = "You are already sitting in the couch."
156        sit = "You sit down on the couch."
157
158        [entities.laptop]
159        description = "Laptop"
160
161            [entities.laptop.boot]
162            dialog = "You turn the computer on..."
163            option_1 = "[wait]"
164
```

```toml
165        [entities.laptop.home]
166        dialog = "Select an option:"
167        option_q = "Power off"
168        option_1 = "/My Pictures"
169        option_2 = "/Funny cat videos"
170        option_3 = "/Marie's document scanner"
171
172        [entities.laptop.pictures]
173        dialog = """There is only one picture in your pictures folder:
174
175 \u1F99D
176
177
178
179
180
181
182
183 """
184        option_q = "Go back."
185
186        [entities.laptop.cat_videos]
187        dialog = "You look at funny cat videos..."
188        option_q = "Neat."
189
190        [entities.laptop.document]
191        dialog = """\u001B[35mMarie\u001B[0m's document scanner"""
192        option_q = "Quit"
193        option_1 = "Send documents"
194
195 [home]
196 first_load = '''
197 You're about to be placed into the world.
198 If at any point you are stuck with what to do,
199 you can use `help` to view all available commands.
200
201 ---
202
203 You wake up to the sound of people chanting outside..
204 You really should've closed the window last night..
205
```

```
206   Curious, you peer out the window to see what's going on.. there's a group of
207   protestors outside the Medical Centre down the street, you can't really make out
208   what they're saying or what their signs say.
209
210   Though it'd not be surprising if something strange is going on in there, but you
211   can't really put your finger on it. Maybe there's something on the news..
212   '''
213
214   enter = '''
215   You enter your apartment.
216   '''
217
218       [home.tv]
219       description = "LG 55NANO966PA 55\" Super UHD 8K HDR Smart LED TV"
220       off = "Turn the TV off."
221       keep_watching = "Keep watching..."
222
223           # Red: \u001B[31m
224           # Green: \u001B[32m
225           # Yellow: \u001B[33m
226           # Cyan: \u001B[36m
227           [home.tv.first_on]
228           dialog = "You turn the TV on.\n\nThe news channel comes up..."
229           dialog_a="""\u001B[31mNews Anchor\u001B[0m: Civil unrest is rising, Sylvasta is facing criticism from all
230   sides, and many people are uneasy about their future as rising tension between
231   human and beastman societies is causing escalated conflict around the city
232   borders.
233
234   \u001B[31mNews Anchor\u001B[0m: We bring you now to scenes outside of the Medical Centre where a
235   group of protestors have shown up in opposition to the research being led at
236   Sylvasta.
237
238   \u001B[36mCorrespondent\u001B[0m: I am here, standing outside with the group of protestors..
239
240   \u001B[36mCorrespondent\u001B[0m to \u001B[33mProtestor\u001B[0m: What brings you here today?
241   """
242
243           dialog_b="""\u001B[33mProtestor\u001B[0m:
244   Protestor: They're taking the city's money and using it for their own gain, they
245   shouldn't receive any funding let alone be allowed to operate here.
246
```

```
247    \u001B[31mNews Anchor\u001B[0m: Bold claims coming straight from outside Sylvasta, whether these
248    claims are grounded in anything is yet to be discovered, we've seen months and
249    months of leaks come out from former employees and internal mishaps but are yet
250    to truly find out the intention behind the people at Sylvasta.
251
252    \u001B[31mNews Anchor\u001B[0m: Sylvasta has personally announced that they refuse to communicate
253    or elaborate any further on their internal research citing public safety,
254    whatever that means nobody outside of their internal staff knows.
255    """
256
257    dialog_c="""\u001B[31mNews Anchor\u001B[0m: It also begs the question whether the protests are unfounded and
258    just there to stir up trouble in the city. Earlier today we also spoke to local
259    residents living in the centre of the city..
260
261    \u001B[32mShopkeeper\u001B[0m: I think these guys just want to cause trouble, Sylvasta was vital in
262    establishing this city and letting us live in peace without having to worry
263    about being attacked, I just don't think there's enough reason to protest.
264
265    \u001B[32mShopkeeper\u001B[0m: We're already seeing the world turn against the city and these sorts
266    of internal conflicts will just give them reason to step in and take control.
267
268    \u001B[31mNews Anchor\u001B[0m: That concludes our broadcast for this morning, and we'll be back
269    for the 1pm News Hour.
270    """
271
272    [apartments]
273    enter = '''
274    You enter the apartment complex reception.
275    '''
276
277        # Cyan: \u001B[36m
278        [apartments.receptionist]
279        description = "The receptionist sitting behind a desk"
280
281            [apartments.receptionist.first_encounter]
282            dialog = "\u001B[36mReceptionist\u001B[0m: Good morning, how are you doing today?"
283            option_1 = "What is that racket outside?"
284            option_q = "Nevermind."
285
286            [apartments.receptionist.protestors]
287            dialog = """\u001B[36mReceptionist\u001B[0m: I don't know much but it looks like a group of people shouting
```

