

World of Deez nuts

Pawel Makles
(K21002534)

Created: 19th November 2021
Last Modified: 3rd December 2021

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 Beastman¹ society, the city being loosely based off Animacity² from the series Brand New Animal.⁴ The plot revolves Sylvesta who are deeply intertwined with the city and own 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 are finding 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, quitting 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`.

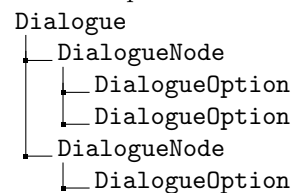
To map these, I had a `HashMap` which mapped unique `Class<? extends Event>` to linked (as to keep order of registration) hash sets of `EventListener<? extends Event>`. The interface simply provides a single method, `onEvent(Event e)`, which is called when the event is triggered.

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:



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 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 throughout 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 → 1 → 2 → 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 → 4 → 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 not changing between chapters.
- 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>

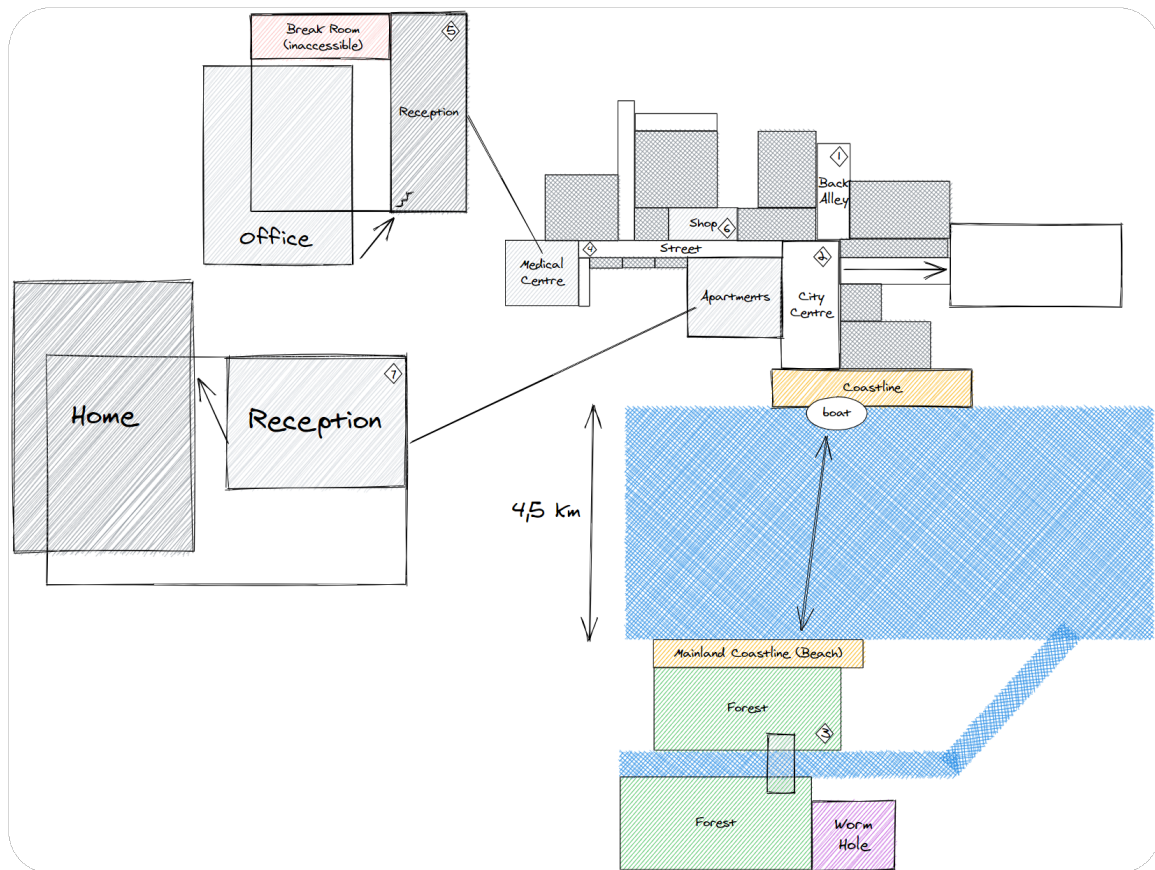


Figure 1: World Map



Figure 2: Shot of Animaparc as seen in Episode 1 at 3:02 of Brand New Animal⁴



Figure 3: Michiru Kagemori and Marie Itami pictured left to right in Episode 1 at 12:51 of Brand New Animal⁴

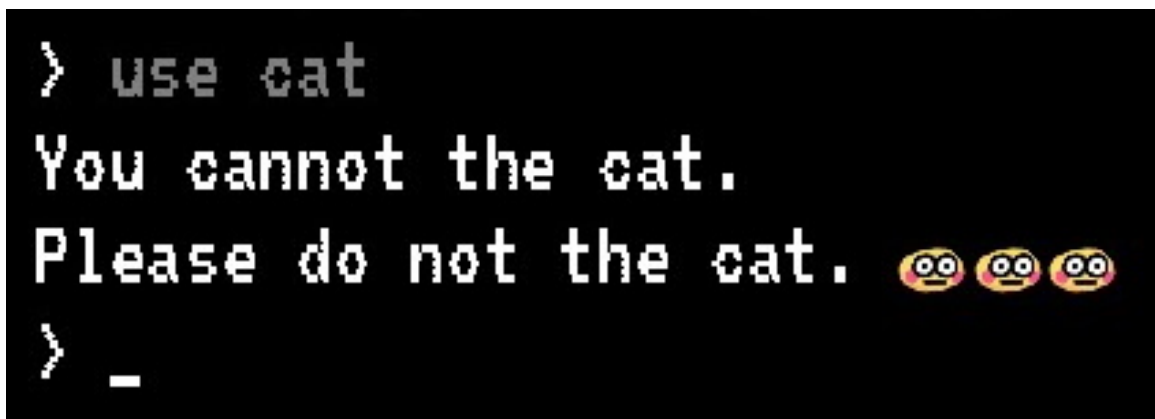


Figure 4: Sample emoji output

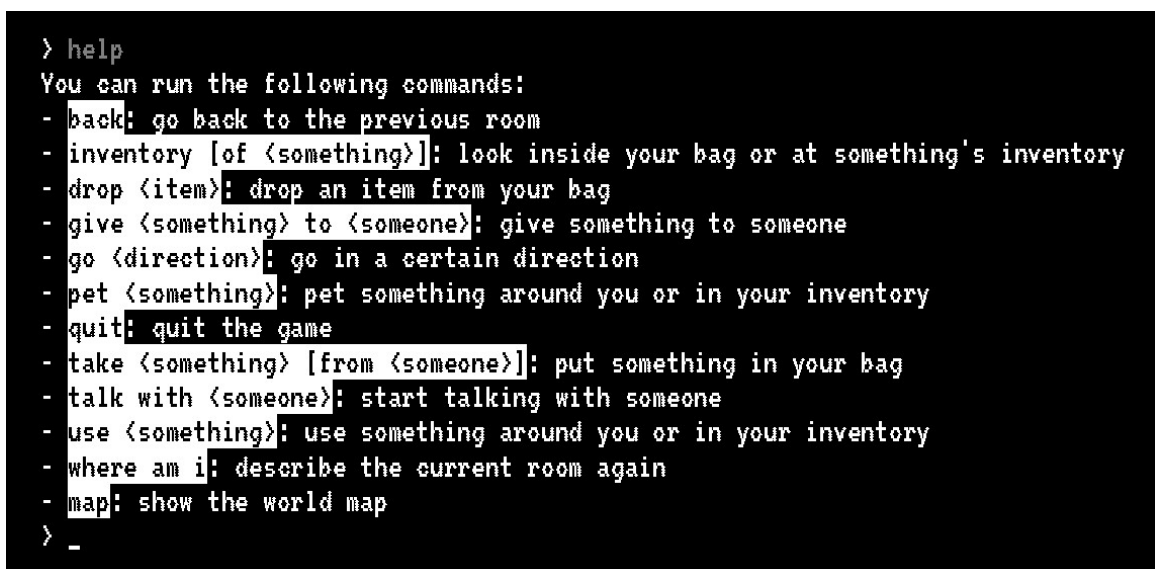


Figure 5: Output from the help command

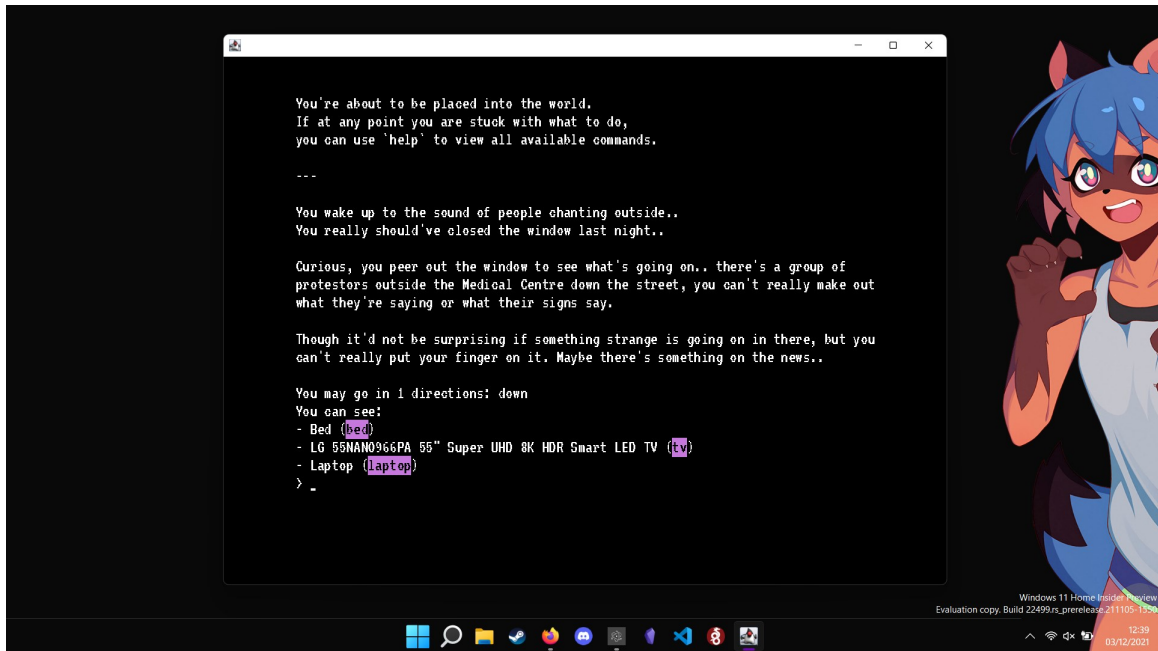


Figure 6: The terminal emulator

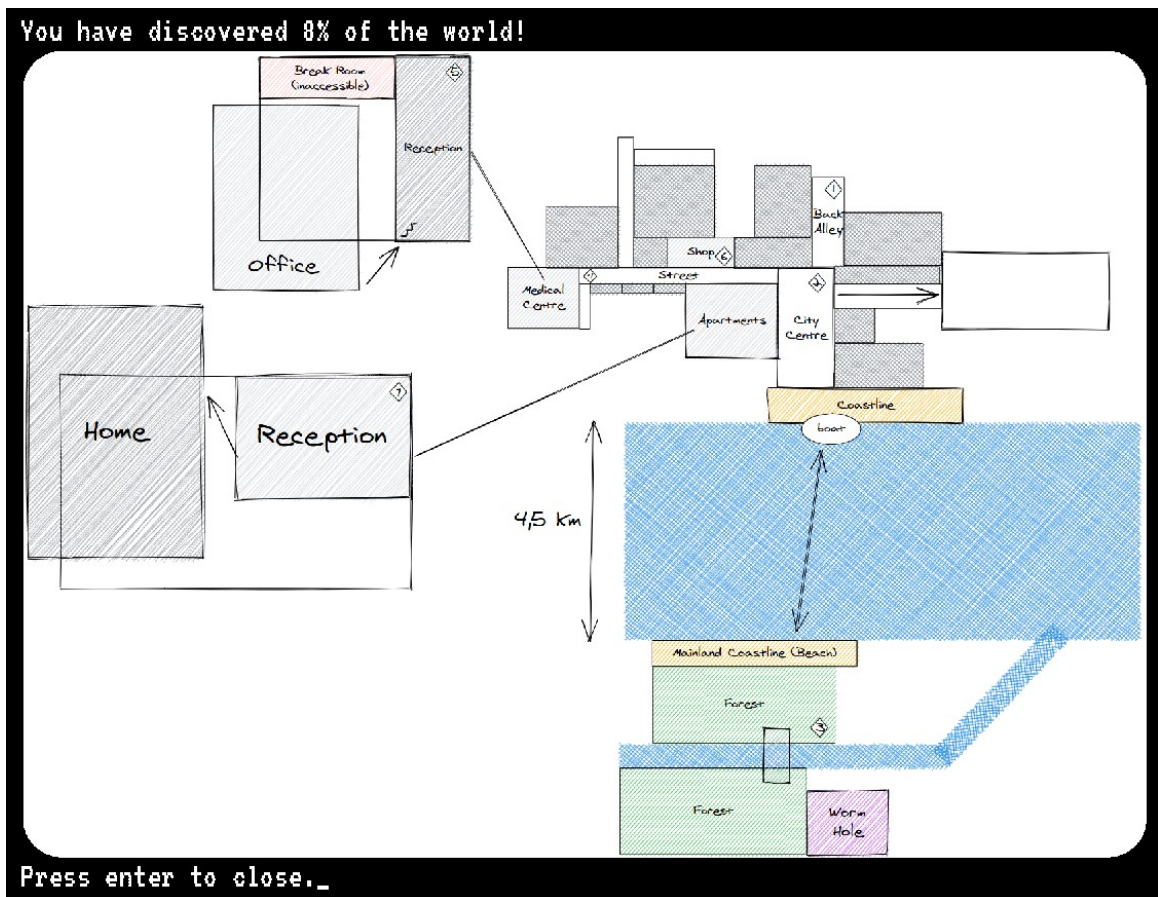


Figure 7: Output from the map command


```

1 package uk.insrt.coursework.zuul.events;
2
3 /**
4  * Represents a single event fired from
5  * any source to be consumed by anything.
6  */
7 public class Event {
8     private boolean propagating = true;
9
10    /**
11     * Whether this event can continue running.
12     * @return Whether propagation of this event was stopped
13     */
14    public boolean canRun() {
15        return this.propagating;
16    }
17
18    /**
19     * Stop further propagation of this event.
20     */
21    public void stopPropagation() {
22        this.propagating = false;
23    }
24 }

```

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

```

1 package uk.insrt.coursework.zuul.io;
2
3 /**
4  * Interface representing an arbitrary IO system.
5  * This can be implemented to input or output from various interfaces.
6  */
7 public interface IOSystem {
8     /**
9      * Print a string out through an arbitrary output channel.
10     * @param out String to print
11     */
12     public void print(String out);
13
14     /**
15      * Print a string out through an arbitrary output channel and append {@code \n}.
16      * @param out String to print
17      */
18     public void println(String out);
19
20     /**
21      * Read a String up until the first encountered {@code \n} from an arbitrary
22      * input channel.
23      * @return String of line read in
24      */
25     public String readLine();
26
27     /**
28      * Dispose of the arbitrary input and output channels.
29      */
30     public void dispose();
31 }

```

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

```

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 public class StandardIO implements IOSystem {
10     private Scanner reader;
11
12     /**
13      * Construct a new StandardIO.
14      */
15     public StandardIO() {
16         this.reader = new Scanner(System.in);
17     }
18
19     @Override
20     public void print(String out) {
21         System.out.print(out);
22     }
23
24     @Override
25     public void println(String out) {
26         System.out.println(out);
27     }
28
29     @Override
30     public String readLine() {
31         return this.reader.nextLine();
32     }
33
34     @Override
35     public void dispose() {}
36 }

```

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

```

1 package uk.insrt.coursework.zuul.io;
2
3 import java.util.regex.Matcher;
4 import java.util.regex.Pattern;
5
6 import uk.insrt.coursework.zuul.util.Localisation;
7
8 /**
9  * Translate and localise any incoming output.
10  */
11 public class LocalisedIO implements IOSystem {
12     private final Pattern pattern = Pattern.compile("<([\\w\\.]+?)>");
13
14     private IOSystem io;
15     private Localisation locale;
16
17     /**
18      * Construct a new LocalisedIO.
19      * @param io Provided IO system we should feed into
20      * @param locale Locale to apply to any i18n strings
21      */
22     public LocalisedIO(IOSystem io, Localisation locale) {
23         this.io = io;
24         this.locale = locale;
25     }
26
27     /**
28      * Replace i18n strings in any given String with their actual localised values.
29      * Using replacement code from https://stackoverflow.com/a/27359491.
30      * @param input String to process
31      * @return Final processed string
32      */
33     private String replace(String input) {
34         StringBuffer result = new StringBuffer();
35         Matcher matcher = this.pattern.matcher(input);
36
37         while (matcher.find()) {
38             matcher.appendReplacement(result, this.locale.get(matcher.group(1)));
39         }
40
41         matcher.appendTail(result);
42         return result.toString();
43     }
44
45     @Override
46     public void print(String out) {
47         this.io.print(this.replace(out));
48     }
49
50     @Override
51     public void println(String out) {
52         this.io.println(this.replace(out));
53     }
54
55     @Override
56     public String readLine() {
57         return this.io.readLine();
58     }
59
60     @Override
61     public void dispose() {
62         this.io.dispose();
63     }
64 }

```

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

```

1 package uk.insrt.coursework.zuul.events;
2
3 import java.util.HashMap;
4 import java.util.HashSet;
5 import java.util.LinkedHashSet;
6
7 /**
8  * Event system which manages taking in events
9  * from different sources and handles them
10  * by firing callbacks on event listeners.
11  */
12 public class EventSystem {
13     private HashMap<Class<? extends Event>, LinkedHashSet<EventListener<? extends
14         Event>>> listeners = new HashMap<>();
15
16     /**
17      * Get existing Event listener list or create a new one if not exists.
18      * @param event Event
19      * @return Set of event listeners
20      */
21     private HashSet<EventListener<? extends Event>> getList(Class<? extends Event>
22         event) {
23         var list = this.listeners.get(event);
24         if (list == null) {
25             list = new LinkedHashSet<>();
26             this.listeners.put(event, list);
27         }
28
29         return list;
30     }
31
32     /**
33      * Add a new event listener to this system.
34      * @param <E> Generic Event type
35      * @param event Event to remove from
36      * @param listener Event listener callback
37      */
38     public<E extends Event> void addListener(Class<E> event, IEventListener<E>
39         listener) {
40         this.getList(event).add(listener);
41     }
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>
50         listener) {
51         this.getList(event).remove(listener);
52     }
53
54     /**
55      * Emit an Event.
56      * @param <E> Generic Event type
57      * @param event Event to emit
58      */
59     @SuppressWarnings("unchecked")
60     public <E extends Event> void emit(E event) {
61         var listeners = this.listeners.get(event.getClass());
62         if (listeners == null) return;
63
64         for (@SuppressWarnings("rawtypes") IEventListener listener : listeners) {
65             listener.onEvent(event);
66             // Previously, there was a try catch ClassCastException
67             // but I've since constricted the types on 'addListener'
68             // and 'removeListener' so this should never happen.
69
70             if (!event.canRun())
71                 break;
72         }
73     }
74 }

```

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


```

1 package uk.insrt.coursework.zuul.dialogue;
2
3 import uk.insrt.coursework.zuul.io.IOSystem;
4
5 /**
6  * An option which branches off a {@link DialogueNode} into another node.
7  */
8 public class DialogueOption<T> {
9     private IDialogueHandler<T> handler;
10
11     private String description;
12     private boolean shouldExit;
13     private T target;
14
15     /**
16      * Construct a new simple DialogueOption with a description and destination.
17      * @param description Description of this option
18      * @param target Target node to jump to
19      */
20     public DialogueOption(String description, T target) {
21         this.target = target;
22         this.description = description;
23     }
24
25     /**
26      * Construct a complex DialogueOption with a description and select handler.
27      * @param description Description of this option
28      * @param handler Method called when this option is selected
29      */
30     public DialogueOption(String description, IDialogueHandler<T> handler) {
31         this.handler = handler;
32         this.description = description;
33     }
34
35     /**
36      * Tell the Dialogue system to exit if this option is selected.
37      * @return This dialogue option so method calls can be chained
38      */
39     public DialogueOption<T> mustExit() {
40         this.shouldExit = true;
41         return this;
42     }
43
44     /**
45      * Get the description of this option.
46      * @return Description string
47      */
48     public String getDescription() {
49         return this.description;
50     }
51
52     /**
53      * Get the destination of this option.
54      * @return Destination if it exists
55      */
56     public T getTarget() {
57         return this.target;
58     }
59
60     /**
61      * Handle the player selecting this dialogue option.
62      * @param io Provided IO system
63      * @return The new node or null if we should exit and stay put.
64      */
65     public T handle(IOSystem io) {
66         if (this.handler != null) {
67             return this.handler.onAction(io);
68         } else if (!this.shouldExit) {
69             return this.target;
70         }
71
72         return null;
73     }
74 }

```

Figure 13: DialogueOption.java
(uk.insrt.coursework.zuul.dialogue.DialogueOption)

```

1 package uk.insrt.coursework.zuul.ui;
2
3 import java.awt.Graphics;
4
5 import uk.insrt.coursework.zuul.events.Event;
6
7 /**
8  * Event fired when the terminal emulator draws a new frame.
9  */
10 public class EventDraw extends Event {
11     private Graphics g;
12     private float ox;
13     private float oy;
14     private float fw;
15     private float fh;
16
17     /**
18      * Construct a new EventDraw Event.
19      * @param g Graphics context
20      * @param ox Origin X position
21      * @param oy Origin Y position
22      * @param fw Font character width
23      * @param fh Font character height
24      */
25     public EventDraw(Graphics g, float ox, float oy, float fw, float fh) {
26         this.g = g;
27         this.ox = ox;
28         this.oy = oy;
29         this.fw = fw;
30         this.fh = fh;
31     }
32
33     /**
34      * Get the Graphics relating to this event.
35      * @return Graphics
36      */
37     public Graphics getGraphics() {
38         return this.g;
39     }
40
41     /**
42      * Get the origin X position of the contents of the terminal.
43      * @return X position
44      */
45     public float getOriginX() {
46         return this.ox;
47     }
48
49     /**
50      * Get the origin Y position of the contents of the terminal.
51      * @return Y position
52      */
53     public float getOriginY() {
54         return this.oy;
55     }
56
57     /**
58      * Get the character width.
59      * @return Character width
60      */
61     public float getCharWidth() {
62         return this.fw;
63     }
64
65     /**
66      * Get the character height.
67      * @return Character height
68      */
69     public float getCharHeight() {
70         return this.fh;
71     }
72 }

```

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

```

1 public<E extends Event> void addListener(Class<E> event ,
2     IEventListener<E> listener) {
3     this.getList(event).add(listener);
4 }

```

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

```

1 /**
2  * Get the current weight of this inventory.
3  * @return Weight (in kg)
4  */
5 public double getWeight() {
6     return this
7         .items
8         .stream()
9         .mapToDouble(Entity::getWeight)
10        .sum();
11 }
12
13 /**
14  * Check if the inventory is full.
15  * @return True if the weight is greater than the max weight
16  */
17 public boolean isFull() {
18     return this.getWeight() >= this.getMaxWeight();
19 }
20
21 /**
22  * Add an entity to this inventory.
23  *
24  * There must be sufficient space for the entity.
25  * @param entity Target Entity
26  * @return Whether we successfully added the new entity.
27  */
28 public boolean add(Entity entity) {
29     if (this.getWeight() + entity.getWeight() > this.maxWeight) {
30         return false;
31     }
32
33     this.items.add(entity);
34     return true;
35 }

```

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

```

1  /**
2   * Write a string value to the text buffer.
3   * @param value String value to write
4   */
5  public void write(String value) {
6      // Write each character sequentially.
7      for (int i=0;i<value.length();i++) {
8          char c = value.charAt(i);
9
10         // If we encounter an Ansi escape character, then take the
11         // substring from this point on and determine if it is a valid
12         // escape code. If it is, apply any changes before continuing.
13         if (c == '\u001B') {
14             Matcher matcher = Ansi.AnsiPattern.matcher(value.substring(i));
15             if (matcher.find()) {
16                 int v = Integer.parseInt(matcher.group(1));
17                 i += 3 + (v > 9 ? 1 : 0);
18
19                 if (v == 0) {
20                     this.bg = Color.BLACK;
21                     this.fg = Color.WHITE;
22                 } else if (v >= 30 && v < 38) {
23                     this.fg = Ansi.fromEscapeCode(v);
24                 } else if (v >= 40 && v < 48) {
25                     this.bg = Ansi.fromEscapeCode(v);
26                 }
27
28                 continue;
29             }
30         }
31
32         this.write(c);
33     }
34 }

```

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


```
1 package uk.insrt.coursework.zuul.commands;
2
3 import java.util.regex.Matcher;
4
5 import uk.insrt.coursework.zuul.world.Direction;
6
7 /**
8  * Wrapper around regex Matcher for deriving the values of given arguments in commands.
9  */
10 public class Arguments {
11     private Matcher matcher;
12
13     /**
14      * Construct a new Arguments wrapper.
15      * @param matcher Regex Matcher
16      */
17     public Arguments(Matcher matcher) {
18         this.matcher = matcher;
19     }
20
21     /**
22      * Take a named group from the Matcher.
23      * We assume that the provided Regex doesn't match the group if it's empty.
24      * @param group Named group
25      * @return String value of named group or null if it doesn't exist
26      */
27     public String group(String group) {
28         // We ignore and return null on error as a bit of convenience.
29         // This may not be best practice but it is justified by the fact
30         // that it avoids an incredible amount of boilerplate further up
31         // the chain, and in my opinion that's worth this design decision.
32         try {
33             return this.matcher.group(group);
34         } catch (Exception e) {
35             return null;
36         }
37     }
38
39     /**
40      * Check whether this named group was matched.
41      * @param group Named group
```

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

```
1  package uk.insrt.coursework.zuul.commands;
2
3  import java.util.List;
4  import java.util.regex.Pattern;
5
6  import uk.insrt.coursework.zuul.entities.Entity;
7  import uk.insrt.coursework.zuul.entities.Inventory;
8  import uk.insrt.coursework.zuul.util.Search;
9  import uk.insrt.coursework.zuul.world.World;
10
11  /**
12   * Representation of an action which can be performed by the user.
13   */
14  public abstract class Command {
15      private Pattern[] patterns;
16      private String syntax;
17      private String usage;
18
19      /**
20       * Construct a new Command.
21       * @param syntax Information about how to use the command
22       * @param usage Information about what the command does
23       * @param patterns Patterns to execute this command on
24       */
25      public Command(String syntax, String usage, Pattern[] patterns) {
26          this.patterns = patterns;
27          this.syntax = syntax;
28          this.usage = usage;
29      }
30
31      /**
32       * Get information about how to use the command.
33       * @return String Information about how to use the command
34       */
35      public String getSyntax() {
36          return this.syntax;
37      }
38
39      /**
40       * Get information about what the command does.
41       * @return String Information about what the command does
```

```
42     */
43     public String getUsage() {
44         return this.usage;
45     }
46
47     /**
48      * Get all applicable patterns to match to execute this command.
49      * @return Regex Pattern array
50      */
51     public Pattern[] getPatterns() {
52         return this.patterns;
53     }
54
55     /**
56      * Check whether this command is visible to the player in the help menu.
57      * @return True if visible
58      */
59     public boolean isVisible() {
60         return true;
61     }
62
63     /**
64      * Run this command within the scope of a world and with any parsed arguments.
65      * @param world Current World object
66      * @param args Arguments passed into command
67      * @return Boolean indicating whether the game loop should exit.
68      */
69     public abstract boolean run(World world, Arguments args);
70
71     /**
72      * Filter entities by those that are in the current room.
73      */
74     public static final int FILTER_ROOM = 1;
75
76     /**
77      * Filter entities by those that are in the player's inventory.
78      */
79     public static final int FILTER_INVENTORY = 2;
80
81     /**
82      * Don't filter entities and instead search through both the current room and the player's inventory.
```



```

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

```

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

```
161     }  
162 }
```

```
1  package uk.insrt.coursework.zuul.commands;
2
3  import java.util.ArrayList;
4  import java.util.List;
5  import java.util.regex.Matcher;
6  import java.util.regex.Pattern;
7
8  import uk.insrt.coursework.zuul.commands.core.CommandBack;
9  import uk.insrt.coursework.zuul.commands.core.CommandBag;
10 import uk.insrt.coursework.zuul.commands.core.CommandDrop;
11 import uk.insrt.coursework.zuul.commands.core.CommandGive;
12 import uk.insrt.coursework.zuul.commands.core.CommandGo;
13 import uk.insrt.coursework.zuul.commands.core.CommandHelp;
14 import uk.insrt.coursework.zuul.commands.core.CommandPet;
15 import uk.insrt.coursework.zuul.commands.core.CommandQuit;
16 import uk.insrt.coursework.zuul.commands.core.CommandTake;
17 import uk.insrt.coursework.zuul.commands.core.CommandTalk;
18 import uk.insrt.coursework.zuul.commands.core.CommandUse;
19 import uk.insrt.coursework.zuul.commands.core.CommandWhereAmI;
20 import uk.insrt.coursework.zuul.world.World;
21
22 /**
23  * Command handler which constructs, then resolves
24  * and executes commands from an arbitrary input.
25  */
26 public class CommandManager {
27     private ArrayList<Command> commands = new ArrayList<>();
28
29     /**
30      * Construct a new CommandManager.
31      *
32      * You should only need one present at any given time.
33      */
34     public CommandManager() {
35         this.initialiseCommands();
36     }
37
38     /**
39      * Register a new Command.
40      * @param command Command
41      */
```



```
42 public void registerCommand(Command command) {
43     this.commands.add(command);
44 }
45
46 /**
47  * Register multiple commands.
48  * @param commands Command array
49  */
50 public void registerCommands(Command[] commands) {
51     for (Command command : commands) {
52         this.registerCommand(command);
53     }
54 }
55
56 /**
57  * Get Commands provided by this Command manager.
58  * @return List of Commands
59  */
60 public List<Command> getCommands() {
61     return this.commands;
62 }
63
64 /**
65  * Initialise all the commands a player can execute.
66  */
67 private void initialiseCommands() {
68     final Command[] DEFAULT_COMMANDS = {
69         new CommandBack(),
70         new CommandBag(),
71         new CommandDrop(),
72         new CommandGive(),
73         new CommandGo(),
74         new CommandHelp(this),
75         new CommandPet(),
76         new CommandQuit(),
77         new CommandTake(),
78         new CommandTalk(),
79         new CommandUse(),
80         new CommandWhereAmI(),
81     };
82 }
```

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

```
1 package uk.insrt.coursework.zuul.commands.core;
2
3 import java.util.regex.Pattern;
4
5 import uk.insrt.coursework.zuul.commands.Arguments;
6 import uk.insrt.coursework.zuul.commands.Command;
7 import uk.insrt.coursework.zuul.world.World;
8
9 /**
10  * Command which allows the Player to walk back through the previous Rooms.
11  */
12 public class CommandBack extends Command {
13     public CommandBack() {
14         super("back", "<commands.back>",
15             new Pattern[] {
16                 Pattern.compile("^(?:?:go|walk)\\s+)*back(?:!\\w)"),
17                 // back, go back, walk back
18             });
19     }
20
21     @Override
22     public boolean run(World world, Arguments arguments) {
23         // We call a specialised method on the player as we keep
24         // track of visited rooms within the Player class itself.
25         world.getPlayer().back();
26         return false;
27     }
28 }
```

```
1 package uk.insrt.coursework.zuul.commands.core;
2
3 import java.text.DecimalFormat;
4 import java.util.regex.Pattern;
5
6 import uk.insrt.coursework.zuul.commands.Arguments;
7 import uk.insrt.coursework.zuul.commands.Command;
8 import uk.insrt.coursework.zuul.entities.Entity;
9 import uk.insrt.coursework.zuul.entities.Inventory;
10 import uk.insrt.coursework.zuul.io.Ansi;
11 import uk.insrt.coursework.zuul.io.IOSystem;
12 import uk.insrt.coursework.zuul.world.World;
13
14 /**
15  * Command which allows the Player to look at their or another Entity's inventory.
16  */
17 public class CommandBag extends Command {
18     private final DecimalFormat format = new DecimalFormat("0.00");
19
20     public CommandBag() {
21         super("inventory [of <selectors.something>]", "<commands.bag.usage>",
22             new Pattern[] {
23                 Pattern.compile("^(?:b(?:ag)*|inv(?:entory)*)(?:\\s+(?<entity>[\\w\\s]+))*"),
24                 // b, bag, inv, inventory, bag of <entity>, inventory of <entity>, (+2)
25             });
26     }
27
28     @Override
29     public boolean run(World world, Arguments arguments) {
30         IOSystem io = world.getIO();
31
32         // Figure out if we're checking our own inventory or another entity's inventory.
33         Entity entity;
34         boolean ours;
35         if (arguments.has("entity")) {
36             ours = false;
37             entity = this.findEntity(world, arguments, "<commands.bag.cant_find>");
38             if (entity == null) return false;
39         } else {
40             ours = true;
41             entity = world.getPlayer();
```

```
42     }
43
44     // Get the selected entity's inventory and provide output if empty.
45     Inventory inv = entity.getInventory();
46     if (inv.getWeight() == 0) {
47         if (ours) {
48             io.println("<commands.bag.empty> <commands.bag.can_carry_kg> "
49                 + inv.getMaxWeight() + " kg.");
50         } else {
51             io.println(entity.getHighlightedName() + " <commands.bag.entity_empty>.");
52         }
53
54         return false;
55     }
56
57     // Otherwise describe some statistics about the inventory.
58     if (ours) {
59         io.println("<commands.bag.are_carrying_kg> " + this.format.format(inv.getWeight())
60             + " / " + inv.getMaxWeight() + " kg.\n<commands.bag.look_in_bag>:");
61     } else {
62         io.println(entity.getHighlightedName() + " <commands.bag.entity_appears_to_have>:");
63     }
64
65     // Describe all the items in this inventory we are currently looking at.
66     for (Entity item : inv.getItems()) {
67         io.println("- " + Ansi.Yellow + item.getWeight() + " kg"
68             + Ansi.Reset + " " + item.describe()
69             + " (" + item.getHighlightedName() + ")");
70     }
71
72     return false;
73 }
74 }
```

```
1  package uk.insrt.coursework.zuul.commands.core;
2
3  import java.util.regex.Pattern;
4
5  import uk.insrt.coursework.zuul.commands.Arguments;
6  import uk.insrt.coursework.zuul.commands.Command;
7  import uk.insrt.coursework.zuul.entities.Entity;
8  import uk.insrt.coursework.zuul.io.Ansi;
9  import uk.insrt.coursework.zuul.world.World;
10
11  /**
12   * Command which allows the Player to drop any item in their inventory.
13   */
14  public class CommandDrop extends Command {
15      public CommandDrop() {
16          super("drop <selectors.item>", "<commands.drop.usage>",
17              new Pattern[] {
18                  Pattern.compile("^(?:drop|place|put down)(?:\\s+(?<entity>[\\w\\s]+))*")
19                  // drop, place, put down, drop <item>, place <item>, put down <item>
20              });
21      }
22
23      @Override
24      public boolean run(World world, Arguments args) {
25          // Find the given entity within our inventory and drop it if it's found.
26          Entity entity = this.findEntity(world, Command.FILTER_INVENTORY, args, "<commands.drop.nothing_specified>");
27          if (entity != null) {
28              world.getIO().println("<commands.drop.dropped.1> " + Ansi.BackgroundWhite + Ansi.Black
29                  + entity.getName() + Ansi.Reset + " <commands.drop.dropped.2>!");
30              entity.setLocation(world.getPlayer().getRoom());
31          }
32
33          return false;
34      }
35  }
```

```
1 package uk.insrt.coursework.zuul.commands.core;
2
3 import java.util.regex.Pattern;
4
5 import uk.insrt.coursework.zuul.commands.Arguments;
6 import uk.insrt.coursework.zuul.commands.Command;
7 import uk.insrt.coursework.zuul.entities.Entity;
8 import uk.insrt.coursework.zuul.entities.EntityPlayer;
9 import uk.insrt.coursework.zuul.entities.actions.IGiveable;
10 import uk.insrt.coursework.zuul.world.World;
11
12 /**
13  * Command which allows the player to give something to someone.
14  */
15 public class CommandGive extends Command {
16     public CommandGive() {
17         super("give <selectors.something> to <selectors.someone>", "<commands.give.usage>",
18             new Pattern[] {
19                 Pattern.compile("^(?:give|put)(?:\\s+(?<item>[\\w\\s]+)\\s+(?:to|in)\\s+(?<entity>[\\w\\s]+))*")
20                 // give, put, give <something> to <someone>, put <something> in <something>, (+2)
21             });
22     }
23
24     @Override
25     public boolean run(World world, Arguments args) {
26         // Find the entity we want to give in the room or our inventory.
27         Entity item = this.findEntity(world, Command.FILTER_ALL, args, "item", "<commands.give.nothing_specified>");
28         if (item == null) return false;
29
30         // Explicitly deny the player being given to anything, otherwise we will end up in an inventory.
31         var io = world.getIO();
32         if (item instanceof EntityPlayer) {
33             io.println("<commands.give.denied_player>");
34             return false;
35         }
36
37         // Find the target to give to.
38         Entity target = this.findEntity(world, args, "<commands.give.no_target>");
39         if (target == null) return false;
40
41         // If the target entity is an IGiveable, check if they accept this item.
```



```
42     if (target instanceof IGivable) {
43         ((IGivable) target).give(item);
44     } else {
45         io.println("<commands.give.denied.1> " + item.getHighlightedName()
46             + " <commands.give.denied.2> " + target.getHighlightedName() + "!");
47     }
48
49     return false;
50 }
51 }
```

```
1  package uk.insrt.coursework.zuul.commands.core;
2
3  import java.util.regex.Pattern;
4
5  import uk.insrt.coursework.zuul.commands.Arguments;
6  import uk.insrt.coursework.zuul.commands.Command;
7  import uk.insrt.coursework.zuul.world.Direction;
8  import uk.insrt.coursework.zuul.world.World;
9
10 /**
11  * Command which allows the Player to walk in a particular Direction.
12  */
13 public class CommandGo extends Command {
14     public CommandGo() {
15         super("go <selectors.direction>", "<commands.go.usage>",
16             new Pattern[] {
17                 Pattern.compile("^(?:go|walk)(?:\\s+(?<direction>[\\w\\s]+))*")
18                 // go, walk, go <direction>, walk <direction>
19             });
20     }
21
22     @Override
23     public boolean run(World world, Arguments arguments) {
24         Direction direction = arguments.direction();
25         if (direction == null) {
26             world.getIO().println("<commands.go.nothing_specified>");
27             return false;
28         }
29
30         world.getPlayer().go(direction);
31         return false;
32     }
33 }
```

```
1  package uk.insrt.coursework.zuul.commands.core;
2
3  import java.util.regex.Pattern;
4  import java.util.stream.Collectors;
5
6  import uk.insrt.coursework.zuul.commands.Arguments;
7  import uk.insrt.coursework.zuul.commands.Command;
8  import uk.insrt.coursework.zuul.commands.CommandManager;
9  import uk.insrt.coursework.zuul.io.Ansi;
10 import uk.insrt.coursework.zuul.world.World;
11
12 /**
13  * Command which allows the player to list all the available commands.
14  */
15 public class CommandHelp extends Command {
16     private CommandManager commandManager;
17
18     public CommandHelp(CommandManager commandManager) {
19         super("help", "<commands.help.usage>",
20             new Pattern[] {
21                 Pattern.compile("(?:h(?:elp)*) (?!\w)")
22                 // h, help
23             });
24
25         this.commandManager = commandManager;
26     }
27
28     @Override
29     public boolean run(World world, Arguments arguments) {
30         // Describe all the commands the player can run.
31         world.getIO()
32             .println(
33                 "<commands.help.can_run>\n" +
34                 this.commandManager
35                     .getCommands()
36                     .stream()
37                     .filter(Command::isVisible)
38                     .map(c -> "- " + Ansi.BackgroundWhite + Ansi.Black
39                         + c.getSyntax() + Ansi.Reset + ": " + c.getUsage())
40                     .collect(Collectors.joining("\n"))
41             );
42     }
43 }
```

```
42
43     return false;
44 }
45
46 @Override
47 public boolean isVisible() {
48     return false;
49 }
50 }
```

```
1  package uk.insrt.coursework.zuul.commands.core;
2
3  import java.util.regex.Pattern;
4
5  import uk.insrt.coursework.zuul.commands.Arguments;
6  import uk.insrt.coursework.zuul.commands.Command;
7  import uk.insrt.coursework.zuul.entities.Entity;
8  import uk.insrt.coursework.zuul.entities.actions.IPettable;
9  import uk.insrt.coursework.zuul.world.World;
10
11  /**
12   * Command which allows the player to pet another entity.
13   */
14  public class CommandPet extends Command {
15      public CommandPet() {
16          super("pet <selectors.something>", "<commands.pet.usage>",
17              new Pattern[] {
18                  Pattern.compile("^pet(?:\\s+(?<entity>[\\w\\s]+))*")
19                  // pet, pet <something>
20              });
21      }
22
23      @Override
24      public boolean run(World world, Arguments args) {
25          // Scan the room for entities that have IPettable.
26          Entity entity = this.findEntity(world, Command.FILTER_ALL, args, "<commands.pet.nothing_specified>");
27          if (entity != null) {
28              if (entity instanceof IPettable) {
29                  ((IPettable) entity).pet();
30              } else {
31                  world.getIO().println("<commands.pet.denied> " + entity.getHighlightedName() + ".");
32              }
33          }
34
35          return false;
36      }
37  }
```

```
1  package uk.insrt.coursework.zuul.commands.core;
2
3  import java.util.regex.Pattern;
4
5  import uk.insrt.coursework.zuul.commands.Arguments;
6  import uk.insrt.coursework.zuul.commands.Command;
7  import uk.insrt.coursework.zuul.world.World;
8
9  /**
10   * Command which allows the player to quit the game.
11   */
12  public class CommandQuit extends Command {
13      public CommandQuit() {
14          super("quit", "<commands.quit>",
15              new Pattern[] {
16                  Pattern.compile("^quit|exit(?:!\\w)"),
17                  // quit
18              });
19      }
20
21      @Override
22      public boolean run(World world, Arguments arguments) {
23          // We return true from run() in order to tell the game loop to exit.
24          return true;
25      }
26  }
```

```
1 package uk.insrt.coursework.zuul.commands.core;
2
3 import java.util.regex.Pattern;
4
5 import uk.insrt.coursework.zuul.commands.Arguments;
6 import uk.insrt.coursework.zuul.commands.Command;
7 import uk.insrt.coursework.zuul.entities.Entity;
8 import uk.insrt.coursework.zuul.entities.Inventory;
9 import uk.insrt.coursework.zuul.io.IOSystem;
10 import uk.insrt.coursework.zuul.util.Search;
11 import uk.insrt.coursework.zuul.world.World;
12
13 /**
14  * Command which allows the Player to take an Entity and put it in their Inventory.
15  * They may also take these Entities from other Entity Inventories.
16  */
17 public class CommandTake extends Command {
18     public CommandTake() {
19         super("take <selectors.something> [from <selectors.someone>]", "<commands.take.usage>",
20             new Pattern[] {
21                 Pattern.compile("^take\\s+(?<entity>[\\w\\s]+)\\s+from\\s+(?<other>[\\w\\s]+)"),
22                 Pattern.compile("^take(?:\\s+(?<entity>[\\w\\s]+))*")
23             },
24             // take, take <item>, take <item> from <entity>
25         );
26     }
27
28     @Override
29     public boolean run(World world, Arguments args) {
30         IOSystem io = world.getIO();
31         Entity player = world.getPlayer();
32         Inventory target = player.getInventory();
33
34         // Detect if we are taking from another entity, in that case run different logic.
35         if (args.has("other")) {
36             String name = args.group("entity");
37             Entity other = this.findEntity(world, args, "other", "<commands.take.nothing_specified>");
38             if (other == null) return false;
39
40             Entity item = Search.findEntity(other.getInventory().getItems(), name, true);
41             if (item == null) {
42                 io.println(other.getHighlightedName() + " <commands.take.entity_does_not_have_entity> " + name + "!");
43             }
44         }
45     }
46 }
```