```
288    outside the Medical Centre..."""
289            option_1 = "Do you know anything more?"
290            option_q = "Alright."
291
292            [apartments.receptionist.protestors2]
293            dialog = """\u001B[36mReceptionist\u001B[0m: They were handing out flyers as they came up, maybe someone
294    nearby would know..."""
295            option_q = "Thanks."
296
297            [apartments.receptionist.repeated]
298            dialog = "\u001B[36mReceptionist\u001B[0m: Hello, what can I help you with?"
299            option_1 = "What's going on with those protestors outside?"
300            option_q = "That's all, thanks."
301
302    [city_centre]
303    first_load = '''
304    You enter the city square, it's quite busy in the mornings. There's a lot of
305    people running around, unlikely you could stop most of them for a chat.
306
307    There's a general unease in the area, some people look quite tense and others
308    look like they're spending their last day on Earth...
309    '''
310
311    enter = '''
312    You enter the city square.
313    '''
314
315        # Yellow: \u001B[33m
316        [city_centre.npc]
317        description = "A person sitting at a bench"
318
319            [city_centre.npc.small_talk]
320            dialog = "\u001B[33mStranger\u001B[0m: Hello, could I help you?"
321            option_1 = "You don't happen to have seen the protestors go by here?"
322            option_2 = "What are you reading?"
323            option_q = "Nevermind."
324
325            [city_centre.npc.protestors]
326            dialog = """\u001B[33mStranger\u001B[0m: Oh yes, they were quite loud..
327    They were rather annoying, disturbed my morning.
328    ...
```

```
329    They did leave me this leaflet though"""
330            option_1 = "Can I see it?"
331
332            [city_centre.npc.enquire]
333            dialog = "\u001B[33mStranger\u001B[0m: These guys went past and left me a leaflet."
334
335            [city_centre.npc.leaflet]
336            # dialog = "\u001B[33mStranger\u001B[0m: Sure, here you go, keep it.\n[\u001B[33mStranger\u001B[0m gave you an
item.]"
337            dialog = """\u001B[33mStranger\u001B[0m: Sure, I've got it here...
338
339    You look at the leaflet:
340
341    \u001B[41mUNETHICAL RESEARCH
342    \u001B[40m\u001B[31mOur city is in danger, Sylvasta further
343    funding into their research yet do nothing
344    to help the city survive.
345
346    Sylvasta is abusing their position to
347    continue work into completely ILLEGAL
348    bioengineering of beastmen.\u001B[0m
349
350    Support us: \u001B[36mhttps://sylvasta.vercel.app\u001B[0m
351    """
352            option_1 = "Thanks."
353
354    [street]
355    first_load = '''
356    You enter the main city street connecting the major parts of the city.
357
358    There's a lot of chaos and noise here, to your west you can see protestors
359    outside of the medical centre complex, they're holding signs up, "UNETHICAL
360    RESEARCH", "OUR RELATIONS AT STAKE", "UNFORSEEN CONSEQUENCES".
361    '''
362
363    enter = '''
364    You enter the city street.
365    '''
366
367        # Red: \u001B[31m
368        [street.protestors]
```