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

```
1 package uk.insrt.coursework.zuul.commands.core;
2
3 import java.util.regex.Pattern;
4
5 import uk.insrt.coursework.zuul.commands.Arguments;
6 import uk.insrt.coursework.zuul.commands.Command;
7 import uk.insrt.coursework.zuul.entities.Entity;
8 import uk.insrt.coursework.zuul.entities.actions.ITalkwith;
9 import uk.insrt.coursework.zuul.world.World;
10
11 /**
12  * Command which allows the Player to talk with other Entities.
13  */
14 public class CommandTalk extends Command {
15     public CommandTalk() {
16         super("talk with <selectors.someone>", "<commands.talk.usage>",
17             new Pattern[] {
18                 Pattern.compile("^talk(?:(?:\\s*with|to)*(?:\\s+(?<entity>[\\w\\s]+))*$")
19                 // talk, talk with <entity>, talk to <entity>
20             });
21     }
22
23     @Override
24     public boolean run(World world, Arguments args) {
25         Entity entity = this.findEntity(world, args, "<commands.talk.nothing_specified>");
26         if (entity != null) {
27             if (entity instanceof ITalkwith) {
28                 ((ITalkwith) entity).talk();
29             } else {
30                 world.getIO().println("<commands.talk.denied> " + entity.getHighlightedName() + ".");
31             }
32         }
33
34         return false;
35     }
36 }
```

```
1  package uk.insrt.coursework.zuul.commands.core;
2
3  import java.util.regex.Pattern;
4
5  import uk.insrt.coursework.zuul.commands.Arguments;
6  import uk.insrt.coursework.zuul.commands.Command;
7  import uk.insrt.coursework.zuul.entities.Entity;
8  import uk.insrt.coursework.zuul.entities.actions.IUseable;
9  import uk.insrt.coursework.zuul.world.World;
10
11  /**
12   * Command which allows the Player to use an Entity.
13   */
14  public class CommandUse extends Command {
15      public CommandUse() {
16          super("use <selectors.something>", "<commands.use.usage>",
17              new Pattern[] {
18                  Pattern.compile("^use(?:\\s+(?<entity>[\\w\\s]+))*")
19                  // use, use <entity>
20              });
21      }
22
23      @Override
24      public boolean run(World world, Arguments args) {
25          Entity entity = this.findEntity(world, Command.FILTER_ALL, args, "<commands.use.nothing_specified>");
26          if (entity != null) {
27              if (entity instanceof IUseable) {
28                  ((IUseable) entity).use(world.getPlayer());
29              } else {
30                  world.getIO().println("<commands.use.denied> " + entity.getHighlightedName() + ".");
31              }
32          }
33
34          return false;
35      }
36  }
```

```
1  package uk.insrt.coursework.zuul.commands.core;
2
3  import java.util.regex.Pattern;
4
5  import uk.insrt.coursework.zuul.commands.Arguments;
6  import uk.insrt.coursework.zuul.commands.Command;
7  import uk.insrt.coursework.zuul.events.world.EventEntityEnteredRoom;
8  import uk.insrt.coursework.zuul.world.World;
9
10 /**
11  * Command which allows the player to reorient themselves in the world.
12  */
13 public class CommandWhereAmI extends Command {
14     public CommandWhereAmI() {
15         super("where am i", "<commands.where_am_i>",
16             new Pattern[] {
17                 Pattern.compile("^where(\\s+am\\s+(i(?:!\\w)*)*)"),
18                 // where am i
19             });
20     }
21
22     @Override
23     public boolean run(World world, Arguments arguments) {
24         // We can just re-emit the enter room event to
25         // trigger the room description logic to run again.
26         world.emit(new EventEntityEnteredRoom(world.getPlayer()));
27         return false;
28     }
29 }
```

```
1  package uk.insrt.coursework.zuul.content.campaign;
2
3  import java.io.IOException;
4  import java.util.HashSet;
5  import java.util.stream.Collectors;
6
7  import uk.insrt.coursework.zuul.content.campaign.StoryFlags.Stage;
8  import uk.insrt.coursework.zuul.content.campaign.entities.EntityCat;
9  import uk.insrt.coursework.zuul.content.campaign.entities.EntityWithDialogue;
10 import uk.insrt.coursework.zuul.content.campaign.events.EventGameStageChanged;
11 import uk.insrt.coursework.zuul.content.campaign.rooms.RoomApartmentsHome;
12 import uk.insrt.coursework.zuul.content.campaign.rooms.RoomApartmentsReception;
13 import uk.insrt.coursework.zuul.content.campaign.rooms.RoomBackAlley;
14 import uk.insrt.coursework.zuul.content.campaign.rooms.RoomCityCentre;
15 import uk.insrt.coursework.zuul.content.campaign.rooms.RoomCoastline;
16 import uk.insrt.coursework.zuul.content.campaign.rooms.RoomForest;
17 import uk.insrt.coursework.zuul.content.campaign.rooms.RoomMainlandCoastline;
18 import uk.insrt.coursework.zuul.content.campaign.rooms.RoomMedicalCentreOffice;
19 import uk.insrt.coursework.zuul.content.campaign.rooms.RoomMedicalCentreReception;
20 import uk.insrt.coursework.zuul.content.campaign.rooms.RoomShop;
21 import uk.insrt.coursework.zuul.content.campaign.rooms.RoomStreet;
22 import uk.insrt.coursework.zuul.content.campaign.rooms.RoomWormHole;
23 import uk.insrt.coursework.zuul.dialogue.DialogueLoader;
24 import uk.insrt.coursework.zuul.entities.Entity;
25 import uk.insrt.coursework.zuul.entities.EntityPlayer;
26 import uk.insrt.coursework.zuul.events.IEventListener;
27 import uk.insrt.coursework.zuul.events.world.EventEntityEnteredRoom;
28 import uk.insrt.coursework.zuul.events.world.EventEntityLeftRoom;
29 import uk.insrt.coursework.zuul.io.Ansi;
30 import uk.insrt.coursework.zuul.io.IOSystem;
31 import uk.insrt.coursework.zuul.world.Room;
32 import uk.insrt.coursework.zuul.world.World;
33
34 /**
35  * The main campaign World.
36  */
37 public class CampaignWorld extends World {
38     private StoryFlags flags;
39     private HashSet<Room> visitedRooms;
40     private DialogueLoader dialogueLoader;
41 }
```

```
42  /**
43   * Construct a new Campaign World
44   * @param io Provided IO system
45   */
46  public CampaignWorld(IOSystem io) {
47      super(io);
48
49      this.visitedRooms = new HashSet<>();
50      this.dialogueLoader = new DialogueLoader();
51      this.flags = new StoryFlags(this.getEventSystem());
52
53      try {
54          this.dialogueLoader.load("/dialogue.toml");
55      } catch (IOException e) {
56          System.err.println("Failed to load resources for campaign world!");
57          e.printStackTrace();
58      }
59
60      this.buildWorld();
61      this.spawnEntities();
62      this.registerEvents();
63  }
64
65  /**
66   * Get this World's Dialogue Loader
67   * @return Dialogue Loader
68   */
69  public DialogueLoader getDialogueLoader() {
70      return this.dialogueLoader;
71  }
72
73  /**
74   * Get the global story flags.
75   * @return Story flags instance
76   */
77  public StoryFlags getStoryFlags() {
78      return this.flags;
79  }
80
81  /**
82   * Check whether the Player has visited a certain Room yet
```

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



```

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

```

```
165
166     // Mark current room as previously visited.
167     this.visitedRooms.add(room);
168
169     // When we enter a new room, list what we can see.
170     String entities = this.getEntitiesInRoom(entity.getRoom())
171         .stream()
172         .filter(e -> !(e instanceof EntityPlayer))
173         .map(e -> "-" + e.describe() + " ("
174             + Ansi.BackgroundPurple + Ansi.Black
175             + e.getName() + Ansi.Reset + ")")
176         .collect(Collectors.joining("\n"));
177
178     if (entities.length() > 0) {
179         this.io.println("<global.sight>\n" + entities);
180     }
181 } else {
182     // If another entity enters the room,
183     // conditionally mention this to the player.
184     EntityPlayer player = this.getPlayer();
185     if (entity.getRoom() == player.getRoom()) {
186         if (entity instanceof EntityCat) {
187             this.io.println("\n<entities.cat.enter>");
188         }
189     }
190 }
191 });
192
193 // Capture all Events for Entities leaving Rooms.
194 this.eventSystem.addListener(EventEntityLeftRoom.class,
195     (EventEntityLeftRoom event) -> {
196         Entity entity = event.getEntity();
197         if (entity instanceof EntityPlayer) return;
198
199         Room room = event.getRoom();
200         if (room != this.player.getRoom()) return;
201
202         // If another entity leaves the room,
203         // conditionally mention this to the player.
204         if (entity instanceof EntityCat) {
205             this.io.println("\n<entities.cat.leave>");
206         }
207     }
208 );
```

```

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

```

```
1  package uk.insrt.coursework.zuul.content.campaign.commands;
2
3  import java.awt.Image;
4  import java.io.InputStream;
5  import java.util.regex.Pattern;
6
7  import javax.imageio.ImageIO;
8
9  import uk.insrt.coursework.zuul.commands.Arguments;
10 import uk.insrt.coursework.zuul.commands.Command;
11 import uk.insrt.coursework.zuul.content.campaign.CampaignWorld;
12 import uk.insrt.coursework.zuul.events.IEventListener;
13 import uk.insrt.coursework.zuul.io.IOSystem;
14 import uk.insrt.coursework.zuul.ui.EventDraw;
15 import uk.insrt.coursework.zuul.world.World;
16
17 /**
18  * Command available for the terminal emulator which displays a graphical map.
19  */
20 public class CommandMap extends Command implements IEventListener<EventDraw> {
21     private boolean visible;
22     private Image image;
23
24     public CommandMap() {
25         super("map", "<commands.map.usage>",
26             new Pattern[] {
27                 Pattern.compile("(?:m(?:ap)*) (?!\w)")
28                 // m, map
29             });
30
31         this.visible = false;
32
33         try {
34             InputStream stream = this.getClass().getResourceAsStream("/map/base.png");
35             this.image = ImageIO.read(stream);
36         } catch (Exception e) {}
37     }
38
39     @Override
40     public boolean run(World world, Arguments arguments) {
41         CampaignWorld campaignWorld = (CampaignWorld) world;
```

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

```
1  package uk.insrt.coursework.zuul.content.campaign.commands;
2
3  import java.util.regex.Pattern;
4
5  import uk.insrt.coursework.zuul.commands.Arguments;
6  import uk.insrt.coursework.zuul.commands.Command;
7  import uk.insrt.coursework.zuul.content.campaign.CampaignWorld;
8  import uk.insrt.coursework.zuul.content.campaign.StoryFlags.Stage;
9  import uk.insrt.coursework.zuul.world.World;
10
11  /**
12   * This command is unlocked after the player completes the final mission.
13   */
14  public class CommandWin extends Command {
15      public CommandWin() {
16          super("win", "<commands.win.usage>",
17              new Pattern[] {
18                  Pattern.compile("^win(?:!\\w)"),
19                  // win
20              });
21      }
22
23      @Override
24      public boolean run(World world, Arguments args) {
25          var io = world.getIO();
26          var w = (CampaignWorld) world;
27          var flags = w.getStoryFlags();
28          if (flags.getStage() != Stage.End) return false;
29
30          io.println("<commands.win.conclusion>\n<commands.win.stats>\n"
31              + "<commands.win.total_ticks>" + flags.getTicks() + "\n"
32              + "<commands.win.sidequests_complete>"
33              + flags.getCompletedQuests() + " / " + flags.getTotalQuests()
34              + "\n\n<commands.win.press_enter_key>");
35
36          io.readLine();
37          return true;
38      }
39
40      @Override
41      public boolean isVisible() {
```

```
42     return false;  
43 }  
44 }
```

```
1  package uk.insrt.coursework.zuul.content.campaign.entities;
2
3  import uk.insrt.coursework.zuul.entities.Entity;
4  import uk.insrt.coursework.zuul.entities.EntityObject;
5  import uk.insrt.coursework.zuul.entities.actions.IUseable;
6  import uk.insrt.coursework.zuul.events.world.EventTick;
7  import uk.insrt.coursework.zuul.world.Location;
8  import uk.insrt.coursework.zuul.world.World;
9
10 /**
11  * Bed entity which lets the player tick the World forwards.
12  */
13 public class EntityBed extends EntityObject implements IUseable {
14     public EntityBed(World world, Location location) {
15         super(world, location, 80, new String[] { "bed" }, "<entities.bed.description>");
16     }
17
18     public void use(Entity target) {
19         // We emit EventTick an arbitrary amount of times to
20         // in-effect push the time forwards. This will trigger
21         // all random events which listen to this event.
22         for (int i=0;i<20;i++) {
23             world.emit(new EventTick());
24         }
25
26         this.world.getIO().println("<entities.bed.use>");
27     }
28 }
```



```
1  package uk.insrt.coursework.zuul.content.campaign.entities;
2
3  import uk.insrt.coursework.zuul.content.campaign.CampaignWorld;
4  import uk.insrt.coursework.zuul.content.campaign.StoryFlags.Stage;
5  import uk.insrt.coursework.zuul.entities.Entity;
6  import uk.insrt.coursework.zuul.entities.Inventory;
7  import uk.insrt.coursework.zuul.entities.actions.IGiveable;
8  import uk.insrt.coursework.zuul.entities.actions.IUseable;
9  import uk.insrt.coursework.zuul.io.IOSystem;
10 import uk.insrt.coursework.zuul.world.Location;
11 import uk.insrt.coursework.zuul.world.Room;
12 import uk.insrt.coursework.zuul.world.World;
13
14 /**
15  * Boat entity which ferries the Player to an arbitrary location.
16  * There is no restriction on location but they should be place as
17  * appropriate and where it would be realistic to put a boat.
18  *
19  * Boats may not be operated by the player while they are carrying
20  * anything so instead they must use the boat's storage.
21  */
22 public class EntityBoat extends Entity implements IUseable, IGiveable {
23     private Room destination;
24
25     public EntityBoat(World world, Location location, Room destination) {
26         super(world, location, 200);
27         this.destination = destination;
28         this.inventory.setMaxWeight(100);
29     }
30
31     @Override
32     public String[] getAliases() {
33         return new String[] { "boat" };
34     }
35
36     @Override
37     public String describe() {
38         return "<entities.boat.description>";
39     }
40
41     @Override
```

```
42 public void use(Entity target) {
43     CampaignWorld world = (CampaignWorld) this.getWorld();
44     IOSystem io = world.getIO();
45     Inventory inventory = target.getInventory();
46
47     // Check if the player has the key to this boat.
48     boolean hasKey = false;
49     for (Entity item : inventory.getItems()) {
50         if (item instanceof EntityBoatKey) {
51             hasKey = true;
52         }
53     }
54
55     if (!hasKey) {
56         if (world.getStoryFlags().getStage() == Stage.Exposition) {
57             io.println("<entities.boat.locked>");
58         } else {
59             io.println("<entities.boat.locked_for_sale>");
60         }
61
62         return;
63     }
64
65     // Check whether the player is carrying too much.
66     if (inventory.getWeight() > 1) {
67         io.println("<entities.boat.denied>");
68         return;
69     }
70
71     // If we're good to go, travel to the other side.
72     io.println("<entities.boat.travel>\n");
73     target.setLocation(this.destination);
74 }
75
76 @Override
77 public void give(Entity item) {
78     var io = this.getWorld().getIO();
79     if (item.setLocation(this.getInventory())) {
80         io.println("<entities.boat.give.1> " + item.getHighlightedName() + " <entities.boat.give.2>.");
81     } else {
82         io.println("<entities.boat.too_heavy>");
```

```
83     }  
84 }  
85 }
```

```
1 package uk.insrt.coursework.zuul.content.campaign.entities;
2
3 import uk.insrt.coursework.zuul.entities.EntityObject;
4 import uk.insrt.coursework.zuul.world.Location;
5 import uk.insrt.coursework.zuul.world.World;
6
7 /**
8  * Boat key object which is used to unlock and start the
9  * speed boat present on the coast.
10 */
11 public class EntityBoatKey extends EntityObject {
12     public EntityBoatKey(World world, Location location) {
13         super(world, location, 0.01d,
14             new String[] { "key" },
15             "<entities.boat_key>");
16     }
17 }
```

```
1  package uk.insrt.coursework.zuul.content.campaign.entities;
2
3  import uk.insrt.coursework.zuul.entities.Entity;
4  import uk.insrt.coursework.zuul.entities.actions.IPettable;
5  import uk.insrt.coursework.zuul.entities.actions.IUseable;
6  import uk.insrt.coursework.zuul.events.world.EventTick;
7  import uk.insrt.coursework.zuul.events.world.behaviours.SimpleWanderAI;
8  import uk.insrt.coursework.zuul.world.Location;
9  import uk.insrt.coursework.zuul.world.Room;
10 import uk.insrt.coursework.zuul.world.World;
11
12 /**
13  * Cat entity which wanders around the map.
14  */
15 public class EntityCat extends Entity implements IPettable, IUseable {
16     public static final int WEIGHT = 5;
17
18     public EntityCat(World world, Location startingLocation) {
19         super(world, startingLocation, WEIGHT);
20     }
21
22     @Override
23     public String[] getAliases() {
24         return new String[] {
25             "cat",
26             "the cat"
27         };
28     }
29
30     @Override
31     public String describe() {
32         return "<entities.cat.description>";
33     }
34
35     @Override
36     public void pet() {
37         this.getWorld().getIO().println("<entities.cat.pet>");
38     }
39
40     @Override
41     public void use(Entity target) {
```

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

```
1  package uk.insrt.coursework.zuul.content.campaign.entities;
2
3  import uk.insrt.coursework.zuul.content.campaign.CampaignWorld;
4  import uk.insrt.coursework.zuul.content.campaign.StoryFlags.Stage;
5  import uk.insrt.coursework.zuul.content.campaign.rooms.RoomMedicalCentreOffice;
6  import uk.insrt.coursework.zuul.content.campaign.rooms.RoomMedicalCentreReception;
7  import uk.insrt.coursework.zuul.dialogue.Dialogue;
8  import uk.insrt.coursework.zuul.dialogue.DialogueOption;
9  import uk.insrt.coursework.zuul.entities.Entity;
10 import uk.insrt.coursework.zuul.entities.actions.IUseable;
11 import uk.insrt.coursework.zuul.world.Location;
12 import uk.insrt.coursework.zuul.world.Room;
13 import uk.insrt.coursework.zuul.world.World;
14
15 /**
16  * Comms entity which is used to communicate between
17  * Marie and the player during the mission.
18  */
19 public class EntityComms extends EntityWithDialogue<String> implements IUseable {
20     public EntityComms(World world, Location location) {
21         super(world, location, 0.3d);
22         this.setupDialogue();
23     }
24
25     @Override
26     public void use(Entity target) {
27         var w = (CampaignWorld) this.world;
28         var io = w.getIO();
29
30         if (w.getStoryFlags().getStage() == Stage.Stealth) {
31             Room room = w.getPlayer().getRoom();
32             if (room instanceof RoomMedicalCentreOffice) {
33                 this.dialogue.setNodeIfPresent("office");
34             } else if (room instanceof RoomMedicalCentreReception) {
35                 if (((RoomMedicalCentreReception) room).getCouch().isSitting()) {
36                     this.dialogue.setNodeIfPresent("in_position");
37                 } else {
38                     this.dialogue.setNodeIfPresent("complaint");
39                 }
40             } else {
41                 this.dialogue.setNodeIfPresent("orientation");
42             }
43         }
44     }
45 }
```

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



```
1  package uk.insrt.coursework.zuul.content.campaign.entities;
2
3  import uk.insrt.coursework.zuul.entities.Entity;
4  import uk.insrt.coursework.zuul.entities.EntityObject;
5  import uk.insrt.coursework.zuul.entities.actions.IUseable;
6  import uk.insrt.coursework.zuul.events.IEventListener;
7  import uk.insrt.coursework.zuul.events.world.EventEntityLeftRoom;
8  import uk.insrt.coursework.zuul.world.Location;
9  import uk.insrt.coursework.zuul.world.World;
10
11  /**
12   * Couch present in the Medical Centre reception area.
13   */
14  public class EntityCouch extends EntityObject implements IUseable, IEventListener<EventEntityLeftRoom> {
15      private boolean isSitting;
16
17      /**
18       * Construct a new EntityCouch.
19       * @param world World
20       * @param location Location
21       */
22      public EntityCouch(World world, Location location) {
23          super(world, location, Double.MAX_VALUE, "couch", "<entities.couch.description>");
24      }
25
26      @Override
27      public void use(Entity target) {
28          var io = this.getWorld().getIO();
29          if (this.isSitting) {
30              io.println("<entities.couch.sitting>");
31          } else {
32              io.println("<entities.couch.sit>");
33              this.isSitting = true;
34          }
35      }
36
37      @Override
38      public void onEvent(EventEntityLeftRoom event) {
39          this.isSitting = false;
40      }
41  }
```

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

```
1  package uk.insrt.coursework.zuul.content.campaign.entities;
2
3  import uk.insrt.coursework.zuul.entities.Entity;
4  import uk.insrt.coursework.zuul.entities.EntityObject;
5  import uk.insrt.coursework.zuul.entities.actions.IUseable;
6  import uk.insrt.coursework.zuul.world.Location;
7  import uk.insrt.coursework.zuul.world.World;
8
9  /**
10   * Documents which the player needs to find and take.
11   */
12  public class EntityDocument extends EntityObject implements IUseable {
13      private int count;
14
15      /**
16       * Construct a new EntityDocument
17       * @param world World
18       * @param location Location
19       * @param count Document Id
20       */
21      public EntityDocument(World world, Location location, int count) {
22          super(world, location, 10, "doc" + count, "<medical_centre_office.books." + count + ".title>");
23          this.count = count;
24      }
25
26      @Override
27      public void use(Entity target) {
28          this.getWorld()
29              .getIO()
30              .println("<medical_centre_office.books." + count + ".contents>");
31      }
32
33      /**
34       * Check whether these are the documents we want.
35       * @return Whether we want this document
36       */
37      public boolean getIsValid() {
38          switch (this.count) {
39              case 2:
40              case 4:
41              case 5: return true;
```

```
42         }
43
44         return false;
45     }
46 }
```

```
1  package uk.insrt.coursework.zuul.content.campaign.entities;
2
3  import java.awt.Desktop;
4  import java.net.URI;
5
6  import uk.insrt.coursework.zuul.content.campaign.CampaignWorld;
7  import uk.insrt.coursework.zuul.content.campaign.StoryFlags.Stage;
8  import uk.insrt.coursework.zuul.dialogue.Dialogue;
9  import uk.insrt.coursework.zuul.dialogue.DialogueNode;
10 import uk.insrt.coursework.zuul.dialogue.DialogueOption;
11 import uk.insrt.coursework.zuul.entities.Entity;
12 import uk.insrt.coursework.zuul.entities.Inventory;
13 import uk.insrt.coursework.zuul.entities.actions.IUseable;
14 import uk.insrt.coursework.zuul.world.Location;
15 import uk.insrt.coursework.zuul.world.World;
16
17 /**
18  * This is the player's laptop which resides in their home.
19  */
20 public class EntityLaptop extends EntityWithDialogue<String> implements IUseable {
21     /**
22      * Construct a new EntityLaptop.
23      * @param world World
24      * @param location Location
25      */
26     public EntityLaptop(World world, Location location) {
27         super(world, location, 2);
28         this.setupDialogue();
29     }
30
31     @Override
32     public void use(Entity target) {
33         this.dialogue.run(this.getWorld().getIO());
34     }
35
36     @Override
37     public void setupDialogue(Dialogue<String> dialogue) {
38         this.setupDialogueFromId(dialogue, "entity_laptop");
39
40         // Add funny cat videos.
41         this.dialogue.addPart("funny_cat_videos",
```

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

```
83
84     @Override
85     public String[] getAliases() {
86         return new String[] { "laptop" };
87     }
88
89     @Override
90     public String describe() {
91         return "<entities.laptop.description>";
92     }
93 }
```

```
1  package uk.insrt.coursework.zuul.content.campaign.entities;
2
3  import uk.insrt.coursework.zuul.content.campaign.CampaignWorld;
4  import uk.insrt.coursework.zuul.content.campaign.StoryFlags.Stage;
5  import uk.insrt.coursework.zuul.dialogue.Dialogue;
6  import uk.insrt.coursework.zuul.dialogue.DialogueOption;
7  import uk.insrt.coursework.zuul.entities.Entity;
8  import uk.insrt.coursework.zuul.entities.Inventory;
9  import uk.insrt.coursework.zuul.world.Location;
10 import uk.insrt.coursework.zuul.world.World;
11
12 /**
13  * Marie Itami
14  * https://brand-new-animal.fandom.com/wiki/Marie\_Itami
15  */
16 public class EntityMarie extends EntityNPC {
17     /**
18      * Construct a new EntityMarie.
19      * @param world World
20      * @param location Location
21      */
22     public EntityMarie(World world, Location location) {
23         super(world, location,
24             "npc_marie",
25             "<marie.description>",
26             new String[] { "marie", "itami", "mink" });
27     }
28
29     @Override
30     public void setupDialogue(Dialogue<String> dialogue) {
31         super.setupDialogue(dialogue);
32         var w = (CampaignWorld) this.world;
33
34         // Progress story if player accepts mission.
35         dialogue.getPart("confirm")
36             .addOption(new DialogueOption<>("<marie.alley.confirm.option_1>",
37                 io -> {
38                     w.getStoryFlags()
39                     .setStage(Stage.Recon);
40
41                     return "recon";
```



```
42         }));
43     }
44
45     @Override
46     public void talk() {
47         if (this.dialogue.getCurrentNode().equals("waiting")) {
48             var w = ((CampaignWorld) this.getWorld());
49             Inventory inv = w.getPlayer().getInventory();
50             for (Entity item : inv.getItems()) {
51                 if (item instanceof EntityComms) {
52                     this.dialogue.setNodeIfPresent("mission_brief");
53                     w.getStoryFlags().setStage(Stage.Stealth);
54                 }
55             }
56         }
57
58         super.talk();
59     }
60 }
```

```
1  package uk.insrt.coursework.zuul.content.campaign.entities;
2
3  import uk.insrt.coursework.zuul.dialogue.Dialogue;
4  import uk.insrt.coursework.zuul.entities.actions.ITalkwith;
5  import uk.insrt.coursework.zuul.world.Location;
6  import uk.insrt.coursework.zuul.world.World;
7
8  /**
9   * NPC entity which provides dialog and can be talked with by the Player.
10  */
11  public class EntityNPC extends EntityWithDialogue<String> implements ITalkwith {
12      private String description;
13      private String alias[];
14      private String id;
15
16      public EntityNPC(World world, Location location, String id, String description, String alias[]) {
17          super(world, location, 75, null);
18
19          this.description = description;
20          this.alias = alias;
21          this.id = id;
22
23          this.setupDialogue();
24      }
25
26      public void talk() {
27          this.dialogue.run(this.getWorld().getIO());
28      }
29
30      @Override
31      public String[] getAliases() {
32          return this.alias;
33      }
34
35      @Override
36      public String describe() {
37          return this.description;
38      }
39
40      @Override
41      public void setupDialogue(Dialogue<String> dialogue) {
```

```
42     this.setupDialogueFromId(dialogue, id);
43 }
44 }
```

```
1  package uk.insrt.coursework.zuul.content.campaign.entities;
2
3  import uk.insrt.coursework.zuul.content.campaign.CampaignWorld;
4  import uk.insrt.coursework.zuul.content.campaign.StoryFlags.Quest;
5  import uk.insrt.coursework.zuul.entities.Entity;
6  import uk.insrt.coursework.zuul.entities.actions.IGiveable;
7  import uk.insrt.coursework.zuul.world.Location;
8  import uk.insrt.coursework.zuul.world.World;
9
10 /**
11  * The old man who is in the forest.
12  */
13 public class EntityOldMan extends EntityNPC implements IGiveable {
14     /**
15      * Construct a new EntityOldMan.
16      * @param world World
17      * @param location Location
18      */
19     public EntityOldMan(World world, Location location) {
20         super(world, location, "npc_old_man",
21             "<forest.old_man.description>",
22             new String[] { "oldman", "man" });
23
24         this.inventory.setMaxWeight(EntityCat.WEIGHT);
25     }
26
27     @Override
28     public void give(Entity item) {
29         var io = this.getWorld().getIO();
30         if (this.inventory.isFull()) {
31             io.println("<forest.old_man.full>");
32             return;
33         }
34
35         if (item instanceof EntityCat) {
36             item.setLocation(this.inventory);
37             this.dialogue.setNodeIfPresent("praise");
38             ((CampaignWorld) this.getWorld())
39                 .getStoryFlags()
40                 .completeSideQuest(Quest.Cat);
41         }
42     }
43 }
```

```
42         io.println("<forest.old_man.accept>");
43     } else {
44         io.println("<forest.old_man.deny> " + item.getHighlightedName());
45     }
46 }
47 }
```

```
1  package uk.insrt.coursework.zuul.content.campaign.entities;
2
3  import java.util.ArrayList;
4  import java.util.HashMap;
5  import java.util.List;
6
7  import uk.insrt.coursework.zuul.content.campaign.CampaignWorld;
8  import uk.insrt.coursework.zuul.content.campaign.StoryFlags.Stage;
9  import uk.insrt.coursework.zuul.dialogue.Dialogue;
10 import uk.insrt.coursework.zuul.dialogue.DialogueNode;
11 import uk.insrt.coursework.zuul.dialogue.DialogueOption;
12 import uk.insrt.coursework.zuul.entities.Entity;
13 import uk.insrt.coursework.zuul.entities.EntityObject;
14 import uk.insrt.coursework.zuul.entities.Inventory;
15 import uk.insrt.coursework.zuul.io.Ansi;
16 import uk.insrt.coursework.zuul.world.Location;
17 import uk.insrt.coursework.zuul.world.World;
18
19 /**
20  * Shop keeper which the player can buy items from in the town.
21  */
22 public class EntityShopkeeper extends EntityNPC {
23     private HashMap<Stage, Entity[]> items;
24     private HashMap<Entity, Integer> stock;
25     private HashMap<Entity, Integer> price;
26     private HashMap<Entity, IEntityFactory> entityFactory;
27
28     /**
29      * Construct a new EntityShopkeeper.
30      * @param world World
31      * @param location Location
32      */
33     public EntityShopkeeper(World world, Location location) {
34         super(world, location,
35             "npc_shopkeeper",
36             "<shop.npc.description>",
37             new String[] { "shopkeeper", "shop", "keeper" });
38
39         this.items = new HashMap<>();
40         this.stock = new HashMap<>();
41         this.price = new HashMap<>();
```

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

```

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

```



```

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

```

```
165         + item.getHighlightedName() + "!");
166     }
167 }
168 }
169
170     return "index";
171 }
172 ));
173 }
174
175     options.add(new DialogueOption<String>("<shop.npc.leave>", "index").mustExit());
176     return options;
177 }
178 }
179
180 @Override
181 public void setupDialogue(Dialogue<String> dialogue) {
182     dialogue.addPart("index", new IndexNode());
183     dialogue.setNodeIfPresent("index");
184 }
185 }
```

```
1  package uk.insrt.coursework.zuul.content.campaign.entities;
2
3  import uk.insrt.coursework.zuul.dialogue.Dialogue;
4  import uk.insrt.coursework.zuul.entities.Entity;
5  import uk.insrt.coursework.zuul.entities.actions.IUseable;
6  import uk.insrt.coursework.zuul.world.Location;
7  import uk.insrt.coursework.zuul.world.World;
8
9  /**
10   * TV entity present in the player's room which they interact
11   * with at the start of the game to learn more about the world.
12   */
13  public class EntityTV extends EntityWithDialogue<String> implements IUseable {
14      public EntityTV(World world, Location location) {
15          super(world, location, 40);
16          this.setupDialogue();
17      }
18
19      @Override
20      public void use(Entity target) {
21          this.dialogue.run(this.getWorld().getIO());
22      }
23
24      @Override
25      public void setupDialogue(Dialogue<String> dialogue) {
26          this.setupDialogueFromId(dialogue, "home_tv");
27      }
28
29      @Override
30      public String[] getAliases() {
31          return new String[] { "tv", "television" };
32      }
33
34      @Override
35      public String describe() {
36          return "<home.tv.description>";
37      }
38  }
```

```
1  package uk.insrt.coursework.zuul.content.campaign.entities;
2
3  import uk.insrt.coursework.zuul.content.campaign.CampaignWorld;
4  import uk.insrt.coursework.zuul.dialogue.Dialogue;
5  import uk.insrt.coursework.zuul.entities.Entity;
6  import uk.insrt.coursework.zuul.world.Location;
7  import uk.insrt.coursework.zuul.world.World;
8
9  /**
10   * Abstract implementation of an entity which provides some sort of dialogue.
11   */
12  public abstract class EntityWithDialogue<T> extends Entity {
13      protected Dialogue<T> dialogue;
14
15      /**
16       * Construct a new EntityWithDialogue with a starting dialogue node
17       * @param world Current World object
18       * @param location Initial Location of this Entity
19       * @param weight The weight (in kg) of this Entity
20       * @param startNode The starting dialogue node
21       */
22      public EntityWithDialogue(World world, Location location, double weight, T startNode) {
23          super(world, location, weight);
24
25          Dialogue<T> dialogue = new Dialogue<T>(startNode);
26          this.dialogue = dialogue;
27      }
28
29      /**
30       * Construct a new EntityWithDialogue without a starting dialogue node
31       * @param world Current World object
32       * @param location Initial Location of this Entity
33       * @param weight The weight (in kg) of this Entity
34       */
35      public EntityWithDialogue(World world, Location location, double weight) {
36          this(world, location, weight, null);
37      }
38
39      /**
40       * Configure this Entity's dialogue,
41       * create nodes and options to add to this Entity.
```

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

```
1  package uk.insrt.coursework.zuul.content.campaign.events;
2
3  import uk.insrt.coursework.zuul.content.campaign.StoryFlags.Stage;
4  import uk.insrt.coursework.zuul.events.Event;
5
6  /**
7   * Event fired when the story stage (chapter) changes.
8   */
9  public class EventGameStageChanged extends Event {
10     private Stage stage;
11
12     /**
13      * Construct a new GameStageChanged event.
14      * @param stage New stage
15      */
16     public EventGameStageChanged(Stage stage) {
17         this.stage = stage;
18     }
19
20     /**
21      * Get the new game stage.
22      * @return Stage
23      */
24     public Stage getStage() {
25         return this.stage;
26     }
27 }
```

```
1  package uk.insrt.coursework.zuul.content.campaign.rooms;
2
3  import uk.insrt.coursework.zuul.content.campaign.CampaignWorld;
4  import uk.insrt.coursework.zuul.world.Room;
5  import uk.insrt.coursework.zuul.world.World;
6
7  /**
8   * Class which overrides getWorld to instead provide the CampaignWorld.
9   */
10 public abstract class CampaignRoom extends Room {
11     /**
12      * Construct a new CampaignRoom.
13      * @param world World
14      * @param name Internal name used to refer to this Room
15      */
16     public CampaignRoom(World world, String name) {
17         super(world, name);
18     }
19
20     @Override
21     public CampaignWorld getWorld() {
22         return (CampaignWorld) super.getWorld();
23     }
24 }
```

```
1  package uk.insrt.coursework.zuul.content.campaign.rooms;
2
3  import uk.insrt.coursework.zuul.content.campaign.entities.EntityBed;
4  import uk.insrt.coursework.zuul.content.campaign.entities.EntityLaptop;
5  import uk.insrt.coursework.zuul.content.campaign.entities.EntityTV;
6  import uk.insrt.coursework.zuul.world.Direction;
7  import uk.insrt.coursework.zuul.world.World;
8
9  /**
10   * The player's home in the apartments complex.
11   */
12  public class RoomApartmentsHome extends CampaignRoom {
13      public RoomApartmentsHome(World world) {
14          super(world, "Apartments: Home");
15      }
16
17      @Override
18      public String describe() {
19          var world = this.getWorld();
20          if (!world.hasVisited(this)) {
21              return "<home.first_load>";
22          }
23
24          return "<home.enter>";
25      }
26
27      @Override
28      protected void setupDirections() {
29          this.setAdjacent(Direction.DOWN, this.getWorld().getRoom("Apartments: Reception"));
30      }
31
32      @Override
33      public void spawnEntities() {
34          World world = this.getWorld();
35
36          world.spawnEntity("tv", new EntityTV(world, this.toLocation()));
37          world.spawnEntity("bed", new EntityBed(world, this.toLocation()));
38          world.spawnEntity("laptop", new EntityLaptop(world, this.toLocation()));
39      }
40  }
```



```
1  package uk.insrt.coursework.zuul.content.campaign.rooms;
2
3  import uk.insrt.coursework.zuul.content.campaign.entities.EntityNPC;
4  import uk.insrt.coursework.zuul.world.Direction;
5  import uk.insrt.coursework.zuul.world.World;
6
7  /**
8   * The reception of the apartments complex.
9   */
10 public class RoomApartmentsReception extends CampaignRoom {
11     public RoomApartmentsReception(World world) {
12         super(world, "Apartments: Reception");
13     }
14
15     @Override
16     public String describe() {
17         return "<apartments.enter>";
18     }
19
20     @Override
21     protected void setupDirections() {
22         World world = this.getWorld();
23         this.setAdjacent(Direction.NORTH, world.getRoom("Street"));
24         this.setAdjacent(Direction.EAST, world.getRoom("City Centre"));
25         this.setAdjacent(Direction.UP, world.getRoom("Apartments: Home"));
26     }
27
28     @Override
29     public void spawnEntities() {
30         World world = this.getWorld();
31         world.spawnEntity("receptionist",
32             new EntityNPC(
33                 world,
34                 this.toLocation(),
35                 "npc_receptionist",
36                 "<apartments.receptionist.description>",
37                 new String[] { "receptionist" }
38             ));
39     }
40 }
```

```
1  package uk.insrt.coursework.zuul.content.campaign.rooms;
2
3  import uk.insrt.coursework.zuul.content.campaign.entities.EntityMarie;
4  import uk.insrt.coursework.zuul.world.Direction;
5  import uk.insrt.coursework.zuul.world.World;
6
7  /**
8   * The back alley in the North East side of the map.
9   */
10 public class RoomBackAlley extends CampaignRoom {
11     public RoomBackAlley(World world) {
12         super(world, "Back Alley");
13     }
14
15     @Override
16     public String describe() {
17         var world = this.getWorld();
18         if (!world.hasVisited(this)) {
19             return "<back_alley.first_load>";
20         }
21
22         return "<back_alley.enter>";
23     }
24
25     @Override
26     protected void setupDirections() {
27         this.setAdjacent(Direction.SOUTH, this.getWorld().getRoom("City Centre"));
28     }
29
30     @Override
31     public void spawnEntities() {
32         World world = this.getWorld();
33         world.spawnEntity("npc_marie", new EntityMarie(world, this.toLocation()));
34     }
35 }
```

```
1  package uk.insrt.coursework.zuul.content.campaign.rooms;
2
3  import uk.insrt.coursework.zuul.content.campaign.entities.EntityCat;
4  import uk.insrt.coursework.zuul.content.campaign.entities.EntityNPC;
5  import uk.insrt.coursework.zuul.world.Direction;
6  import uk.insrt.coursework.zuul.world.Room;
7  import uk.insrt.coursework.zuul.world.World;
8
9  /**
10   * The city centre connecting most major locations.
11   */
12  public class RoomCityCentre extends CampaignRoom {
13      public RoomCityCentre(World world) {
14          super(world, "City Centre");
15      }
16
17      @Override
18      public String describe() {
19          var world = this.getWorld();
20          if (!world.hasVisited(this)) {
21              return "<city_centre.first_load>";
22          }
23
24          return "<city_centre.enter>";
25      }
26
27      @Override
28      protected void setupDirections() {
29          World world = this.getWorld();
30          this.setAdjacent(Direction.NORTH, world.getRoom("Back Alley"));
31          this.setAdjacent(Direction.NORTH_WEST, world.getRoom("Street"));
32          this.setAdjacent(Direction.WEST, world.getRoom("Apartments: Reception"));
33          this.setAdjacent(Direction.SOUTH, world.getRoom("Coastline"));
34      }
35
36      @Override
37      public void spawnEntities() {
38          World world = this.getWorld();
39
40          EntityCat cat = new EntityCat(world, this.toLocation());
41          world.spawnEntity("cat", cat);
42      }
43  }
```

```
42     cat.useWanderAI(  
43         new Room[] {  
44             world.getRoom("City Centre"),  
45             world.getRoom("Street"),  
46             world.getRoom("Shop"),  
47             world.getRoom("Street"),  
48             world.getRoom("City Centre"),  
49             world.getRoom("Back Alley"),  
50             world.getRoom("City Centre")  
51         },  
52         8  
53     );  
54  
55     world.spawnEntity("city_npc",  
56         new EntityNPC(  
57             world,  
58             this.toLocation(),  
59             "npc_city_centre",  
60             "<city_centre.npc.description>",  
61             new String[] { "stranger", "person", "people" }  
62         ));  
63 }  
64 }
```

```
1  package uk.insrt.coursework.zuul.content.campaign.rooms;
2
3  import uk.insrt.coursework.zuul.content.campaign.entities.EntityBoat;
4  import uk.insrt.coursework.zuul.world.Direction;
5  import uk.insrt.coursework.zuul.world.World;
6
7  /**
8   * The coast which connects the main city to the mainland.
9   */
10 public class RoomCoastline extends CampaignRoom {
11     public RoomCoastline(World world) {
12         super(world, "Coastline");
13     }
14
15     @Override
16     public String describe() {
17         return "<coastline.enter>";
18     }
19
20     @Override
21     protected void setupDirections() {
22         this.setAdjacent(Direction.NORTH, this.getWorld().getRoom("City Centre"));
23     }
24
25     @Override
26     public void spawnEntities() {
27         World world = this.getWorld();
28         world.spawnEntity("boat1",
29             new EntityBoat(world, this.toLocation(),
30                 world.getRoom("Mainland: Coastline")));
31     }
32 }
```

```
1  package uk.insrt.coursework.zuul.content.campaign.rooms;
2
3  import uk.insrt.coursework.zuul.content.campaign.entities.EntityOldMan;
4  import uk.insrt.coursework.zuul.world.Direction;
5  import uk.insrt.coursework.zuul.world.World;
6
7  /**
8   * A forest on the mainland side.
9   */
10 public class RoomForest extends CampaignRoom {
11     public RoomForest(World world) {
12         super(world, "Forest");
13     }
14
15     @Override
16     public String describe() {
17         return "<forest.enter>";
18     }
19
20     @Override
21     protected void setupDirections() {
22         World world = this.getWorld();
23         this.setAdjacent(Direction.NORTH, world.getRoom("Mainland: Coastline"));
24         this.setAdjacent(Direction.EAST, world.getRoom("Worm Hole"));
25     }
26
27     @Override
28     public void spawnEntities() {
29         World world = this.getWorld();
30         world.spawnEntity("npc_old_man", new EntityOldMan(world, this.toLocation()));
31     }
32 }
```

```
1  package uk.insrt.coursework.zuul.content.campaign.rooms;
2
3  import uk.insrt.coursework.zuul.content.campaign.entities.EntityBoat;
4  import uk.insrt.coursework.zuul.world.Direction;
5  import uk.insrt.coursework.zuul.world.World;
6
7  /**
8   * The coast which connects the mainland to the main city.
9   */
10 public class RoomMainlandCoastline extends CampaignRoom {
11     public RoomMainlandCoastline(World world) {
12         super(world, "Mainland: Coastline");
13     }
14
15     @Override
16     public String describe() {
17         return "<mainland_coastline.enter>";
18     }
19
20     @Override
21     protected void setupDirections() {
22         this.setAdjacent(Direction.SOUTH, this.getWorld().getRoom("Forest"));
23     }
24
25     @Override
26     public void spawnEntities() {
27         World world = this.getWorld();
28         world.spawnEntity("boat2",
29             new EntityBoat(world, this.toLocation(),
30                 world.getRoom("Coastline")));
31     }
32 }
```

```
1  package uk.insrt.coursework.zuul.content.campaign.rooms;
2
3  import uk.insrt.coursework.zuul.content.campaign.entities.EntityDocument;
4  import uk.insrt.coursework.zuul.world.Direction;
5  import uk.insrt.coursework.zuul.world.World;
6
7  /**
8   * Private, usually inaccessible room within the Medical Centre complex.
9   */
10 public class RoomMedicalCentreOffice extends CampaignRoom {
11     public RoomMedicalCentreOffice(World world) {
12         super(world, "Medical Centre: Office");
13     }
14
15     @Override
16     public String describe() {
17         return "<medical_centre_office.enter>";
18     }
19
20     @Override
21     protected void setupDirections() {
22         this.setAdjacent(Direction.UP, this.getWorld().getRoom("Medical Centre: Reception"));
23     }
24
25     @Override
26     public void spawnEntities() {
27         World world = this.getWorld();
28         for (int i=1;i<=6;i++) {
29             world.spawnEntity("doc" + i, new EntityDocument(world, this.toLocation(), i));
30         }
31     }
32 }
```



```
1  package uk.insrt.coursework.zuul.content.campaign.rooms;
2
3  import uk.insrt.coursework.zuul.content.campaign.entities.EntityCouch;
4  import uk.insrt.coursework.zuul.content.campaign.entities.EntityNPC;
5  import uk.insrt.coursework.zuul.entities.Entity;
6  import uk.insrt.coursework.zuul.events.world.EventEntityLeftRoom;
7  import uk.insrt.coursework.zuul.world.Direction;
8  import uk.insrt.coursework.zuul.world.World;
9
10 /**
11  * Reception of the Medical Centre complex.
12  */
13 public class RoomMedicalCentreReception extends CampaignRoom {
14     private Entity guardEntity;
15     private EntityCouch couchEntity;
16
17     public RoomMedicalCentreReception(World world) {
18         super(world, "Medical Centre: Reception");
19     }
20
21     @Override
22     public String describe() {
23         return "<medical_centre.enter>";
24     }
25
26     @Override
27     protected void setupDirections() {
28         World world = this.getWorld();
29         this.setAdjacent(Direction.EAST, world.getRoom("Street"));
30         this.setAdjacent(Direction.DOWN, world.getRoom("Medical Centre: Office"));
31     }
32
33     @Override
34     public boolean canLeave(Direction direction) {
35         if (direction == Direction.DOWN) {
36             if (this.guardEntity.getRoom() == this) {
37                 this.getWorld()
38                     .getIO()
39                     .println("<medical_centre.guard.blocking>");
40
41                 return false;
42             }
43         }
44         return true;
45     }
46 }
```