```
369        description = "A group of protestors holding signs"
370        blocking = "There is a group of protestors blocking the way in."
371
372            [street.protestors.small_talk]
373            dialog = "\u001B[31mProtestor\u001B[0m: Have you heard about what Sylvasta is doing and why we're out here?"
374            option_1 = "No, enlighten me."
375            option_2 = "Nevermind."
376
377            [street.protestors.enquire]
378            dialog = """\u001B[31mProtestor\u001B[0m: They are using the city's money to further their frankly illegal
379    research, you've seen the leaks right?"""
380            option_1 = "No."
381            option_2 = "Right.. I'll leave you to it."
382
383            [street.protestors.leaks]
384            dialog = """\u001B[31mProtestor\u001B[0m: Well some former employees leaked information about their current
385    research projects and from what we can tell is that they're doing bioengineering
386    on beastmen, this shouldn't be tolerated let alone funded by the city."""
387            option_1 = "Ok."
388
389            [street.protestors.confrontation]
390            dialog = "\u001B[31mProtestor\u001B[0m: You taking me for some sort of conspiracy theorist?"
391            option_1 = "Yes."
392            option_2 = "I'm leaving now."
393
394    [shop]
395    enter = '''
396    You enter the shop, the bell rings as you close the door behind you.
397    '''
398
399        # Green: \u001B[32m
400        [shop.npc]
401        description = "The shop keeper"
402        leave = "Nevermind."
403
404        currently_have_amount_of_money = "You currently have:"
405        not_enough = "You don't have enough money to buy"
406        too_heavy = "Too heavy for you to carry!"
407
408        out_of_stock = "out of stock"
409        x_left = "left"
```

```
410
411            [shop.npc.fake_item]
412            cat = "A singular cat"
413
414            [shop.npc.item_out_of_stock]
415            1 = "This item is"
416            2 = "you may not buy it"
417
418            [shop.npc.bought]
419            1 = "You buy"
420            2 = "and put it in your bag"
421
422            [shop.npc.greeting]
423            Exposition = "\u001B[32mShopkeeper\u001B[0m: Good day, what would you like to buy?"
424            Recon = "\u001B[32mShopkeeper\u001B[0m: Hello, anything in mind today?"
425            Stealth = "\u001B[32mShopkeeper\u001B[0m: Looking to buy something?"
426            End = "\u001B[32mShopkeeper\u001B[0m: Need anything new?"
427
428 [back_alley]
429 first_load = '''
430 You enter a dark alley, it is quite quiet here, a stark contrast to the city
431 centre yet located nearly right in the middle. The sides of the alley are lined
432 with derelict buildings and dim lighting.
433
434 You make out a familiar figure further ahead.
435 '''
436
437 enter = '''
438 You enter the back alley.
439 '''
440
441 [coastline]
442 enter = '''
443 You arrive at the city's coastline, the water is rather calm and still.
444 '''
445
446 [mainland_coastline]
447 enter = '''
448 You arrive at the coastline on the mainland,
449 the sea is crashing against rocks lined against it,
450 it is significantly colder out here away from the concrete jungle.
```

```toml
451    '''
452
453    [forest]
454    enter = '''
455    You wander through the forest, you hear birds whistling at the peaks of trees...
456    There's a river flowing through the middle with an old wooden bridge above it.
457    It would be quite easy to lose yourself here for a few hours.
458    '''
459
460        # Yellow: \u001B[33m
461        [forest.old_man]
462        description = "An old man wandering through the forest"
463        full = "There's nothing else you can give them."
464        accept = "\u001B[33mOld Man\u001B[0m: Thank you for bringing him back to me."
465        deny = "\u001B[33mOld Man\u001B[0m: I don't want your"
466
467            [forest.old_man.small_talk]
468            dialog = "\u001B[33mOld Man\u001B[0m: You haven't seen my cat anywhere have you?"
469            option_1 = "I've seen this black stray around town..."
470            option_q = "No."
471
472            [forest.old_man.request]
473            dialog = "\u001B[33mOld Man\u001B[0m: Could you bring him back to me?"
474            option_1 = "Sure."
475            option_q = "Sorry, not right now."
476
477            [forest.old_man.praise]
478            dialog = "\u001B[33mOld Man\u001B[0m: Thank you for helping me!"
479            option_q = "[leave]"
480
481    [worm_hole]
482    enter = '''
483    You step into the worm hole...
484    '''
485
486    [medical_centre]
487    enter = '''
488    You're now at the Medical Centre's reception.
489    '''
490
491        # Red: \u001B[31m
```