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

```
1  package uk.insrt.coursework.zuul.content.campaign.rooms;
2
3  import uk.insrt.coursework.zuul.content.campaign.entities.EntityNPC;
4  import uk.insrt.coursework.zuul.content.campaign.entities.EntityShopkeeper;
5  import uk.insrt.coursework.zuul.world.Direction;
6  import uk.insrt.coursework.zuul.world.World;
7
8  /**
9   * A shop within the city, the only one the player can interact with.
10  */
11  public class RoomShop extends CampaignRoom {
12      public RoomShop(World world) {
13          super(world, "Shop");
14      }
15
16      @Override
17      public String describe() {
18          return "<shop.enter>";
19      }
20
21      @Override
22      protected void setupDirections() {
23          this.setAdjacent(Direction.SOUTH, this.getWorld().getRoom("Street"));
24      }
25
26      @Override
27      public void spawnEntities() {
28          World world = this.getWorld();
29          world.spawnEntity("npc_shopkeeper",
30              new EntityShopkeeper(world, this.toLocation()));
31      }
32  }
```

```
1  package uk.insrt.coursework.zuul.content.campaign.rooms;
2
3  import uk.insrt.coursework.zuul.content.campaign.StoryFlags.Stage;
4  import uk.insrt.coursework.zuul.content.campaign.entities.EntityNPC;
5  import uk.insrt.coursework.zuul.content.campaign.events.EventGameStageChanged;
6  import uk.insrt.coursework.zuul.entities.Entity;
7  import uk.insrt.coursework.zuul.events.IEventListener;
8  import uk.insrt.coursework.zuul.world.Direction;
9  import uk.insrt.coursework.zuul.world.World;
10
11  /**
12   * One of the major connecting points between locations in the city.
13   */
14  public class RoomStreet extends CampaignRoom implements IEventListener<EventGameStageChanged> {
15      private Entity protestorsEntity;
16
17      public RoomStreet(World world) {
18          super(world, "Street");
19      }
20
21      @Override
22      public String describe() {
23          var world = this.getWorld();
24          if (!world.hasVisited(this)) {
25              return "<street.first_load>";
26          }
27
28          return "<street.enter>";
29      }
30
31      @Override
32      protected void setupDirections() {
33          World world = this.getWorld();
34          this.setAdjacent(Direction.SOUTH, world.getRoom("Apartments: Reception"));
35          this.setAdjacent(Direction.EAST, world.getRoom("City Centre"));
36          this.setAdjacent(Direction.NORTH, world.getRoom("Shop"));
37          this.setAdjacent(Direction.WEST, world.getRoom("Medical Centre: Reception"));
38      }
39
40      @Override
41      public boolean canLeave(Direction direction) {
```

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

```
1  package uk.insrt.coursework.zuul.content.campaign.rooms;
2
3  import java.util.Random;
4
5  import uk.insrt.coursework.zuul.entities.Entity;
6  import uk.insrt.coursework.zuul.entities.EntityPlayer;
7  import uk.insrt.coursework.zuul.events.IEventListener;
8  import uk.insrt.coursework.zuul.events.world.EventEntityEnteredRoom;
9  import uk.insrt.coursework.zuul.world.Room;
10 import uk.insrt.coursework.zuul.world.World;
11
12 /**
13  * Teleporter room implemented as required by the challenge tasks.
14  * Any Entity that walks into the worm hole is transported into a random public location.
15  */
16 public class RoomWormHole extends CampaignRoom implements IEventListener<EventEntityEnteredRoom> {
17     public RoomWormHole(World world) {
18         super(world, "Worm Hole");
19     }
20
21     @Override
22     public String describe() {
23         return "";
24     }
25
26     @Override
27     protected void setupDirections() {}
28
29     @Override
30     public void onEvent(EventEntityEnteredRoom event) {
31         Entity entity = event.getEntity();
32         Room room = entity.getRoom();
33         if (room != this) return;
34         event.stopPropagation();
35
36         // This is a restricted set of locations as to not break
37         // the game's plot, say if we were transported to the medical
38         // centre complex office when we're not meant to go there yet.
39         final Random random = new Random();
40         final String[] locations = {
41             "City Centre",
```

```
42         "Coastline",
43         "Mainland: Coastline",
44         "Forest",
45         "Street",
46         "Back Alley"
47     };
48
49     var io = this.getWorld().getIO();
50     io.println("\n<worm_hole.enter>");
51
52     try {
53         Thread.sleep(1000);
54
55         final int WIDTH = 79;
56
57         // Transport animation, this will take 1800 ms.
58         for (int i=0;i<5;i++) {
59             io.println("".repeat(i*3) + "\\ "
60                 + " ".repeat(WIDTH - i * 6 - 2) + "/" + "".repeat(i*3));
61
62             Thread.sleep(60);
63         }
64
65         for (int i=0;i<30;i++) {
66             var out = "";
67             for (int j=0;j<WIDTH;j++) {
68                 out += random.nextInt(8) == 0 ? "*" : " ";
69             }
70
71             io.println(out);
72             Thread.sleep(40);
73         }
74
75         for (int i=5;i>0;i--) {
76             io.println("".repeat(i*3) + "/"
77                 + " ".repeat(WIDTH - i * 6 - 2) + "\\ " + "".repeat(i*3));
78
79             Thread.sleep(60);
80         }
81     } catch (InterruptedException e) {
82         e.printStackTrace();
83     }
```

```
83         io.println("There was a disruption when travelling.");
84     }
85
86     io.print("\n");
87
88     // Pick a random location and put the entering entity in it.
89     String location = locations[random.nextInt(locations.length)];
90     Room target = this.getWorld().getRoom(location);
91     entity.setLocation(target);
92
93     // If it was the player, clear their walk history.
94     if (entity instanceof EntityPlayer) {
95         ((EntityPlayer) entity).clearHistory();
96     }
97 }
98 }
```



```
1  package uk.insrt.coursework.zuul.content.campaign;
2
3  import java.util.HashSet;
4
5  import uk.insrt.coursework.zuul.content.campaign.events.EventGameStageChanged;
6  import uk.insrt.coursework.zuul.events.EventSystem;
7  import uk.insrt.coursework.zuul.events.world.EventTick;
8
9  /**
10   * Class which controls story progression within the Campaign World.
11   */
12  public class StoryFlags {
13      /**
14       * The current story chapter.
15       */
16      public enum Stage {
17          Exposition, // Ch 1.
18          Recon, // Ch 2.
19          Stealth, // Ch 3.
20          End, // Current Ending
21
22          Twist, // Ch 4. Skipped
23          Conclusion, // Ch 5. Skipped
24      }
25
26      /**
27       * Side-quests available in the game.
28       */
29      public enum Quest {
30          Cat
31      }
32
33      private EventSystem eventSystem;
34      private HashSet<Quest> quests;
35      private int balance;
36      private Stage stage;
37      private int ticks;
38
39      /**
40       * Construct a new instance of StoryFlags
41       * @param eventSystem World event system
```

```
42     */
43     public StoryFlags(EventSystem eventSystem) {
44         this.eventSystem = eventSystem;
45         this.stage = Stage.Exposition;
46         this.quests = new HashSet<>();
47         this.balance = 100_000;
48         this.ticks = 0;
49
50         this.eventSystem.addListener(EventTick.class, e -> this.ticks++);
51     }
52
53     /**
54      * Get the current stage (chapter) of the story.
55      * @return Current stage
56      */
57     public Stage getStage() {
58         return this.stage;
59     }
60
61     /**
62      * Set the current stage (chapter) of the story.
63      * @param stage New stage
64      */
65     public void setStage(Stage stage) {
66         this.stage = stage;
67         this.eventSystem.emit(new EventGameStageChanged(stage));
68     }
69
70     /**
71      * Get the player's balance
72      * @return Player's balance
73      */
74     public int getBalance() {
75         return this.balance;
76     }
77
78     /**
79      * Set player's new balance.
80      * @param balance New balance
81      */
82     public void setBalance(int balance) {
```

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

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

```
1  package uk.insrt.coursework.zuul.dialogue;
2
3  import java.util.HashMap;
4
5  import uk.insrt.coursework.zuul.io.IOSystem;
6
7  /**
8   * Simple dialogue engine which navigates between {@link DialogueNode}(s).
9   */
10 public class Dialogue<T> {
11     private HashMap<T, DialogueNode<T>> parts;
12     private T currentNode;
13
14     /**
15      * Construct a new Dialogue engine.
16      */
17     public Dialogue() {
18         this.parts = new HashMap<>();
19     }
20
21     /**
22      * Construct a new Dialogue engine and initialise us at a starting node.
23      * @param start Starting node
24      */
25     public Dialogue(T start) {
26         this();
27         this.currentNode = start;
28     }
29
30     /**
31      * Get the current node
32      */
33     public T getCurrentNode() {
34         return this.currentNode;
35     }
36
37     /**
38      * Set the current node
39      * @param node New node
40      */
41     public void setCurrentNode(T node) {
```

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

```
83     if (target == null) {
84         T newTarget = option.getTarget();
85         if (newTarget != null) {
86             this.currentNode = newTarget;
87         }
88
89
90         return;
91     }
92
93     this.currentNode = target;
94     this.run(io);
95 }
96 }
```

```
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 il8n is being used in the dialogue data.
15  */
16 public class DialogueLoader {
17     private Map<String, Object> data;
18
19     /**
20      * Construct a new DialogueLoader
21      */
22     public DialogueLoader() {
23         this.data = new HashMap<>();
24     }
25
26     /**
27      * Load all necessary data for populating Dialogue.
28      * @param path Path to the dialogue resource file
29      * @throws IOException if we can't read the dialogue file
30      */
31     public void load(String path) throws IOException {
32         InputStream stream = DialogueLoader.class.getResourceAsStream(path);
33         this.data = new Toml().read(stream).toMap();
34     }
35
36     /**
37      * Populate a Dialogue system using a specific dialog definition represented by a given key.
38      *
39      * This method is unchecked as we expect a valid data structure to have
40      * been loaded from the resource, this should be verified by the developer.
41      * @param dialogue Dialogue system
```



```
42     * @param key Key to lookup
43     */
44     @SuppressWarnings("unchecked")
45     public void populate(Dialogue<String> dialogue, String key) {
46         Map<String, Object> nodes = (Map<String, Object>) this.data.get(key);
47
48         // Process any special keys first before we continue.
49         String prefix = "";
50         for (String nodeKey : nodes.keySet()) {
51             if (nodeKey.equals("_prefix")) {
52                 prefix = (String) nodes.get(nodeKey);
53             } else if (nodeKey.equals("_start")) {
54                 dialogue.getCurrentNode((String) nodes.get(nodeKey));
55             }
56         }
57
58         for (Entry<String, Object> node : nodes.entrySet()) {
59             // Ignore any keys starting with _, as they are used above.
60             String nodeKey = node.getKey();
61             if (nodeKey.startsWith("_")) continue;
62
63             // Read each node's values and find the description and options.
64             Map<String, Object> values = (Map<String, Object>) node.getValue();
65
66             // Description strings are assumed to be i18n paths.
67             String description = "<" + prefix + (String) values.get("description") + ">";
68             List<Map<String, Object>> options = (List<Map<String, Object>>) values.get("options");
69
70             // Construct a new Dialogue Node with the given data.
71             DialogueNode<String> dialogueNode = new DialogueNode<>(description);
72             for (Map<String, Object> object : options) {
73                 String desc = "<" + prefix + (String) object.get("description") + ">";
74                 String to = (String) object.get("to");
75                 Boolean mustExit = (Boolean) object.get("mustExit");
76
77                 if (mustExit == null) {
78                     dialogueNode.addOption(desc, to);
79                 } else {
80                     dialogueNode.addOption(desc, to, mustExit);
81                 }
82             }
83         }
84     }
```

```
83
84     dialogue.addPart(nodeKey, dialogueNode);
85 }
86 }
87 }
```

```
1  package uk.insrt.coursework.zuul.dialogue;
2
3  import java.util.ArrayList;
4  import java.util.List;
5
6  import uk.insrt.coursework.zuul.io.IOSystem;
7
8  /**
9   * A node in a {@link Dialogue} system.
10  */
11  public class DialogueNode<T> {
12      private String description;
13      private ArrayList<DialogueOption<T>> options;
14
15      /**
16       * Construct a new node.
17       * @param description Description of this node
18       */
19      public DialogueNode(String description) {
20          this.description = description;
21          this.options = new ArrayList<>();
22      }
23
24      /**
25       * Get this node's description
26       * @return Description string
27       */
28      public String getDescription() {
29          return this.description;
30      }
31
32      /**
33       * Add a new option which branches off this node.
34       * @param option Dialogue Option
35       * @return This Dialogue Node so other method calls can be chained
36       */
37      public DialogueNode<T> addOption(DialogueOption<T> option) {
38          this.options.add(option);
39          return this;
40      }
41  }
```

```
42  /**
43   * Create a new option which branches off this node.
44   * @param description Description of this option
45   * @param stage The next stage of the dialogue this should jump to
46   * @param mustExit Whether we must exit from the dialogue after selecting this option
47   * @return This Dialogue Node so other method calls can be chained
48   */
49  public DialogueNode<T> addOption(String description, T stage, boolean mustExit) {
50      var option = new DialogueOption<T>(description, stage);
51      if (mustExit) option.mustExit();
52      this.options.add(option);
53      return this;
54  }
55
56  /**
57   * Create a new option which branches off this node.
58   * @param description Description of this option
59   * @param stage The next stage of the dialogue this should jump to
60   * @return This Dialogue Node so other method calls can be chained
61   */
62  public DialogueNode<T> addOption(String description, T stage) {
63      return this.addOption(description, stage, false);
64  }
65
66  /**
67   * Get the options available to this node.
68   * @return List of options
69   */
70  protected List<DialogueOption<T>> getOptions() {
71      return this.options;
72  }
73
74  /**
75   * Ask the player to pick one of the valid options branching off this node.
76   * @param io Provided IO system
77   * @return The selected option
78   */
79  public DialogueOption<T> pickOption(IOSystem io) {
80      List<DialogueOption<T>> options = this.getOptions();
81      for (int i=0;i<options.size();i++) {
82          io.println((i + 1) + ". " + options.get(i).getDescription());
```

```
83     }
84
85     while (true) {
86         io.print("Choice: ");
87         String value = io.readLine();
88         try {
89             int v = Integer.parseInt(value);
90             if (v < 1 || v > options.size()) {
91                 io.println("Provide a valid option!");
92                 continue;
93             }
94
95             return options.get(v - 1);
96         } catch (Exception e) {
97             io.println("Provide a valid number!");
98         }
99     }
100 }
101 }
```

```
1  package uk.insrt.coursework.zuul.dialogue;
2
3  import uk.insrt.coursework.zuul.io.IOSystem;
4
5  /**
6   * An option which branches off a {@link DialogueNode} into another node.
7   */
8  public class DialogueOption<T> {
9      private IDialogueHandler<T> handler;
10
11     private String description;
12     private boolean shouldExit;
13     private T target;
14
15     /**
16      * Construct a new simple DialogueOption with a description and destination.
17      * @param description Description of this option
18      * @param target Target node to jump to
19      */
20     public DialogueOption(String description, T target) {
21         this.target = target;
22         this.description = description;
23     }
24
25     /**
26      * Construct a complex DialogueOption with a description and select handler.
27      * @param description Description of this option
28      * @param handler Method called when this option is selected
29      */
30     public DialogueOption(String description, IDialogueHandler<T> handler) {
31         this.handler = handler;
32         this.description = description;
33     }
34
35     /**
36      * Tell the Dialogue system to exit if this option is selected.
37      * @return This dialogue option so method calls can be chained
38      */
39     public DialogueOption<T> mustExit() {
40         this.shouldExit = true;
41         return this;
42     }
43 }
```

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

```
1  package uk.insrt.coursework.zuul.dialogue;
2
3  import uk.insrt.coursework.zuul.io.IOSystem;
4
5  /**
6   * Interface implemented to provide an onAction method.
7   */
8  public interface IDialogueHandler<T> {
9      /**
10       * Handle the selection of a dialogue option.
11       * @param io Provided IO system
12       * @return Destination node, may be null
13       */
14     public T onAction(IOSystem io);
15 }
```



```
1 package uk.insrt.coursework.zuul.entities.actions;
2
3 import uk.insrt.coursework.zuul.entities.Entity;
4
5 /**
6  * Interface implemented to provide the ability for
7  * an Entity to have other Entities given to them.
8  */
9 public interface IGiveable {
10     /**
11      * Give this entity another entity.
12      * @param item Item being given
13      */
14     public void give(Entity item);
15 }
```

```
1 package uk.insrt.coursework.zuul.entities.actions;
2
3 /**
4  * Interface implemented to provide the
5  * ability for an Entity to be pet.
6  */
7 public interface IPettable {
8     /**
9      * Pet this entity.
10     */
11     public void pet();
12 }
```

```
1  package uk.insrt.coursework.zuul.entities.actions;
2
3  /**
4   * Interface implemented to provide the
5   * ability for an Entity to talk with the player.
6   */
7  public interface ITalkwith {
8      /**
9       * Talk with this entity.
10      */
11      public void talk();
12  }
```

```
1  package uk.insrt.coursework.zuul.entities.actions;
2
3  import uk.insrt.coursework.zuul.entities.Entity;
4
5  /**
6   * Interface implemented to provide the
7   * ability for an Entity to be used by the player.
8   */
9  public interface IUseable {
10     /**
11      * Use this entity.
12      * @param target The Entity taking this entity.
13      */
14     public void use(Entity target);
15 }
```

```
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 public abstract class Entity {
19     protected World world;
20     protected Inventory inventory;
21
22     private Location location;
23     private double weight;
24
25     /**
26      * Construct a new Entity.
27      * @param world Current World object
28      * @param location Initial Location of this Entity
29      * @param weight The weight (in kg) of this Entity
30      */
31     public Entity(World world, Location location, double weight) {
32         this.world = world;
33         this.location = location;
34         this.inventory = new Inventory();
35         this.weight = weight;
36     }
37
38     /**
39      * Construct a new Entity.
40      *
41      * Weight value is set to {@link Integer#MAX_VALUE}.
```

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

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

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

```
165     * @return String describing the Entity
166     */
167     public abstract String describe();
168 }
```

```
1  package uk.insrt.coursework.zuul.entities;
2
3  import uk.insrt.coursework.zuul.world.Location;
4  import uk.insrt.coursework.zuul.world.World;
5
6  /**
7   * Generic object class which avoids some boilerplate.
8   * Use this for entities which are guaranteed to never change.
9   */
10 public class EntityObject extends Entity {
11     private String description;
12     private String[] aliases;
13
14     /**
15      * Construct a new EntityObject
16      * @param world Current World object
17      * @param location Initial Location of this Entity
18      * @param weight The weight (in kg) of this Entity
19      * @param aliases Aliases which this object can be referred to by
20      * @param description A description of this object
21      */
22     public EntityObject(World world, Location location, double weight, String[] aliases, String description) {
23         super(world, location, weight);
24         this.description = description;
25         this.aliases = aliases;
26     }
27
28     /**
29      * Construct a new EntityObject
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      * @param name Name of this object
34      * @param description A description of this object
35      */
36     public EntityObject(World world, Location location, double weight, String alias, String description) {
37         this(world, location, weight, new String[] { alias }, description);
38     }
39
40     @Override
41     public String describe() {
```

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

```
1  package uk.insrt.coursework.zuul.entities;
2
3  import java.util.ArrayList;
4
5  import uk.insrt.coursework.zuul.io.IOSystem;
6  import uk.insrt.coursework.zuul.world.Direction;
7  import uk.insrt.coursework.zuul.world.Location;
8  import uk.insrt.coursework.zuul.world.Room;
9  import uk.insrt.coursework.zuul.world.World;
10
11  /**
12   * Player entity which we can control and move around.
13   */
14  public class EntityPlayer extends Entity {
15      private ArrayList<Room> previousRooms;
16      private ArrayList<Direction> retreatingDirection;
17
18      /**
19       * Construct a new Player Entity.
20       * @param world World to place Player in
21       */
22      public EntityPlayer(World world) {
23          super(world, new Location(), 70);
24          this.previousRooms = new ArrayList<>();
25          this.retreatingDirection = new ArrayList<>();
26          this.inventory.setMaxWeight(35);
27      }
28
29      @Override
30      public String[] getAliases() {
31          return new String[] {
32              "player", "me", "myself", "self", "yourself"
33          };
34      }
35
36      @Override
37      public String describe() {
38          // We may skip defining how the Player looks,
39          // this is because EntityPlayer is ignored
40          // when looking around the room.
41          return "";
```

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

```
83         if (this.getRoom().hasExit(lastDirection)) {
84             this.retreatingDirection.remove(index);
85             this.setLocation(this.previousRooms.remove(index));
86         } else {
87             io.println("Cannot leave the room this way.");
88         }
89     }
90
91     /**
92      * Clear walk history.
93      */
94     public void clearHistory() {
95         this.retreatingDirection.clear();
96         this.previousRooms.clear();
97     }
98 }
```

```
1  package uk.insrt.coursework.zuul.entities;
2
3  import java.util.ArrayList;
4
5  /**
6   * Representation of an Entity's inventory
7   * and what they are holding.
8   */
9  public class Inventory {
10     private ArrayList<Entity> items = new ArrayList<>();
11     private double maxWeight;
12
13     /**
14      * Construct a new Inventory.
15      */
16     public Inventory() {
17         super();
18         this.maxWeight = 0;
19     }
20
21     /**
22      * Set the max weight that can be carried in this inventory.
23      * @param maxWeight Max weight (in kg)
24      */
25     public void setMaxWeight(double maxWeight) {
26         this.maxWeight = maxWeight;
27     }
28
29     /**
30      * Get the maximum weight that can be carried in this inventory.
31      * @return Maximum weight that can be carried
32      */
33     public double getMaxWeight() {
34         return this.maxWeight;
35     }
36
37     /**
38      * Get the current weight of this inventory.
39      * @return Weight (in kg)
40      */
41     public double getWeight() {
```