```toml
492        [medical_centre.guard]
493        description = "Security guard stationed at the door"
494        blocking = "There is security watching the stairs, there's no way to get past them."
495
496            [medical_centre.guard.small_talk]
497            dialog = "\u001B[31mGuard\u001B[0m: What do you want?"
498            option_q = "Nevermind."
499
500  [medical_centre_office]
501  enter = '''
502  You find yourself at the Medical Centre's office.
503  You definitely shouldn't be here...
504  '''
505
506        [medical_centre_office.books]
507            [medical_centre_office.books.1]
508            title = "Alan Stern. Chasing New Horizons"
509            contents = """
510  On July 14, 2015, something amazing happened. More than 3 billion miles from
511  Earth, a small NASA spacecraft called New Horizons screamed past Pluto at more
512  than 32,000 miles per hour, focusing its instruments on the long mysterious icy
513  worlds of the Pluto system, and then, just as quickly, continued on its journey
514  out into the beyond.
515  """
516
517            [medical_centre_office.books.2] # yes
518            title = "Stack of research papers"
519            contents = """
520  Upon our latest series of tests, we have found the effect of \u001B[47m\u001B[37m[redacted]\u001B[0m has
521  considerable negative effects. These tests were performed over a series of 8
522  days and caused \u001B[47m\u001B[37m[rdctd]\u001B[0m and \u001B[47m\u001B[37m[redact]\u001B[0m in the body.
523
524  This document is confidential and should not be shared with third parties.
525  """
526
527            [medical_centre_office.books.3]
528            title = "Barry A. Burd. Java For Dummies"
529            contents = """
530  Java is the platform—independent, object—oriented programming language used for
531  developing web and mobile applications. The revised version offers new
532  functionality and features that have programmers excited, and this popular guide
```

```
533    covers them all.
534
535    You will learn how to program good in Java, like this:
536
537    \u001B[36mimport \u001B[33mjava\u001B[37m.\u001B[33mio\u001B[37m.\u001B[33mBufferedReader\u001B[37m;
538    \u001B[36mimport \u001B[33mjava\u001B[37m.\u001B[33mio\u001B[37m.\u001B[33mInputStreamReader\u001B[37m;
539    \u001B[33mpublic class \u001B[31mMain  \u001B[34m{
540        \u001B[33mpublic static  void \u001B[31mmain  \u001B[34m(\u001B[33mString[] args\u001B[34m) \u001B[34m{
541            \u001B[33mtry{ String a=new \u001B[31mBufferedReader \u001B[34m(
542                \u001B[33mnew \u001B[31mInputStreamReader\u001B[34m( System\u001B[37m.\u001B[33mout\u001B[37m \u001B[34m))
\u001B[37m.
543                  \u001B[31mreadLine() \u001B[37m;
544            \u001B[33mif
545                \u001B[34m(a == \u001B[32m"hi"\u001B[34m) {
546                System\u001B[37m.\u001B[33mout\u001B[37m.\u001B[31mprintln\u001B[34m(\u001B[32m"Hello!!"\u001B[34m)
\u001B[37m;
547            \u001B[34m} \u001B[33melse
548            \u001B[34m{
549                System\u001B[37m.\u001B[33mout\u001B[37m.\u001B[31mprintln\u001B[34m(\u001B[32m"you did not greet
me!"\u001B[34m) \u001B[37m;
550            \u001B[34m}
551        } \u001B[33mcatch \u001B[34m(\u001B[33mException e\u001B[34m) {\u001B[35m/*something bad happen! */ \u001B[34m} }
552    \u001B[0m"""
553
554            [medical_centre_office.books.4] # yes
555            title = "Dr. Mike Ox. Introducing Impostors in Society"
556            contents = """
557    [Stamped for internal use only.]
558
559    This book details methods which we can use to manipulate and control the general
560    populous, I will be going into detail about ways we can subtly remove dangerous
561    ideas and to keep your secrets under covers.
562    """
563
564            [medical_centre_office.books.5] # yes
565            title = "RNA sequences"
566            contents = """
567    Confidential document
568
569    Figure 1.
570    GAGAATAAACTAGTATTCTTCTGGTCCCCACAGACTCAGAGAGAACCCGCCACCATGTTCGTGTTCCTGGTGCTGCTGCC
```

```toml
571 TCTGGTGTCCAGCCAGTGTGTGAACCTGACCACCAGAACACAGCTGCCTCCAGCCTACACCAACAGCTTTACCAGAGGCG
572 TGTACTACCCCGACAAGGTGTTCAGATCCAGCGTGCTGCACTCTACCCAGGACCTGTTCCTGCCTTTCTTCAGCAACGTG
573 ACCTGGTTCCACGCCATCCACGTGTCCGGCACCAATGGCACCAAGAGATTCGACAACCCCGTGCTGCCCTTCAACGACGG
574 GGTGTACTTTGCCAGCACCGAGAAGTCCAACATCATCAGAGGCTGGATCTTCGGCACCACACTGGACAGCAAGACCCAGA
575 GCCTGCTGATCGTGAACAACGCCACCAACGTGGTCATCAAAGTGTGCGAGTTCCAGTTCTGCAACGACCCCTTCCTGGGC
576 GTCTACTACCACAAGAACAACAAGAGCTGGATGGAAAGCGAGTTCCGGGTGTACAGCAGCGCCAACAACTGCACCTTCGA
577 GTACGTGTCCCAGCCTTTCCTGATGGACCTGGAAGGCAAGCAGGGCAACTTCAAGAACCTGCGCGAGTTCGTGTTTAAGA
578 ACATCGACGGCTACTTCAAGATCTACAGCAAGCACACCCCTATCAACCTCGTGCGGGATCTGCCTCAGGGCTTCTCTGCT
579 CTGGAACCCCTGGTGGATCTGCCCATCGGCATCAACATCACCCGGTTTCAGACACTGCTGGCCCTGCACAGAAGCTACCT
580 GACACCTGGCGATAGCAGCAGCGGATGGACAGCTGGTGCCGCCGCTTACTATGTGGGCTACCTGCAGCCTAGAACCTTCC
581 TGCTGAAGTACAACGAGAACGGCACCATCACCGACGCCGTGGATTGTGCTCTGGATCCTCTGAGCGAGACAAAGTGCACC
582 """
583
584         [medical_centre_office.books.6]
585         title = "Jules Verne. Around the World in Eighty Days"
586         contents = """
587 One ill-fated evening at the Reform Club, Phileas Fogg rashly bets his
588 companions £20,000 that he can travel around the entire globe in just eighty
589 days - and he is determined not to lose. Breaking the well-established routine
590 of his daily life, the reserved Englishman immediately sets off for Dover,
591 accompanied by his hot-blooded French manservant Passepartout. Travelling by
592 train, steamship, sailing boat, sledge and even elephant, they must overcome
593 storms, kidnapping, natural disaster, Sioux attacks and the dogged Inspector Fix
594 of Scotland Yard - who believes that Fogg has robbed the Bank of England - to
595 win the extraordinary wager.
596 """
597
598 [marie]
599 description = "\u001B[35mMarie the Mink\u001B[0m"
600
601     # Purple: \u001B[35m
602     [marie.alley]
603
604         [marie.alley.small_talk]
605         dialog = "\u001B[35mMarie\u001B[0m: What brings you here?"
606         option_1 = "Did you hear about the protests?"
607         option_2 = "Sylvasta is up to something..."
608         option_q = "Nevermind, have a good day."
609
610         [marie.alley.protests]
611         dialog = """"\u001B[35mMarie\u001B[0m: Yeah, those Sylvasta guys are definitely up to
```

```
612   something but I don't know whether I believe the crowds showing up."""
613           option_1 = "Do you want to help me find out?"
614           option_q = "Nevermind."
615
616           [marie.alley.sylvasta]
617           dialog = "\u001B[35mMarie\u001B[0m: You think I don't know? You want to help me find out?"
618           option_1 = "Sure."
619           option_q = "No thanks."
620
621           [marie.alley.confirm]
622           dialog = "\u001B[35mMarie\u001B[0m: I suppose we could break in and find out the truth."
623           option_1 = "Let's do it! [progress story]"
624           option_q = "No way!"
625
626           [marie.alley.recon]
627           dialog = """\u001B[35mMarie\u001B[0m: Ok, while I go figure out how we get in, I need you to fetch something we
628   can talk over, I'm sure the shop will have something, meet me back here when you
629   get something."""
630           option_q = "Got it."
631
632           [marie.alley.waiting]
633           dialog = "\u001B[35mMarie\u001B[0m: Well? Go get it."
634           option_q = "Ok."
635
636           [marie.alley.mission_brief]
637           dialog = """\u001B[35mMarie\u001B[0m: Ok so the plan is, you go to the Medical Centre, casually stroll in to
638   the reception, try to blend in though... Maybe find a chair to sit on or
639   something, either way, once you're there and you think security isn't looking,
640   use your comms device to let me know.
641
642   \u001B[35mMarie\u001B[0m: From there, I will send a distraction and lead them into their breakroom
643   while you sneak into the office and try to find anything you can... although
644   once you're down there, let me know.
645
646   \u001B[35mMarie\u001B[0m: If you want to, you can test the comms device now."""
647           option_q = "Ok."
648
649       [marie.comms]
650       received = """\u001B[35mMarie\u001B[0m: Alright, I've received the documents, I'm sending them out far and wide,
651   good job on getting in and getting these."""
652       bad_documents = "\u001B[35mMarie\u001B[0m: These don't look like the right documents..."
```