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

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

```
1  package uk.insrt.coursework.zuul.events;
2
3  /**
4   * Represents a single event fired from
5   * any source to be consumed by anything.
6   */
7  public class Event {
8      private boolean propagating = true;
9
10     /**
11      * Whether this event can continue running.
12      * @return Whether propagation of this event was stopped
13      */
14     public boolean canRun() {
15         return this.propagating;
16     }
17
18     /**
19      * Stop further propagation of this event.
20      */
21     public void stopPropagation() {
22         this.propagating = false;
23     }
24 }
```

```
1  package uk.insrt.coursework.zuul.events;
2
3  import java.util.HashMap;
4  import java.util.HashSet;
5  import java.util.LinkedHashSet;
6
7  /**
8   * Event system which manages taking in events
9   * from different sources and handles them
10  * by firing callbacks on event listeners.
11  */
12  public class EventSystem {
13      private HashMap<Class<? extends Event>, LinkedHashSet<IEventListener<? extends Event>>> listeners = new HashMap<>();
14
15      /**
16       * Get existing Event listener list or create a new one if not exists.
17       * @param event Event
18       * @return Set of event listeners
19       */
20      private HashSet<IEventListener<? extends Event>> getList(Class<? extends Event> event) {
21          var list = this.listeners.get(event);
22          if (list == null) {
23              list = new LinkedHashSet<>();
24              this.listeners.put(event, list);
25          }
26
27          return list;
28      }
29
30      /**
31       * Add a new event listener to this system.
32       * @param <E> Generic Event type
33       * @param event Event to remove from
34       * @param listener Event listener callback
35       */
36      public<E extends Event> void addListener(Class<E> event, IEventListener<E> listener) {
37          this.getList(event).add(listener);
38      }
39
40      /**
41       * Remove an new event listener from this system.
```

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

```
1  package uk.insrt.coursework.zuul.events;
2
3  /**
4   * Interface implementing an listener for an arbitrary {@link Event}.
5   */
6  public interface IEventListener<E extends Event> {
7      /**
8       * Method called when this specific Event is emitted.
9       * @param event Event to handle
10      */
11     public void onEvent(E event);
12 }
```

```
1  package uk.insrt.coursework.zuul.events.world.behaviours;
2
3  import java.util.Random;
4
5  import uk.insrt.coursework.zuul.entities.Entity;
6  import uk.insrt.coursework.zuul.events.IEventListener;
7  import uk.insrt.coursework.zuul.events.world.EventTick;
8  import uk.insrt.coursework.zuul.world.Room;
9
10 /**
11  * This is a simple behaviour which just randomly decides to move an Entity
12  * through a set path whenever the game ticks forwards.
13  */
14 public class SimpleWanderAI implements IEventListener<EventTick> {
15     private Entity entity;
16     private Room[] path;
17     private int chance;
18
19     private int index;
20     private Random random;
21
22     /**
23      * Construct a new wandering behaviour for an Entity with a given path.
24      * @param entity Entity which should be moved
25      * @param path Path that this Entity should follow
26      * @param chance The chance x that this entity moves, where x gives a 1/x fractional chance of moving.
27      */
28     public SimpleWanderAI(Entity entity, Room[] path, int chance) {
29         this.entity = entity;
30         this.path = path;
31         this.chance = chance;
32
33         this.index = 0;
34         this.random = new Random();
35     }
36
37     @Override
38     public void onEvent(EventTick event) {
39         if (this.entity.getRoom() != this.path[this.index]) return;
40         if (random.nextInt(this.chance) > 0) return;
41     }
```

```
42     this.index = (this.index + 1) % this.path.length;
43     this.entity.setLocation(this.path[this.index]);
44 }
45 }
```

```
1  package uk.insrt.coursework.zuul.events.world;
2
3  import uk.insrt.coursework.zuul.entities.Entity;
4  import uk.insrt.coursework.zuul.events.Event;
5
6  /**
7   * Event fired when an Entity enters a room.
8   */
9  public class EventEntityEnteredRoom extends Event {
10     private Entity entity;
11
12     /**
13      * Construct a new EntityEnteredRoom Event.
14      * @param entity Target Entity
15      */
16     public EventEntityEnteredRoom(Entity entity) {
17         this.entity = entity;
18     }
19
20     /**
21      * Get the Entity relating to this event.
22      * @return Entity
23      */
24     public Entity getEntity() {
25         return this.entity;
26     }
27 }
```



```
1  package uk.insrt.coursework.zuul.events.world;
2
3  import uk.insrt.coursework.zuul.entities.Entity;
4  import uk.insrt.coursework.zuul.events.Event;
5  import uk.insrt.coursework.zuul.world.Room;
6
7  /**
8   * Event fired when an Entity enters a room.
9   */
10 public class EventEntityLeftRoom extends Event {
11     private Entity entity;
12     private Room room;
13
14     /**
15      * Construct a new EntityLeftRoom Event.
16      * @param entity Target Entity
17      * @param room Room the entity left
18      */
19     public EventEntityLeftRoom(Entity entity, Room room) {
20         this.entity = entity;
21         this.room = room;
22     }
23
24     /**
25      * Get the Entity relating to this event.
26      * @return Entity
27      */
28     public Entity getEntity() {
29         return this.entity;
30     }
31
32     /**
33      * Get the Room relating to this event.
34      * @return Room
35      */
36     public Room getRoom() {
37         return this.room;
38     }
39 }
```

```
1  package uk.insrt.coursework.zuul.events.world;
2
3  import uk.insrt.coursework.zuul.events.Event;
4
5  /**
6   * Event fired when an arbitrary command is about to be run.
7   */
8  public class EventProcessCommand extends Event {
9      private String cmd;
10
11     /**
12      * Construct a new EventProcessCommand Event.
13      * @param cmd Target command
14      */
15     public EventProcessCommand(String cmd) {
16         this.cmd = cmd;
17     }
18
19     /**
20      * Set command for this event.
21      * @param cmd Overwrite current command
22      */
23     public void setCommand(String cmd) {
24         this.cmd = cmd;
25     }
26
27     /**
28      * Get the command relating to this event.
29      * @return Arbitrary command
30      */
31     public String getCommand() {
32         return this.cmd;
33     }
34 }
```

```
1 package uk.insrt.coursework.zuul.events.world;
2
3 import uk.insrt.coursework.zuul.events.Event;
4
5 /**
6  * Event fired when the game ticks forward.
7  * Such as when the player performs an action or goes to sleep.
8  */
9 public class EventTick extends Event {}
```

```
1 package uk.insrt.coursework.zuul;
2
3 import java.io.IOException;
4
5 import javax.swing.JOptionPane;
6
7 import uk.insrt.coursework.zuul.commands.CommandManager;
8 import uk.insrt.coursework.zuul.content.campaign.CampaignWorld;
9 import uk.insrt.coursework.zuul.content.campaign.commands.CommandMap;
10 import uk.insrt.coursework.zuul.content.campaign.commands.CommandWin;
11 import uk.insrt.coursework.zuul.events.world.EventProcessCommand;
12 import uk.insrt.coursework.zuul.events.world.EventTick;
13 import uk.insrt.coursework.zuul.io.IOSystem;
14 import uk.insrt.coursework.zuul.io.LocalisedIO;
15 import uk.insrt.coursework.zuul.io.SanitiseIO;
16 import uk.insrt.coursework.zuul.io.StandardIO;
17 import uk.insrt.coursework.zuul.ui.EventDraw;
18 import uk.insrt.coursework.zuul.ui.TerminalEmulator;
19 import uk.insrt.coursework.zuul.util.BlueJ;
20 import uk.insrt.coursework.zuul.util.Localisation;
21 import uk.insrt.coursework.zuul.world.World;
22
23 /**
24  * Class for managing the game loop and initialising the world.
25  */
26 public class Game {
27     public static final String GAME_NAME = "World of Deez";
28
29     private World world;
30     private IOSystem io;
31     private CommandManager commands;
32
33     /**
34      * Entrypoint to our application.
35      * @param args Arguments provided to the application
36      */
37     public static void main(String[] args) {
38         new Game().play();
39     }
40
41     /**
```

```

42     * Initialise and start the game.
43     */
44     public void play() {
45         this.init();
46         this.start();
47     }
48
49     /**
50     * Initialise all required resources for the game to run.
51     */
52     private void init() {
53         // Determine how the game should run.
54         boolean inBlueJ = BlueJ.isRunningInBlueJ();
55         int selection = JOptionPane.showConfirmDialog(null, "Play full experience?\nUses custom terminal emulator.
\n(recommended option)", GAME_NAME, JOptionPane.YES_NO_OPTION);
56         if (selection == 0) {
57             if (inBlueJ) {
58                 // Fullscreen minimises itself immediately
59                 // when running from BlueJ, not sure what's
60                 // going on exactly, just disabling it in general.
61                 selection = 1;
62             } else {
63                 selection = JOptionPane.showConfirmDialog(null, "Immersive mode?\nRuns emulator in fullscreen.
\n(recommended option)", GAME_NAME, JOptionPane.YES_NO_OPTION);
64             }
65
66             this.io = new TerminalEmulator(selection == 0);
67         } else {
68             this.io = new StandardIO();
69
70             if (inBlueJ) {
71                 selection = JOptionPane.showConfirmDialog(null, "Is this running from inside BlueJ?", GAME_NAME, JOptionPane.YES_NO_OPTION);
72                 if (selection == 0) {
73                     this.io = new SanitiseIO(this.io);
74                 }
75             }
76         }
77
78         // Setup the command manager.
79         this.commands = new CommandManager();

```

```
80     this.commands.registerCommand(new CommandWin());
81
82     // Register the Map command if we're in term emu mode.
83     // We draw images here so it's not available generally.
84     if (this.io instanceof TerminalEmulator) {
85         CommandMap map = new CommandMap();
86         this.commands.registerCommand(map);
87         ((TerminalEmulator) this.io).getEventSystem().addListener(EventDraw.class, map);
88     }
89
90     // Load all the data we need and initialise world.
91     Localisation locale = new Localisation();
92     this.io = new LocalisedIO(this.io, locale);
93
94     try {
95         locale.loadLocale("en_GB");
96     } catch (IOException e) {
97         System.err.println("Failed to load translations!");
98         e.printStackTrace();
99     }
100
101     this.world = new CampaignWorld(this.io);
102 }
103
104 /**
105  * Start the game loop.
106  */
107 private void start() {
108     this.world.spawnPlayer();
109
110     while (true) {
111         this.io.print("> ");
112         String input = this.io.readLine().toLowerCase();
113
114         EventProcessCommand event = new EventProcessCommand(input);
115         this.world.emit(event);
116
117         if (this.commands.runCommand(this.world, event.getCommand())) {
118             break;
119         }
120     }
```

```
121         this.world.emit(new EventTick());
122     }
123
124     this.io.println("Goodbye.");
125
126     try {
127         Thread.sleep(1000);
128         this.io.dispose();
129     } catch (Exception e) {}
130 }
131 }
```

```
1  package uk.insrt.coursework.zuul.io;
2
3  import java.awt.Color;
4  import java.util.regex.Pattern;
5
6  /**
7   * ANSI escape codes
8   * Used https://stackoverflow.com/a/5762502 as a reference.
9   */
10 public class Ansi {
11     public static final String Reset = "\u001B[0m";
12     public static final String Black = "\u001B[30m";
13     public static final String Red = "\u001B[31m";
14     public static final String Green = "\u001B[32m";
15     public static final String Yellow = "\u001B[33m";
16     public static final String Blue = "\u001B[34m";
17     public static final String Purple = "\u001B[35m";
18     public static final String Cyan = "\u001B[36m";
19     public static final String White = "\u001B[37m";
20
21     public static final String BackgroundBlack = "\u001B[40m";
22     public static final String BackgroundRed = "\u001B[41m";
23     public static final String BackgroundGreen = "\u001B[42m";
24     public static final String BackgroundYellow = "\u001B[43m";
25     public static final String BackgroundBlue = "\u001B[44m";
26     public static final String BackgroundPurple = "\u001B[45m";
27     public static final String BackgroundCyan = "\u001B[46m";
28     public static final String BackgroundWhite = "\u001B[47m";
29
30     /**
31      * Regex Pattern used to match Ansi codes forwards.
32      */
33     public static final Pattern AnsiPattern = Pattern.compile("^\\u001B\\[(\\d{1,3})m");
34
35     private static final Color ColorBlack = new Color(0, 0, 0);
36     private static final Color ColorRed = new Color(224, 108, 117);
37     private static final Color ColorGreen = new Color(152, 195, 121);
38     private static final Color ColorYellow = new Color(229, 192, 123);
39     private static final Color ColorBlue = new Color(97, 175, 239);
40     private static final Color ColorMagenta = new Color(198, 120, 221);
41     private static final Color ColorCyan = new Color(86, 182, 194);
```



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

```
1 package uk.insrt.coursework.zuul.io;
2
3 /**
4  * Interface representing an arbitrary IO system.
5  * This can be implemented to input or output from various interfaces.
6  */
7 public interface IOSystem {
8     /**
9      * Print a string out through an arbitrary output channel.
10     * @param out String to print
11     */
12     public void print(String out);
13
14     /**
15      * Print a string out through an arbitrary output channel and append {@code \n}.
16      * @param out String to print
17      */
18     public void println(String out);
19
20     /**
21      * Read a String up until the first encountered {@code \n} from an arbitrary input channel.
22      * @return String of line read in
23      */
24     public String readLine();
25
26     /**
27      * Dispose of the arbitrary input and output channels.
28      */
29     public void dispose();
30 }
```

```
1  package uk.insrt.coursework.zuul.io;
2
3  import java.util.regex.Matcher;
4  import java.util.regex.Pattern;
5
6  import uk.insrt.coursework.zuul.util.Localisation;
7
8  /**
9   * Translate and localise any incoming output.
10  */
11  public class LocalisedIO implements IOSystem {
12      private final Pattern pattern = Pattern.compile("<([\\w\\.]+?)>");
13
14      private IOSystem io;
15      private Localisation locale;
16
17      /**
18       * Construct a new LocalisedIO.
19       * @param io Provided IO system we should feed into
20       * @param locale Locale to apply to any i18n strings
21       */
22      public LocalisedIO(IOSystem io, Localisation locale) {
23          this.io = io;
24          this.locale = locale;
25      }
26
27      /**
28       * Replace i18n strings in any given String with their actual localised values.
29       * Using replacement code from https://stackoverflow.com/a/27359491.
30       * @param input String to process
31       * @return Final processed string
32       */
33      private String replace(String input) {
34          StringBuffer result = new StringBuffer();
35          Matcher matcher = this.pattern.matcher(input);
36
37          while (matcher.find()) {
38              matcher.appendReplacement(result, this.locale.get(matcher.group(1)));
39          }
40
41          matcher.appendTail(result);
```

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

```
1  package uk.insrt.coursework.zuul.io;
2
3  /**
4   * Sanitise incoming output and remove any Ansi escape sequences.
5   * This is required to print out into the BlueJ console without additional characters.
6   */
7  public class SanitiseIO implements IOSystem {
8      private final String ansiPattern = "\\u001B\\[(\\d{1,3})m";
9      private IOSystem io;
10
11     /**
12      * Construct a new SanitiseIO.
13      * @param io Provided IO system we should feed into
14      */
15     public SanitiseIO(IOSystem io) {
16         this.io = io;
17     }
18
19     @Override
20     public void print(String out) {
21         this.io.print(out.replaceAll(this.ansiPattern, " "));
22     }
23
24     @Override
25     public void println(String out) {
26         this.io.println(out.replaceAll(this.ansiPattern, " "));
27     }
28
29     @Override
30     public String readLine() {
31         return this.io.readLine();
32     }
33
34     @Override
35     public void dispose() {
36         this.io.dispose();
37     }
38 }
```

```
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  public class StandardIO implements IOSystem {
10     private Scanner reader;
11
12     /**
13      * Construct a new StandardIO.
14      */
15     public StandardIO() {
16         this.reader = new Scanner(System.in);
17     }
18
19     @Override
20     public void print(String out) {
21         System.out.print(out);
22     }
23
24     @Override
25     public void println(String out) {
26         System.out.println(out);
27     }
28
29     @Override
30     public String readLine() {
31         return this.reader.nextLine();
32     }
33
34     @Override
35     public void dispose() {}
36 }
```

```
1 package uk.insrt.coursework.zuul.ui;
2
3 import java.awt.Image;
4
5 /**
6  * Representation of a single Emoji which can be rendered in the terminal emulator.
7  */
8 public class Emoji {
9     private Image image;
10    private int width;
11    private int height;
12
13    /**
14     * Construct a new Emoji given the image and unicode representation.
15     * @param image Image to render when this Emoji is used
16     * @param unicode Unicode representation of this Emoji, used to determine width
17     */
18    public Emoji(Image image, String unicode) {
19        this.image = image;
20        this.width = (int) unicode.chars().count();
21        this.height = 1;
22    }
23
24    /**
25     * Construct a new Emoji given the image and size constraints.
26     * @param image Image to render when this Emoji is used
27     * @param width Width of this Emoji
28     * @param height Height of this Emoji
29     */
30    public Emoji(Image image, int width, int height) {
31        this.image = image;
32        this.width = width;
33        this.height = height;
34    }
35
36    /**
37     * Get the Image for this Emoji
38     * @return Image
39     */
40    public Image getImage() {
41        return this.image;
42    }
43}
```

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



```
1  package uk.insrt.coursework.zuul.ui;
2
3  import java.awt.Image;
4  import java.io.IOException;
5  import java.io.InputStream;
6  import java.nio.charset.StandardCharsets;
7  import java.util.ArrayList;
8  import java.util.Arrays;
9  import java.util.HashMap;
10 import java.util.List;
11
12 import javax.imageio.ImageIO;
13
14 import com.moandjiezana.toml.Toml;
15
16 import org.apache.commons.io.IOUtils;
17
18 import uk.insrt.coursework.zuul.util.Tree;
19
20 /**
21  * Class which helps manage loading and resolving Emojis.
22  */
23 public class EmojiManager {
24     private HashMap<String, Emoji> emojis;
25     private Tree<Character, String> emojiTree;
26     private Tree<Character, String> currentNode;
27
28     /**
29      * Construct a new EmojiManager.
30      */
31     public EmojiManager() {
32         this.emojis = new HashMap<>();
33         this.emojiTree = new Tree<>();
34         this.currentNode = this.emojiTree;
35     }
36
37     /**
38      * Check whether a given Emoji is present in this manager.
39      * @param emoji Unicode representation of Emoji
40      * @return True if the Emoji is available
41      */
42 }
```

```
42 public boolean hasEmoji(String emoji) {
43     return this.emojis.containsKey(emoji);
44 }
45
46 /**
47  * Get an Emoji by its Unicode representation.
48  * @param emoji Unicode representation of Emoji
49  * @return The Emoji or null if it doesn't exist
50  */
51 public Emoji getEmoji(String emoji) {
52     return this.emojis.get(emoji);
53 }
54
55 /**
56  * Get currently matched emoji and resets position.
57  * @return Emoji if it was found, or null if not
58  */
59 public Emoji getEmoji() {
60     String value = this.currentNode.getValue();
61     Emoji emoji = this.emojis.get(value);
62     this.resetState();
63     return emoji;
64 }
65
66 /**
67  * Reset the state of the matching mechanism.
68  */
69 public void resetState() {
70     this.currentNode = this.emojiTree;
71 }
72
73 /**
74  * The matching mechanism has not matched any characters to potential emojis.
75  */
76 public static final int MATCH_NONE = 0;
77
78 /**
79  * The matching mechanism has matched some characters to potential emojis.
80  */
81 public static final int MATCH_SOME = 1;
82
```

```
83  /**
84   * The matching mechanism has matched an emoji.
85   */
86  public static final int MATCH_FOUND = 2;
87
88  /**
89   * Match the next character.
90   * @param c Character to match against
91   * @return One of {@link #MATCH_NONE}, {@link #MATCH_SOME} or {@link #MATCH_FOUND}
92   */
93  public int match(char c) {
94      var child = this.currentNode.getChild(c);
95      if (child != null) {
96          this.currentNode = child;
97          if (child.getValue() == null) return MATCH_SOME;
98          return MATCH_FOUND;
99      }
100
101      if (this.currentNode != this.emojiTree) {
102          this.currentNode = this.emojiTree;
103          child = this.emojiTree.getChild(c);
104          if (child != null) {
105              if (child.getValue() != null) return MATCH_SOME;
106              return MATCH_FOUND;
107          }
108      }
109
110      return MATCH_NONE;
111  }
112
113  /**
114   * Load emoji definitions and resources from a given resource directory.
115   * @param rootDir Root directory at which we expect a valid {@code definitions.toml} to exist
116   * @throws IOException if the definition file is missing or defined emojis are invalid
117   */
118  public void loadResources(String rootDir) throws IOException {
119      InputStream defnStream = this.getClass().getResourceAsStream(rootDir + "/definitions.toml");
120      // We need to force UTF-8 encoding or else unicode emojis may get mangled.
121      String defnString = IOUtils.toString(defnStream, StandardCharsets.UTF_8);
122      Toml defn = new Toml().read(defnString);
123      List<HashMap<String, Object>> emojis = defn.getList("emojis");
```

```

124
125 // Load each emoji in sequence
126 for (var emoji : emojis) {
127     String path = (String) emoji.get("path");
128     String unicode = (String) emoji.get("unicode");
129
130     InputStream stream = this.getClass().getResourceAsStream(rootDir + "/" + path);
131     Image image = ImageIO.read(stream);
132
133     Emoji newEmoji;
134     if (emoji.containsKey("width") && emoji.containsKey("height")) {
135         int width = ((Long) emoji.get("width")).intValue();
136         int height = ((Long) emoji.get("height")).intValue();
137         newEmoji = new Emoji(image, width, height);
138     } else {
139         newEmoji = new Emoji(image, unicode);
140     }
141     this.emojis.put(unicode, newEmoji);
142     this.emojiTree.addChildWithPath(
143         new ArrayList<>(Arrays.asList(
144             unicode.chars()
145                 .mapToObj(c -> (char)c)
146                 .toArray(Character[]::new))),
147         unicode
148     );
149 }
150 }
151 }

```

```
1  package uk.insrt.coursework.zuul.ui;
2
3  import java.awt.Graphics;
4
5  import uk.insrt.coursework.zuul.events.Event;
6
7  /**
8   * Event fired when the terminal emulator draws a new frame.
9   */
10 public class EventDraw extends Event {
11     private Graphics g;
12     private float ox;
13     private float oy;
14     private float fw;
15     private float fh;
16
17     /**
18      * Construct a new EventDraw Event.
19      * @param g Graphics context
20      * @param ox Origin X position
21      * @param oy Origin Y position
22      * @param fw Font character width
23      * @param fh Font character height
24      */
25     public EventDraw(Graphics g, float ox, float oy, float fw, float fh) {
26         this.g = g;
27         this.ox = ox;
28         this.oy = oy;
29         this.fw = fw;
30         this.fh = fh;
31     }
32
33     /**
34      * Get the Graphics relating to this event.
35      * @return Graphics
36      */
37     public Graphics getGraphics() {
38         return this.g;
39     }
40
41     /**
```