```toml
653        no_access = "\u001B[31mAccess denied!\u001B[0m"
654
655            [marie.comms.orientation]
656            dialog = "\u001B[35mMarie\u001B[0m: What do you want?"
657            option_1 = "Where do I go again?"
658            option_2 = "Can you hear me?"
659            option_q = "Nevermind."
660
661            [marie.comms.directions]
662            dialog = """\u001B[35mMarie\u001B[0m: You need to go to the Medical Centre, once you're there, try to blend in
663    and let me know once you're in position."""
664            option_q = "Got it."
665
666            [marie.comms.hear]
667            dialog = "\u001B[35mMarie\u001B[0m: Loud and clear."
668            option_q = "Neat."
669
670            [marie.comms.complaint]
671            dialog = "\u001B[35mMarie\u001B[0m: Try to blend in..."
672            option_q = "Ok."
673
674            [marie.comms.in_position]
675            dialog = "\u001B[35mMarie\u001B[0m: Ok, I'll create a distraction, give me a moment..."
676            option_1 = "[wait]"
677
678            [marie.comms.distraction]
679            dialog = """\u001B[31mGuard\u001B[0m: Who's there?
680    You better come out or we're going to have a problem.
681
682    (you see the guard go into the breakroom)
683    """
684            option_1 = "[wait]"
685
686            [marie.comms.coast_is_clear]
687            dialog = "\u001B[35mMarie\u001B[0m: Ok, now is your chance."
688            option_q = "Got it."
689
690            [marie.comms.gogogo]
691            dialog = "\u001B[35mMarie\u001B[0m: What are you waiting for??"
692            option_q = "AAAAAAA"
693
```

```
694        [marie.comms.office]
695        dialog = "\u001B[35mMarie\u001B[0m: What do you see?"
696        option_1 = "A bunch of documents scattered around."
697
698        [marie.comms.documents]
699        dialog = """\u001B[35mMarie\u001B[0m: Ok, look through them and find anything that looks relevant, look out for
700 anything that mentions the stuff they were protesting about. Once you do, you'll
701 need to upload them to me, I guess go back to your apartments and we'll do it
702 from there, let me know once you're there.
703
704 \u001B[35mMarie\u001B[0m: I'll be making sure your cover is not blown.
705
706 \u001B[33mHint: you can use the documents to read them.\u001B[0m"""
707        option_q = "Ok."
708
709        [marie.comms.home]
710        dialog = """\u001B[35mMarie\u001B[0m: Presumably you have the documents, turn your computer on, go to the link
711 I've sent you, it should show up, and upload them."""
712
713 [stage]
714 reached_conclusion = """\n
715 You've reached the end of the game, you're now in an open world mode, keep
716 exploring anything you haven't to maximise your score.
717
718 You can conclude by typing \u001B[47m\u001B[30mwin\u001B[0m
719 """
```