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

```
1  package uk.insrt.coursework.zuul.ui;
2
3  import java.awt.BorderLayout;
4  import java.awt.Dimension;
5  import java.awt.GraphicsEnvironment;
6  import java.awt.event.KeyEvent;
7  import java.awt.event.KeyListener;
8
9  import javax.swing.JFrame;
10
11  /**
12   * Window frame for {@link TerminalEmulator}
13   */
14  public class JTerminalFrame extends JFrame {
15      private JTerminalView view;
16      private boolean fullscreen;
17
18      /**
19       * Construct a new JTerminalFrame
20       * @param emulator Terminal emulator this frame belongs to
21       */
22      public JTerminalFrame(TerminalEmulator emulator) {
23          this.view = new JTerminalView(emulator);
24          this.fullscreen = emulator.isFullscreen();
25          this.makeFrame(emulator);
26      }
27
28      /**
29       * Make and display all of the elements within this frame.
30       * @param emulator Terminal emulator this frame belongs to
31       */
32      public void makeFrame(TerminalEmulator emulator) {
33          this.setLayout(new BorderLayout());
34          this.add(this.view);
35          this.pack();
36          this.setLocationRelativeTo(null);
37          this.setMinimumSize(new Dimension(640, 640));
38          this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
39
40          // Add a listener for any user input.
41          // https://stackoverflow.com/a/21970006
```

```
42     this.addKeyListener(new KeyListener() {
43         public void keyPressed(KeyEvent event) {
44             int code = event.getKeyCode();
45             switch (code) {
46                 case 8: {
47                     emulator.pop();
48                     return;
49                 }
50                 case 10: {
51                     emulator.flush();
52                     break;
53                 }
54                 default: {
55                     if ((code >= 65 && code <= 90) || (code >= 48 && code <= 57) || code == 32) {
56                         emulator.push(event.getKeyChar());
57                     }
58                 }
59             }
60         }
61
62         public void keyTyped(KeyEvent e) {}
63         public void keyReleased(KeyEvent e) {}
64     });
65
66     // We are ready to display, show everything.
67     // Prerequisite to making the frame fullscreen too.
68     this.setVisible(true);
69
70     // If we are allowed to launch in fullscreen, switch to that mode now.
71     if (this.fullscreen) {
72         GraphicsEnvironment
73             .getLocalGraphicsEnvironment()
74             .getDefaultScreenDevice()
75             .setFullScreenWindow(this);
76     }
77 }
78
79 @Override
80 public void dispose() {
81     super.dispose();
82     this.view.dispose();

```



```
83     }  
84 }
```

```
1  package uk.insrt.coursework.zuul.ui;
2
3  import java.awt.Color;
4  import java.awt.Dimension;
5  import java.awt.Font;
6  import java.awt.FontFormatException;
7  import java.awt.Graphics;
8  import java.awt.event.ComponentAdapter;
9  import java.awt.event.ComponentEvent;
10 import java.awt.font.FontRenderContext;
11 import java.awt.geom.AffineTransform;
12 import java.io.IOException;
13 import java.io.InputStream;
14
15 import javax.swing.JPanel;
16
17 import com.moandjiezana.toml.Toml;
18
19 /**
20  * Rendering component of {@link TerminalEmulator}
21  */
22 public class JTerminalView extends JPanel {
23     private TerminalEmulator emulator;
24
25     private EmojiManager emojiManager;
26     private Font derivedFont;
27     private Font font;
28
29     private Thread blinkThread;
30     private boolean blinkState;
31
32     private float fw, fh, foffset, fratio;
33
34     /**
35      * Construct a TerminalView
36      * @param emulator Terminal emulator this view belongs to
37      */
38     public JTerminalView(TerminalEmulator emulator) {
39         this.emulator = emulator;
40         this.emojiManager = new EmojiManager();
41         this.blinkState = false;
```

```
42     this.loadResources();
43     this.makeFrame();
44 }
45
46 /**
47  * Prepare the terminal view for rendering.
48  */
49 public void makeFrame() {
50     var view = this;
51     this.setBackground(Color.BLACK);
52
53     // Register a listener to repaint and adjust measurements when resizing window.
54     // https://stackoverflow.com/a/8917978
55     this.addComponentListener(new ComponentAdapter() {
56         public void componentResized(ComponentEvent e) {
57             view.setBackground(Color.BLACK);
58             view.deriveFont();
59             view.repaint();
60         }
61     });
62
63     // Start a new thread for blinking the cursor.
64     this.blinkThread = new Thread("Blink Thread") {
65         public void run() {
66             try {
67                 while (true) {
68                     view.blinkState = !view.blinkState;
69                     view.repaint();
70
71                     Thread.sleep(500);
72                 }
73             } catch (InterruptedException e) {}
74         }
75     };
76
77     this.blinkThread.start();
78 }
79
80 /**
81  * Load any resources required by the terminal view to render itself properly.
82  */
```

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

```

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

```

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

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

```
247         }
248
249         this.emojiManager.resetState();
250     }
251
252     // Draw a blinker to indicate the user can type here.
253     // Offset it a bit to not interfere with any text on-screen.
254     if (this.blinkState) {
255         g.setColor(Color.WHITE);
256         g.fillRect(
257             (int) (ox + this.fw * buffer.getPosX() + 1),
258             (int) (oy + this.fh * buffer.getPosY() + this.foffset),
259             (int) this.fw - 2,
260             (int) (this.fh / 8)
261         );
262     }
263
264     // Fire draw event for custom rendering.
265     this.emulator.getEventSystem().emit(new EventDraw(g, ox, oy, this.fw, this.fh));
266 }
267 }
```



```
1  package uk.insrt.coursework.zuul.ui;
2
3  import java.awt.Color;
4  import java.awt.EventQueue;
5  import java.util.concurrent.BlockingQueue;
6  import java.util.concurrent.LinkedBlockingQueue;
7
8  import uk.insrt.coursework.zuul.events.EventSystem;
9  import uk.insrt.coursework.zuul.io.IOSystem;
10
11  /**
12   * A terminal emulator which implements an IO system to
13   * be arbitrarily plugged into any existing components.
14   */
15  public class TerminalEmulator implements IOSystem {
16      public static final int TERMINAL_WIDTH = 80;
17      public static final int TERMINAL_HEIGHT = 25;
18
19      private BlockingQueue<String> queue;
20      private EventSystem eventSystem;
21      private JTerminalFrame frame;
22      private boolean fullscreen;
23      private TextBuffer buffer;
24      private String input;
25
26      /**
27       * Construct and build a new TerminalEmulator
28       * @param fullscreen Whether to launch the emulator in fullscreen
29       */
30      public TerminalEmulator(boolean fullscreen) {
31          this.queue = new LinkedBlockingQueue<>();
32          this.eventSystem = new EventSystem();
33          this.buffer = new TextBuffer(TERMINAL_WIDTH, TERMINAL_HEIGHT);
34          this.fullscreen = fullscreen;
35          this.input = new String();
36
37          this.buildFrame();
38      }
39
40      /**
41       * Construct a new TerminalEmulator and force windowed mode.
```

```
42     */
43     public TerminalEmulator() {
44         this(false);
45     }
46
47     /**
48      * Build and show the terminal emulator.
49      */
50     public void buildFrame() {
51         var emulator = this;
52         EventQueue.invokeLater(new Runnable() {
53             @Override
54             public void run() {
55                 emulator.frame = new JTerminalFrame(emulator);
56             }
57         });
58     }
59
60     /**
61      * Get the local event system for this emulator.
62      * @return The event system
63      */
64     public EventSystem getEventSystem() {
65         return this.eventSystem;
66     }
67
68     /**
69      * Get the emulator's text buffer.
70      * @return The text buffer
71      */
72     public TextBuffer getBuffer() {
73         return this.buffer;
74     }
75
76     /**
77      * Tell the terminal frame to repaint contents.
78      */
79     private void repaint() {
80         if (this.frame != null) {
81             this.frame.repaint();
82         }
83     }
```

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

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

```
1  package uk.insrt.coursework.zuul.ui;
2
3  import java.awt.Color;
4  import java.util.regex.Matcher;
5
6  import uk.insrt.coursework.zuul.io.Ansi;
7
8  /**
9   * Representation of a text and colour buffer.
10  * Provides various utilities for manipulating text on screen.
11  */
12  public class TextBuffer {
13      private char[][] buffer;
14      private Color[][] bufferBg;
15      private Color[][] bufferFg;
16
17      private int width;
18      private int height;
19
20      private int posX;
21      private int posY;
22
23      private Color bg;
24      private Color fg;
25
26      private boolean overflow;
27
28      /**
29       * Construct a new TextBuffer with given constraints.
30       * @param width Buffer width
31       * @param height Buffer height
32       */
33      public TextBuffer(int width, int height) {
34          this.buffer = new char[height][width];
35          this.bufferBg = new Color[height][width];
36          this.bufferFg = new Color[height][width];
37
38          this.width = width;
39          this.height = height;
40
41          this.posX = 0;
```

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

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

```
124
125     if (this.posY == this.height - 1) {
126         this.shift();
127     } else {
128         this.posY++;
129     }
130
131     // Set a flag to say we just naturally overflowed and to ignore
132     // the next newline character that may appear.
133     this.overflow = true;
134 }
135
136
137 /**
138  * Write a string value to the text buffer.
139  * @param value String value to write
140  */
141 public void write(String value) {
142     // Write each character sequentially.
143     for (int i=0;i<value.length();i++) {
144         char c = value.charAt(i);
145
146         // If we encounter an Ansi escape character, then take the
147         // substring from this point on and determine if it is a valid
148         // escape code. If it is, apply any changes before continuing.
149         if (c == '\u001B') {
150             Matcher matcher = Ansi.AnsiPattern.matcher(value.substring(i));
151             if (matcher.find()) {
152                 int v = Integer.parseInt(matcher.group(1));
153                 i += 3 + (v > 9 ? 1 : 0);
154
155                 if (v == 0) {
156                     this.bg = Color.BLACK;
157                     this.fg = Color.WHITE;
158                 } else if (v >= 30 && v < 38) {
159                     this.fg = Ansi.fromEscapeCode(v);
160                 } else if (v >= 40 && v < 48) {
161                     this.bg = Ansi.fromEscapeCode(v);
162                 }
163
164                 continue;
165             }
166         }
167     }
168 }
```



```
165         }
166     }
167
168     this.write(c);
169 }
170
171
172 /**
173  * Get a character at a certain position
174  * @param x X position
175  * @param y Y position
176  * @return Character at given position
177  */
178 public char getChar(int x, int y) {
179     return this.buffer[y][x];
180 }
181
182 /**
183  * Get the background colour at a certain position
184  * @param x X position
185  * @param y Y position
186  * @return Background colour at given position
187  */
188 public Color getBg(int x, int y) {
189     return this.bufferBg[y][x];
190 }
191
192 /**
193  * Get the foreground colour at a certain position
194  * @param x X position
195  * @param y Y position
196  * @return Foreground colour at given position
197  */
198 public Color getFg(int x, int y) {
199     return this.bufferFg[y][x];
200 }
201
202 /**
203  * Get the width of this buffer.
204  * @return Width
205  */
```

```
206     public int getWidth() {
207         return this.width;
208     }
209
210     /**
211      * Get the height of this buffer.
212      * @return Height
213      */
214     public int getHeight() {
215         return this.height;
216     }
217
218     /**
219      * Get the current X position of the cursor.
220      * @return X position of cursor
221      */
222     public int getPosX() {
223         return this.posX;
224     }
225
226     /**
227      * Get the current Y position of the cursor.
228      * @return Y position of cursor
229      */
230     public int getPosY() {
231         return this.posY;
232     }
233 }
```

```
1  package uk.insrt.coursework.zuul.util;
2
3  import java.lang.reflect.Field;
4  import java.net.URLClassLoader;
5  import java.util.Vector;
6
7  /**
8   * Utilities for detecting we are running in BlueJ.
9   *
10  * @author Paul Makles <https://insrt.uk>
11  * @version 2.0
12  */
13  public class BlueJ {
14      /**
15       * Whether to ignore deprecation warnings.
16       * Enable to allow isRunningInBlueJ() to confidently determine status.
17       */
18      private static boolean liveOnTheEdge = false;
19
20      /**
21       * Check whether this is being exported as BlueJ using maven-bluej
22       * https://github.com/KCLOSS/maven-bluej
23       * @return Whether this was exported as a BlueJ project.
24       */
25      public static boolean isExportedAsBlueJ() {
26          return BlueJ.class.getResource("/ThisIsABlueJProject") != null;
27      }
28
29      /**
30       * Detect whether we are currently running under BlueJ.
31       * @return Whether we are running from BlueJ.
32       */
33      public static boolean isRunningInBlueJ() {
34          ClassLoader classLoader = Thread.currentThread().getContextClassLoader();
35
36          // When we load the project typically, i.e. from a JAR file, it is instead
37          // loaded by jdk.internal.loader.ClassLoaders$PlatformClassLoader and then
38          // $AppClassLoader, which we should also see further up the chain from the
39          // java.net.URLClassLoader loader.
40          if (classLoader instanceof URLClassLoader) {
41              if (getJavaVersion() > 8 && !liveOnTheEdge) {
```

```

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     return false;
68 }
69
70 /**
71  * Gets the current Java version as a single integer.
72  * Taken from https://stackoverflow.com/a/2591122
73  * @return Current Java major version number.
74  */
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 }
```

```
1  package uk.insrt.coursework.zuul.util;
2
3  import java.io.IOException;
4  import java.io.InputStream;
5  import java.util.Arrays;
6  import java.util.HashMap;
7  import java.util.Map;
8  import java.util.stream.Collectors;
9
10 import com.moandjiezana.toml.Toml;
11
12 /**
13  * This class provides localisation capabilities for the game.
14  */
15 public class Localisation {
16     private Map<String, Object> map;
17
18     /**
19      * Construct a new instance of Localisation.
20      */
21     public Localisation() {
22         this.map = new HashMap<>();
23     }
24
25     /**
26      * Load a certain locale by name.
27      * @param locale The target locale to load
28      * @throws IOException if the locale does not exist in resources
29      */
30     public void loadLocale(String locale) throws IOException {
31         InputStream stream = this
32             .getClass()
33             .getResourceAsStream("/locale/" + locale + ".toml");
34
35         this.map = new Toml().read(stream).toMap();
36     }
37
38     /**
39      * Given a path of keys, find the value at the end of the path.
40      *
41      * Unchecked errors are suppressed as they would only occur if the
```