```toml
 1   # 1. Marie located in Back Alley
 2   [npc_marie]
 3       _prefix = "marie.alley."
 4       _start = "small_talk"
 5       small_talk = { description = "small_talk.dialog", options = [
 6           { description = "small_talk.option_1", to = "protests" },
 7           { description = "small_talk.option_2", to = "sylvasta" },
 8           { description = "small_talk.option_q", to = "small_talk", mustExit = true }
 9       ] }
10       protests = { description = "protests.dialog", options = [
11           { description = "protests.option_1", to = "confirm" },
12           { description = "protests.option_q", to = "small_talk", mustExit = true }
13       ] }
14       sylvasta = { description = "sylvasta.dialog", options = [
15           { description = "sylvasta.option_1", to = "confirm" },
16           { description = "sylvasta.option_q", to = "small_talk", mustExit = true }
17       ] }
18       confirm = { description = "confirm.dialog", options = [
19           { description = "confirm.option_q", to = "small_talk", mustExit = true }
20       ] }
21
22       recon = { description = "recon.dialog", options = [
23           { description = "recon.option_q", to = "waiting", mustExit = true }
24       ] }
25       waiting = { description = "waiting.dialog", options = [
26           { description = "waiting.option_q", to = "waiting", mustExit = true }
27       ] }
28
29       mission_brief = { description = "mission_brief.dialog", options = [
30           { description = "mission_brief.option_q", to = "mission_brief", mustExit = true }
31       ] }
32
33   # 2. Random City NPC located in the City Centre
34   [npc_city_centre]
35       _prefix = "city_centre.npc."
36       _start = "small_talk"
37       small_talk = { description = "small_talk.dialog", options = [
38           { description = "small_talk.option_1", to = "protestors" },
39           { description = "small_talk.option_2", to = "enquire" },
40           { description = "small_talk.option_q", to = "small_talk", mustExit = true }
41       ] }
```

```toml
    protestors = { description = "protestors.dialog", options = [
        { description = "protestors.option_1", to = "leaflet" },
        { description = "small_talk.option_q", to = "small_talk", mustExit = true }
    ] }
    enquire = { description = "enquire.dialog", options = [
        { description = "protestors.option_1", to = "leaflet", },
        { description = "small_talk.option_q", to = "small_talk", mustExit = true }
    ] }
    leaflet = { description = "leaflet.dialog", options = [
        { description = "leaflet.option_1", to= "small_talk", mustExit = true }
    ] }
    #recon = { description = "testing", options = [
    #    { description = "sususus!", to = "recon", mustExit = true }
    #] }

# 3. Old Man located in Forest
[npc_old_man]
    _prefix = "forest.old_man."
    _start = "small_talk"
    small_talk = { description = "small_talk.dialog", options = [
        { description = "small_talk.option_1", to = "request" },
        { description = "small_talk.option_q", to = "small_talk", mustExit = true }
    ] }
    request = { description = "request.dialog", options = [
        { description = "request.option_1", to = "small_talk", mustExit = true },
        { description = "request.option_q", to = "small_talk", mustExit = true }
    ] }
    praise = { description = "praise.dialog", options = [
        { description = "praise.option_q", to = "praise", mustExit = true }
    ] }

# 4. Protestors located on Street
[npc_protestors]
    _prefix = "street.protestors."
    _start = "small_talk"
    small_talk = { description = "small_talk.dialog", options = [
        { description = "small_talk.option_1", to = "enquire" },
        { description = "small_talk.option_2", to = "small_talk", mustExit = true }
    ] }
    enquire = { description = "enquire.dialog", options = [
        { description = "enquire.option_1", to = "leaks" },
```

```toml
 83            { description = "enquire.option_2", to = "confrontation" }
 84        ] }
 85        leaks = { description = "leaks.dialog", options = [
 86            { description = "leaks.option_1", to = "small_talk", mustExit = true }
 87        ] }
 88        confrontation = { description = "confrontation.dialog", options = [
 89            { description = "confrontation.option_1", to = "small_talk", mustExit = true },
 90            { description = "confrontation.option_2", to = "small_talk", mustExit = true }
 91        ] }
 92
 93    # 5. Security guard stationed at Medical Centre
 94    [npc_security_guard]
 95        _prefix = "medical_centre.guard."
 96        _start = "small_talk"
 97        small_talk = { description = "small_talk.dialog", options = [
 98            { description = "small_talk.option_q", to = "small_talk", mustExit = true }
 99        ] }
100
101    # 6. Shop shopkeeper in the Shop
102    [npc_shopkeeper]
103        _prefix = "shop.npc."
104        _start = "index"
105        index = { description = "index", options = [
106            { description = "leave", to = "index", mustExit = true }
107        ] }
108        recon = { description = "recon", options = [
109            { description = "leave", to = "recon", mustExit = true }
110        ] }
111        stealth = { description = "stealth", options = [
112            { description = "leave", to = "stealth", mustExit = true }
113        ] }
114
115    # 7. Receptionist located in Apartments
116    [npc_receptionist]
117        _prefix = "apartments.receptionist."
118        _start = "first_encounter"
119        first_encounter = { description = "first_encounter.dialog", options = [
120            { description = "first_encounter.option_1", to = "protestors" },
121            { description = "first_encounter.option_q", to = "repeated", mustExit = true }
122        ] }
123        protestors = { description = "protestors.dialog", options = [
```

```toml
124          { description = "protestors.option_1", to = "protestors2" },
125          { description = "protestors.option_q", to = "repeated" }
126      ] }
127      protestors2 = { description = "protestors2.dialog", options = [
128          { description = "protestors2.option_q", to = "repeated" }
129      ] }
130      repeated = { description = "repeated.dialog", options = [
131          { description = "repeated.option_1", to = "protestors" },
132          { description = "repeated.option_q", to = "repeated", mustExit = true }
133      ] }

# 8. TV located in Apartments
[home_tv]
    _prefix = "home.tv."
    _start = "first_on"
    first_on = { description = "first_on.dialog", options = [
        { description = "keep_watching", to = "dialog_a" },
        { description = "off", to = "first_on", mustExit = true }
    ] }
    dialog_a = { description = "first_on.dialog_a", options = [
        { description = "keep_watching", to = "dialog_b" },
        { description = "off", to = "first_on", mustExit = true }
    ] }
    dialog_b = { description = "first_on.dialog_b", options = [
        { description = "keep_watching", to = "dialog_c" },
        { description = "off", to = "first_on", mustExit = true }
    ] }
    dialog_c = { description = "first_on.dialog_c", options = [
        { description = "off", to = "first_on", mustExit = true }
    ] }

# 9. Communicator device with Marie
[comms_marie]
    _prefix = "marie.comms."
    _start = "orientation"

    orientation = { description = "orientation.dialog", options = [
        { description = "orientation.option_1", to = "directions" },
        { description = "orientation.option_2", to = "hear" },
        { description = "orientation.option_q", to = "orientation", mustExit = true }
    ] }
```

```
165        directions = { description = "directions.dialog", options = [
166            { description = "directions.option_q", to = "orientation" }
167        ] }
168        hear = { description = "hear.dialog", options = [
169            { description = "hear.option_q", to = "orientation" }
170        ] }
171
172        complaint = { description = "complaint.dialog", options = [
173            { description = "complaint.option_q", to = "complaint", mustExit = true }
174        ] }
175
176        in_position = { description = "in_position.dialog", options = [
177            { description = "in_position.option_1", to = "distraction" }
178        ] }
179        distraction = { description = "distraction.dialog", options = [
180            #{ description = "in_position.option_1", to = "coast_is_clear" }
181        ] }
182        coast_is_clear = { description = "coast_is_clear.dialog", options = [
183            { description = "coast_is_clear.option_q", to = "gogogo", mustExit = true }
184        ] }
185        gogogo = { description = "gogogo.dialog", options = [
186            { description = "gogogo.option_q", to = "gogogo", mustExit = true }
187        ] }
188
189        office = { description = "office.dialog", options = [
190            { description = "office.option_1", to = "documents" }
191        ] }
192        documents = { description = "office.dialog", options = [
193            { description = "office.option_q", to = "office", mustExit = true }
194        ] }
195
196        home = { description = "home.dialog", options = [
197            { description = "home.option_q", to = "home", mustExit = true }
198        ] }
199
200    # 10. Laptop
201    [entity_laptop]
202        _prefix = "entities.laptop."
203        _start = "boot"
204
205        boot = { description = "boot.dialog", options = [
```

```toml
206          { description = "boot.option_1", to = "home" }
207      ] }
208      home = { description = "home.dialog", options = [
209          { description = "home.option_q", to = "boot", mustExit = true },
210          { description = "home.option_1", to = "pictures" },
211          { description = "home.option_2", to = "funny_cat_videos" },
212          { description = "home.option_3", to = "document" }
213      ] }
214      pictures = { description = "pictures.dialog", options = [
215          { description = "pictures.option_q", to = "home" }
216      ] }
217      document = { description = "document.dialog", options = [
218          { description = "document.option_q", to = "home" }
219      ] }
```

```toml
1    # Path to emoji root directory
2    emojis = "/emojis"
3
4    # Path to font file
5    # https://fonts.google.com/specimen/VT323?category=Monospace#standard-styles
6    font = "/VT323-Regular.ttf"
```