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

```
1  package uk.insrt.coursework.zuul.util;
2
3  import java.util.Arrays;
4
5  import uk.insrt.coursework.zuul.entities.Entity;
6
7  /**
8   * Utilities for searching through data structures related to the game
9   */
10 public class Search {
11     /**
12      * Find an Entity within an Iterable of entities given certain parameters.
13      * @param entities Iterable of Entities which we search through
14      * @param name Query which we are matching for
15      * @param fuzzy Whether to match whether the alias contains this name in contrast to just doing exact matching
16      * @return The Entity if it is found or null
17      */
18     public static Entity findEntity(Iterable<Entity> entities, String name, boolean fuzzy) {
19         String normalised = name.toLowerCase();
20         for (Entity entity : entities) {
21             String[] aliases = entity.getAliases();
22             for (String alias : aliases) {
23                 if (fuzzy) {
24                     if (Arrays.asList(normalised.split("\\s"))
25                         .contains(alias)) {
26                         return entity;
27                     }
28                 } else if (normalised.equals(alias)) {
29                     return entity;
30                 }
31             }
32         }
33
34         return null;
35     }
36 }
```



```
1  package uk.insrt.coursework.zuul.util;
2
3  import java.util.HashMap;
4  import java.util.List;
5
6  /**
7   * This is an implementation of a Tree-like data structure.
8   * Each node has a one or more children identified by a key
9   * and each node can have more children or have a value.
10  */
11  public class Tree<K, V> {
12      private HashMap<K, Tree<K, V>> children = new HashMap<>();
13      private Tree<K, V> parent;
14      private V value;
15
16      /**
17       * Construct a new Empty Tree node.
18       */
19      public Tree() {}
20
21      /**
22       * Construct a new Tree node with a parent only.
23       * @param parent Tree node which owns this node
24       */
25      public Tree(Tree<K, V> parent) {
26          this.parent = parent;
27      }
28
29      /**
30       * Construct a new Tree node with parent and value.
31       * @param parent Tree node which owns this node
32       * @param value The value this node should hold
33       */
34      public Tree(Tree<K, V> parent, V value) {
35          this.parent = parent;
36          this.value = value;
37      }
38
39      /**
40       * Get a child of this Tree node with a given key K.
41       * @param key Given key
```

```
42     * @return The child represented by this key if it exists, otherwise returns null
43     */
44     public Tree<K, V> getChild(K key) {
45         return this.children.get(key);
46     }
47
48     /**
49     * Private method used to accumulate the edges travelled up to the root node.
50     * @param acc Accumulator value
51     * @return The current accumulator value
52     */
53     private int getHeight(int acc) {
54         if (this.parent == null) return acc;
55         return this.parent.getHeight(++acc);
56     }
57
58     /**
59     * The height of the Tree from this point.
60     * This is the number of edges to get from this node to the root node.
61     * @return The height of the tree
62     */
63     public int getHeight() {
64         return this.getHeight(0);
65     }
66
67     /**
68     * Whether this Tree node has a value.
69     * @return True if this node has a value
70     */
71     public boolean hasValue() {
72         return this.value != null;
73     }
74
75     /**
76     * Get the value of this Tree node.
77     * @return Value stored if there is one, otherwise null.
78     */
79     public V getValue() {
80         return this.value;
81     }
82
```

```
83  /**
84   * Add a child to this Tree node represented by a key K.
85   * @param key Key to represent this new child
86   * @param node Child node to add
87   */
88  public void addChild(K key, Tree<K, V> node) {
89      this.children.put(key, node);
90  }
91
92  /**
93   * Recurse through a given key path and add value as a node at the bottom of the path.
94   * @param keys Keys to iterate through
95   * @param value Value to add at the end of the path
96   */
97  public void addChildWithPath(List<K> keys, V value) {
98      Tree<K, V> node = this;
99      while (keys.size() > 0) {
100         K key = keys.remove(0);
101         Tree<K, V> child = node.getChild(key);
102         if (child == null) {
103             child = new Tree<>(node, keys.size() == 0 ? value : null);
104             node.addChild(key, child);
105         }
106
107         node = child;
108     }
109 }
110 }
```

```
1  package uk.insrt.coursework.zuul.world;
2
3  import java.util.Arrays;
4  import java.util.List;
5
6  /**
7   * Enum which represents a Cardinal direction.
8   */
9  public enum Direction {
10     NORTH(new String[] { "N" }),
11     NORTH_EAST(new String[] { "NE", "NORTH EAST" }),
12     EAST(new String[] { "E" }),
13     SOUTH_EAST(new String[] { "SE", "SOUTH EAST" }),
14     SOUTH(new String[] { "S" }),
15     SOUTH_WEST(new String[] { "SW", "SOUTH WEST" }),
16     WEST(new String[] { "W" }),
17     NORTH_WEST(new String[] { "NW", "NORTH WEST" }),
18
19     UP(new String[] {}),
20     DOWN(new String[] {});
21
22     private List<String> aliases;
23
24     /**
25      * Consturct a new Direction
26      * @param aliases Alternative ways to refer to this Direction
27      */
28     private Direction(String[] aliases) {
29         this.aliases = Arrays.asList(aliases);
30     }
31
32     /**
33      * Check whether this Direction matches the given aliases.
34      * @param direction Direction in String format
35      * @return Whether it matches.
36      */
37     private boolean matches(String direction) {
38         return this.aliases.contains(direction);
39     }
40
41     /**
```

```
42     * Flip a given Direction in the opposite direction.
43     * @return Direction in the opposite direction.
44     */
45     public Direction flip() {
46         switch (this) {
47             default:
48                 case NORTH: return Direction.SOUTH;
49                 case NORTH_EAST: return Direction.SOUTH_WEST;
50                 case EAST: return Direction.WEST;
51                 case SOUTH_EAST: return Direction.NORTH_WEST;
52                 case SOUTH: return Direction.NORTH;
53                 case SOUTH_WEST: return Direction.NORTH_EAST;
54                 case WEST: return Direction.EAST;
55                 case NORTH_WEST: return Direction.SOUTH_EAST;
56                 case UP: return Direction.DOWN;
57                 case DOWN: return Direction.UP;
58         }
59     }
60
61     /**
62     * Convert an arbitrary String to a Direction.
63     * @param direction Raw string representing a Direction
64     * @return Direction or null from given string
65     */
66     public static Direction fromString(String direction) {
67         if (direction == null) return null;
68
69         String directionFormatted = direction.toUpperCase();
70         try {
71             return Direction.valueOf(directionFormatted);
72         } catch (Exception ex) {
73             for (Direction dir : Direction.values()) {
74                 if (dir.matches(directionFormatted)) {
75                     return dir;
76                 }
77             }
78
79             return null;
80         }
81     }
82 }
```



```
1  package uk.insrt.coursework.zuul.world;
2
3  import uk.insrt.coursework.zuul.entities.Inventory;
4
5  /**
6   * Representation of a physical location in the world,
7   * whether it is a room or inventory or neither but not both.
8   */
9  public class Location {
10     private Room room;
11     private Inventory inventory;
12
13     /**
14      * Construct a new Location outside of the World.
15      */
16     public Location() {}
17
18     /**
19      * Construct a new Location pointing to a Room.
20      * @param room Room
21      */
22     public Location(Room room) {
23         this.room = room;
24     }
25
26     /**
27      * Construct a new Location pointing to an Inventory.
28      * @param inventory Inventory
29      */
30     public Location(Inventory inventory) {
31         this.inventory = inventory;
32     }
33
34     /**
35      * Change this Location to point to a Room.
36      * @param room Room
37      */
38     public void setLocation(Room room) {
39         this.room = room;
40         this.inventory = null;
41     }
```

```
42
43  /**
44   * Change this Location to point to an Inventory.
45   * @param inventory Inventory
46   */
47  public void setLocation(Inventory inventory) {
48      this.room = null;
49      this.inventory = inventory;
50  }
51
52  /**
53   * Reset the Location and put us outside of the World.
54   */
55  public void clear() {
56      this.room = null;
57      this.inventory = null;
58  }
59
60  /**
61   * Get the current Room this Location represents.
62   * @return Room or null if in an inventory or out of the World.
63   */
64  public Room getRoom() {
65      return this.room;
66  }
67
68  /**
69   * Get the current Inventory this Location represents.
70   * @return Inventory or null if in a room or out of this World.
71   */
72  public Inventory getInventory() {
73      return this.inventory;
74  }
75  }
```



```
1  package uk.insrt.coursework.zuul.world;
2
3  import java.util.HashMap;
4  import java.util.Set;
5
6  /**
7   * Representation of a Room within the World.
8   * Handles how entities can move from this to other Rooms.
9   */
10 public abstract class Room {
11     private World world;
12     private String name;
13     private HashMap<Direction, Room> adjacentRooms;
14
15     /**
16      * Construct a new Room in a given World with a given name.
17      * @param world World
18      * @param name Internal name used to refer to this Room
19      */
20     public Room(World world, String name) {
21         this.world = world;
22         this.name = name;
23         this.adjacentRooms = new HashMap<>();
24     }
25
26     /**
27      * Get the World that this Room is in.
28      * @return World
29      */
30     public World getWorld() {
31         return this.world;
32     }
33
34     /**
35      * Get the internal name of this Room.
36      * @return Internal name
37      */
38     public String getName() {
39         return this.name;
40     }
41 }
```

```
42  /**
43   * Make another Room adjacent to this Room in a particular Direction.
44   * @param direction Direction the other Room is in
45   * @param room Room which we are making adjacent
46   */
47  public void setAdjacent(Direction direction, Room room) {
48      if (room == null) System.err.println("Warning: assigned null Room to direction " + direction + " for the Room " +
this.name);
49      this.adjacentRooms.put(direction, room);
50  }
51
52  /**
53   * Get an adjacent Room in a particular Direction.
54   * @param direction Direction to look at
55   * @return The Room if one is present in that Direction, otherwise null
56   */
57  public Room getAdjacent(Direction direction) {
58      return this.adjacentRooms.get(direction);
59  }
60
61  /**
62   * Whether the player can leave in any particular direction.
63   * Should print reason if not.
64   * @param direction Direction which we are checking
65   * @return Whether the player can leave
66   */
67  public boolean canLeave(Direction direction) {
68      return true;
69  }
70
71  /**
72   * Get Directions that you can leave this Room in.
73   * @return Set of Directions we can leave in
74   */
75  public Set<Direction> getDirections() {
76      return this.adjacentRooms.keySet();
77  }
78
79  /**
80   * Check whether there is an exit in a particular Direction.
81   * @param direction Direction to check
```

```
82     * @return True if there is an exit in a given Direction
83     */
84     public boolean hasExit(Direction direction) {
85         return this.adjacentRooms.containsKey(direction);
86     }
87
88     /**
89     * Reset adjacent Rooms and reconfigure adjacent Rooms.
90     * This should be called after all Rooms have been spawned into the World.
91     */
92     public void linkRooms() {
93         this.adjacentRooms.clear();
94         this.setupDirections();
95     }
96
97     /**
98     * Spawn Entities in this World.
99     * By default, nothing is done but this should be used further up to spawn
100    * the Entities for this particular Room.
101    */
102    public void spawnEntities() {}
103
104    /**
105    * Convert this Room into a Location.
106    * @return Location representation of Room
107    */
108    public Location toLocation() {
109        return new Location(this);
110    }
111
112    /**
113    * Describe what this Room looks like.
114    * @return Description of this Room
115    */
116    public abstract String describe();
117
118    /**
119    * Setup adjacent Rooms.
120    */
121    protected abstract void setupDirections();
122 }
```



```
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 public class World {
21     protected Map<String, Room> rooms = new HashMap<>();
22     protected Map<String, Entity> entities = new HashMap<>();
23     protected EntityPlayer player;
24
25     protected IOSystem io;
26     protected EventSystem eventSystem;
27
28     /**
29      * Construct a new game World with a given IO system.
30      * @param io IO system to provide to everything
31      */
32     public World(IOSystem io) {
33         this.io = io;
34         this.eventSystem = new EventSystem();
35         this.player = new EntityPlayer(this);
36         this.entities.put("player", this.player);
37     }
38
39     /**
40      * Find an Entity by its ID.
41      * @param id Entity ID
```

```
42     * @return Entity if it exists, otherwise null.
43     */
44     public Entity getEntity(String id) {
45         return this.entities.get(id);
46     }
47
48     /**
49     * Find an Room by its ID.
50     * @param room Room ID
51     * @return Room if it exists, otherwise null.
52     */
53     public Room getRoom(String room) {
54         return this.rooms.get(room);
55     }
56
57     /**
58     * Get the Player entity.
59     * @return The player entity
60     */
61     public EntityPlayer getPlayer() {
62         return this.player;
63     }
64
65     /**
66     * Get the IO system provided to this World.
67     * @return IO system
68     */
69     public IOSystem getIO() {
70         return this.io;
71     }
72
73     /**
74     * Get this World's event system.
75     * @return World event system
76     */
77     public EventSystem getEventSystem() {
78         return this.eventSystem;
79     }
80
81     /**
82     * Add a Room to this World.
```

```
83     * @param room Room to add
84     */
85     protected void addRoom(Room room) {
86         this.rooms.put(room.getName(), room);
87     }
88
89     /**
90     * Spawn a new Entity in the World.
91     * @param id Unique Entity ID
92     * @param entity Entity to spawn
93     */
94     public void spawnEntity(String id, Entity entity) {
95         this.entities.put(id, entity);
96     }
97
98     /**
99     * Get all the Entities found in a given Room.
100    * @param room Room to search for
101    * @return List of Entities in the World in a given Room
102    */
103    public List<Entity> getEntitiesInRoom(Room room) {
104        return this
105            .entities
106            .values()
107            .stream()
108            .filter(e -> e.getRoom() == room)
109            .collect(Collectors.toList());
110    }
111
112    protected void linkRooms() {
113        for (Room room : this.rooms.values()) {
114            room.linkRooms();
115        }
116    }
117
118    public void emit(Event event) {
119        this.eventSystem.emit(event);
120    }
121
122    /**
123    * Try to spawn the player in the first available room.
```

```
124     */
125     public void spawnPlayer() {
126         this.player.setLocation(this.rooms.values().iterator().next());
127     }
128 }
```



```
1  emojis = [  
2    { unicode = "\u1F633", path = "flosh.png" }, # 😬  
3    { unicode = "\u1F601", path = "trol.png" }, # 😏  
4    { unicode = "\u1F438", path = "monkaStare.png" }, # 🤪  
5    { unicode = "\u1F642", path = "pauseChamp.png" }, # 😬  
6    { unicode = "\u1F610", path = "weirdChamp.png" }, # 😬  
7    { unicode = "\u1F604", path = "peepoHappy.png" }, # 😏  
8    { unicode = "\u1F99D", path = "they is stuck.jpg", width = 24, height = 8 }, # 🐱  
9  
10   # Example definitions:  
11   #{ unicode = "[wideChamp]", path = "weirdChamp.png" },  
12   #{ unicode = "[widePeepo]", path = "peepoHappy.png" },  
13   #{ unicode = "😏", path = "peepoHappy.png", width = 8, height = 4 },  
14 ]
```

```
1  # Translations for World of Deez
2  # By Infi
3
4  [global]
5  sight = "Du kannst Folgendes sehen:"
6
7      [global.can_go_in_x_directions]
8      1 = "Du kannst in"
9      2 = "Richtungen gehen"
10
11  [selectors]
12  direction = "<Richtung>"
13  something = "<Etwas>"
14  someone = "<Jemand>"
15  item = "<Item>"
16
17      [selectors.cant_find]
18      1 = "Du schaust nach"
19      2 = ", aber du findest nichts"
20
21  [commands]
22  unknown = "Keine Ahnung, was du gerade versuchst."
23
24  back = "Zurück zum vorherigen Raum"
25  quit = "Spiel beenden"
26  where_am_i = "Das Zimmer nochmal beschreiben"
27
28      [commands.bag]
29      usage = "Schau in deinen Beutel oder nach dem Inventar von etwas"
30      cant_find = "Was du suchst kann nicht gefunden werden"
31      empty = "Dein Beutel ist leer!"
32      entity_empty = "hat nichts im Inventar"
33      can_carry_kg = "Deine Tragekraft beträgt"
34      are_carrying_kg = "Du trägst"
35      look_in_bag = "Du schaust in deinen Beutel und siehst folgendes:"
36      entity_appears_to_have = "hat anscheinend"
37
38      [commands.drop]
39      usage = "Lege ein Item ab"
40      nothing_specified = "Was willst du ablegen?"
41
```

```
42     [commands.drop.dropped]
43     1 = "Du legst"
44     2 = "ab"
45
46     [commands.give]
47     usage = "Etwas an jemanden geben"
48     nothing_specified = "Was willst du geben?"
49     no_target = "Wem oder in was willst du das geben?"
50     denied_player = "Du kannst dich selbst nicht selber geben \u1F438"
51
52     [commands.give.denied]
53     1 = "Du kannst"
54     2 = "nicht geben an"
55
56     [commands.go]
57     usage = "Gehe in eine bestimmte Richtung"
58     nothing_specified = "Wohin gehst du?"
59
60     [commands.help]
61     usage = "Hilfemenü anzeigen"
62     can_run = "Du kannst die folgenden Befehle ausführen:"
63
64     [commands.pet]
65     usage = "Streichel etwas in deiner Umgebung oder deinem Inventar"
66     nothing_specified = "Was willst du streicheln?"
67     denied = "Du kannst nicht folgendes streicheln:"
68
69     [commands.take]
70     usage = "Nimm ein Item auf"
71     nothing_specified = "Von wem willst du etwas nehmen?"
72     entity_does_not_have_entity = "hat nicht"
73     item_not_specified = "Was willst du nehmen?"
74
75     [commands.take.took]
76     1 = "Du nimmst"
77     2 = "von"
78     3 = "und machst es in deinen Beutel"
79
80     [commands.take.denied]
81     1 = "Du kannst"
82     2 = "nicht nehmen, es ist zu schwer für dich und deinen Beutel"
```

```

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 - Sylvasta 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: "

```

```
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"
```

```

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, schaut 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 55NAN0966PA 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, Sylvesta 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 Sylvesta erschienen ist.
244
245 \u001B[36mKorrespondent\u001B[0m: Ich bin hier, stehe draußen 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 Konflikte 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 drau\u00dfen los?"
296     option_q = "Egal."
297
298     [apartments.receptionist.protestors]
299     dialog = """"\u001B[36mRezeptionist\u001B[0m: Ich wei\u00df nicht viel, aber anscheinend schreien
300 Leute au\u00ferhalb 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 w\u00fcrde
306 jemand in der N\u00e4he 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 drau\u00dfen 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 f\u00fcr ein Gespr\u00e4ch
318 anhalten kannst.
319
320 Es ist generell sehr unruhig im Gebiet, einige Menschen sehen angespannt aus,
321 und andere, als w\u00fcrde die Welt heute enden...
322     '''
323
324     enter = '''
325 Du kommst in die Stadtmitte.
326     '''
327

```

```

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, Sylvasta 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 = "Danke schön."
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
```

```
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.
```

```
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."
```

```
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. Chasing New Horizons"
524     contents = """
525 On July 14, 2015, something amazing happened. More than 3 billion miles from
526 Earth, a small NASA spacecraft called New Horizons screamed past Pluto at more
527 than 32,000 miles per hour, focusing its instruments on the long mysterious icy
528 worlds of the Pluto system, and then, just as quickly, continued on its journey
529 out into the beyond.
530 """
531
```

```

532     [medical_centre_office.books.2] # yes
533     title = "Stack of research papers"
534     contents = ""
535     Upon our latest series of tests, we have found the effect of \u001B[47m\u001B[37m[redacted]\u001B[0m has
536     considerable negative effects. These tests were performed over a series of 8
537     days and caused \u001B[47m\u001B[37m[rctd]\u001B[0m and \u001B[47m\u001B[37m[redact]\u001B[0m in the body.
538
539     This document is confidential and should not be shared with third parties.
540     ""
541
542     [medical_centre_office.books.3]
543     title = "Barry A. Burd. Java For Dummies"
544     contents = ""
545     Java is the platform-independent, object-oriented programming language used for
546     developing web and mobile applications. The revised version offers new
547     functionality and features that have programmers excited, and this popular guide
548     covers them all.
549
550     You will learn how to program good in Java, like this:
551
552     \u001B[36mimport \u001B[33mjava\u001B[37m.\u001B[33mio\u001B[37m.\u001B[33mBufferedReader\u001B[37m;
553     \u001B[36mimport \u001B[33mjava\u001B[37m.\u001B[33mio\u001B[37m.\u001B[33mInputStreamReader\u001B[37m;
554     \u001B[33mpublic class \u001B[31mMain \u001B[34m{
555         \u001B[33mpublic static void \u001B[31mmain \u001B[34m(\u001B[33mString[] args\u001B[34m) \u001B[34m{
556             \u001B[33mtry{ String a=new \u001B[31mBufferedReader \u001B[34m(
557                 \u001B[33mnew \u001B[31mInputStreamReader\u001B[34m( System\u001B[37m.\u001B[33mout\u001B[37m \u001B[34m))
\u001B[37m.
558                 \u001B[31mreadLine() \u001B[37m;
559             \u001B[33mif
560                 \u001B[34m(a == \u001B[32m"hi"\u001B[34m) {
561                 System\u001B[37m.\u001B[33mout\u001B[37m.\u001B[31mprintln\u001B[34m(\u001B[32m>Hello!\u001B[34m)
\u001B[37m;
562             \u001B[34m} \u001B[33melse
563             \u001B[34m{
564                 System\u001B[37m.\u001B[33mout\u001B[37m.\u001B[31mprintln\u001B[34m(\u001B[32m"you did not greet me!\u001B[34m)
34m) \u001B[37m;
565             \u001B[34m}
566         } \u001B[33mcatch \u001B[34m(\u001B[33mException e\u001B[34m) {\u001B[35m/*something bad happen! */ \u001B[34m} } }
567     \u001B[0m""
568
569     [medical_centre_office.books.4] # yes

```

```

570         title = "Dr. Mike Ox. Introducing Impostors in Society"
571         contents = ""
572     [Stamped for internal use only.]
573
574     This book details methods which we can use to manipulate and control the general
575     populous, I will be going into detail about ways we can subtly remove dangerous
576     ideas and to keep your secrets under covers.
577     ""
578
579     [medical_centre_office.books.5] # yes
580     title = "RNA sequences"
581     contents = ""
582     Confidential document
583
584     Figure 1.
585     GAGAATAAACTAGTATTCTTCTGGTCCCCACAGACTCAGAGAGAACCCGCCACCATGTTTCGTGTTCTTGGTGCTGCTGCC
586     TCTGGTGTCCAGCCAGTGTGTGAACCTGACCACCAGAACACAGCTGCCTCCAGCCTACACCAACAGCTTTACCAGAGGCG
587     TGTAACCCCCGACAAGGTGTTTCAGATCCAGCGTGCTGCACTCTACCCAGGACCTGTTCTGCCTTTCTTCAGCAACGTG
588     ACCTGGTTCCACGCCATCCACGTGTCCGGCACCAATGGCACCAAGAGATTCGACAACCCCGTGCTGCCCTTCAACGACGG
589     GGTGTACTTTGCCAGCACCAGAGAAGTCCAACATCATCAGAGGCTGGATCTTCGGCACCACTGGACAGCAAGACCCAGA
590     GCCTGCTGATCGTGAACAACGCCACCAACGTGGTCATCAAAGTGTGCGAGTTCCAGTTCTGCAACGACCCCTTCTGGGC
591     GTCTACTACCACAAGAACAACAAGAGCTGGATGGAAGCGAGTTCCGGGTGTACAGCAGCGCCAACAACCTGCACCTTCGA
592     GTACGTGTCCAGCCTTTCTGATGGACCTGGAAGGCAAGCAGGGCAACTTCAAGAACCTGCGCGAGTTCGTGTTTAAGA
593     ACATCGACGGCTACTTCAAGATCTACAGCAAGCACACCCCTATCAACCTCGTGCGGGATCTGCCTCAGGGCTTCTCTGCT
594     CTGGAACCCCTGGTGGATCTGCCCATCGGCATCAACATCACCCGGTTTCAGACACTGCTGGCCCTGCACAGAAGCTACCT
595     GACACCTGGCGATAGCAGCAGCGGATGGACAGCTGGTGCCGCCGCTTACTATGTGGGCTACCTGCAGCCTAGAACCTTCC
596     TGCTGAAGTACAACGAGAACGGCACCATCACCGACGCCGTGGATTGTGCTCTGGATCCTCTGAGCGAGACAAAGTGCACC
597     ""
598
599     [medical_centre_office.books.6]
600     title = "Jules Verne. Around the World in Eighty Days"
601     contents = ""
602     One ill-fated evening at the Reform Club, Phileas Fogg rashly bets his
603     companions £20,000 that he can travel around the entire globe in just eighty
604     days - and he is determined not to lose. Breaking the well-established routine
605     of his daily life, the reserved Englishman immediately sets off for Dover,
606     accompanied by his hot-blooded French manservant Passepartout. Travelling by
607     train, steamship, sailing boat, sledge and even elephant, they must overcome
608     storms, kidnapping, natural disaster, Sioux attacks and the dogged Inspector Fix
609     of Scotland Yard - who believes that Fogg has robbed the Bank of England - to
610     win the extraordinary wager.

```



```

611  ""
612
613  [marie]
614  description = "\u001B[35mMarie der Nerz\u001B[0m"
615
616      # Purple: \u001B[35m
617      [marie.alley]
618
619          [marie.alley.small_talk]
620          dialog = "\u001B[35mMarie\u001B[0m: Was bringt dich hierher?"
621          option_1 = "Hast du von den Protesten gehört?"
622          option_2 = "Sylvasta hat etwas böses vor..."
623          option_q = "Egal, hab einen schönen Tag."
624
625          [marie.alley.protests]
626          dialog = ""\u001B[35mMarie\u001B[0m: Ja, die von Sylvasta haben auf jeden Fall
627 nichts Gutes auf dem Schirm, aber ich weiß auch nicht ob ich den Demonstranten
628 trauen sollte.""
629          option_1 = "Willst du mir dabei helfen, das herauszufinden?"
630          option_q = "Egal."
631
632          [marie.alley.sylvasta]
633          dialog = "\u001B[35mMarie\u001B[0m: Glaubst du, ich weiß das nicht? Willst du mir helfen, es herauszufinden?"
634          option_1 = "Klar."
635          option_q = "Nein danke."
636
637          [marie.alley.confirm]
638          dialog = "\u001B[35mMarie\u001B[0m: Wir könnten einbrechen und die Wahrheit herausfinden."
639          option_1 = "Lass machen! [bringt die Story weiter]"
640          option_q = "Auf keinen Fall!"
641
642          [marie.alley.recon]
643          dialog = ""\u001B[35mMarie\u001B[0m: Okay, während ich nachdenke, wie wir reinkommen, musst du mir
644 etwas bringen, worüber wir reden müssen, ich bin mir sicher, dass es im Laden etwas gibt, komm wieder hierher
645 wenn du zurückkommst.""
646          option_q = "Verstanden."
647
648          [marie.alley.waiting]
649          dialog = "\u001B[35mMarie\u001B[0m: Na dann? Hol es dir."
650          option_q = "Okay."
651

```

```
652     [marie.alley.mission_brief]
653     dialog = ""\u001B[35mMarie\u001B[0m: Also, der Plan ist, du gehst zum Medizinzentrum,
654 bummelst lässig zur Rezeption, du darfst aber nicht auffallen... Finde dir zum Beispiel einen
655 Stuhl um darauf zu sitzen oder so was. Jedenfalls, wenn du dort bist und denkst dass die
656 Security nicht zuschaut, nutz dein Kommunikationsgerät um mir bescheid zu sagen.
657
658 \u001B[35mMarie\u001B[0m: Alsdann werde ich eine Ablenkung senden und sie in ihr
659 Pausenzimmer führen während du ins Büro schleichst und versuchst, alles was du willst, zu finden...
660 Aber wenn du dort bist, sag mir bescheid.
661
662 \u001B[35mMarie\u001B[0m: Wenn du willst, kannst du das Kommunikationsgerät jetzt testen.""
663     option_q = "Okay."
664
665     [marie.comms]
666     received = ""\u001B[35mMarie\u001B[0m: Alles klar, ich hab die Dokumente gefunden,
667 Ich sende sie an die breiten Massen. Gut gemacht mit dem Einbruch.""
668     bad_documents = "\u001B[35mMarie\u001B[0m: Das sieht nicht wie die richtigen Dokumente aus..."
669     no_access = "\u001B[31mZugriff verweigert!\u001B[0m"
670
671     [marie.comms.orientation]
672     dialog = "\u001B[35mMarie\u001B[0m: Was willst du?"
673     option_1 = "Wohin muss ich nochmal?"
674     option_2 = "Kannst du mich hören?"
675     option_q = "Egal."
676
677     [marie.comms.directions]
678     dialog = ""\u001B[35mMarie\u001B[0m: Du musst ins Medizinzentrum, wenn du dort bist, versuch
679 nicht aufzufallen und sage mir, wenn du in Position bist.""
680     option_q = "Verstanden."
681
682     [marie.comms.hear]
683     dialog = "\u001B[35mMarie\u001B[0m: Laut und deutlich."
684     option_q = "Nett."
685
686     [marie.comms.complaint]
687     dialog = "\u001B[35mMarie\u001B[0m: Versuche, nicht aufzufallen..."
688     option_q = "Okay."
689
690     [marie.comms.in_position]
691     dialog = "\u001B[35mMarie\u001B[0m: Okay, Ich werde eine Ablenkung schaffen, gib mir einen Moment..."
692     option_1 = "[warten]"
```

```
693
694     [marie.comms.distraction]
695     dialog = ""\u001B[31mWachmann\u001B[0m: Wer ist da?
696     Komm raus, oder wir haben ein Problem.
697
698     (Du siehst, wie der Wache in den Pausenraum geht)
699     ""
700     option_1 = "[warten]"
701
702     [marie.comms.coast_is_clear]
703     dialog = "\u001B[35mMarie\u001B[0m: Okay, jetzt ist deine Chance."
704     option_q = "Verstanden."
705
706     [marie.comms.gogogo]
707     dialog = "\u001B[35mMarie\u001B[0m: Auf was wartest du??"
708     option_q = "AAAAAAA"
709
710     [marie.comms.office]
711     dialog = "\u001B[35mMarie\u001B[0m: Was siehst du?"
712     option_1 = "Rumgestreute Dokumente."
713
714     [marie.comms.documents]
715     dialog = ""\u001B[35mMarie\u001B[0m: Ok, schau sie dir durch und finde alles, was relevant
716     ist, achte besonders auf alles, was die Sachen erwähnt, über die demonstriert wird.
717     Wenn du das gemacht hast, solltest du die Dokumente mir hochladen,
718     geh zurück zu deiner Wohnung und wir machen es von dort, sag mir wenn du dort bist.
719
720     \u001B[35mMarie\u001B[0m: Ich schau genau darauf, dass deine Deckung nicht platzt.
721
722     \u001B[33mTipp: Du kannst die Dokumente benutzen (use), um sie zu lesen.\u001B[0m""
723     option_q = "Okay."
724
725     [marie.comms.home]
726     dialog = ""\u001B[35mMarie\u001B[0m: Warscheinlich hast du die Dokumente, schalte deinen PC an,
727     geh auf den Link den ich dir gesendet hab, er sollte dort erscheinen, und lade dann hoch.""
728
729     [stage]
730     reached_conclusion = ""\n
731     Du hast das Spielende erreicht, du bist jetzt in einem Open-World-Modus,
732     erkunde weiter um deine Ergebnisse zu maximieren.
733
```

734 Du kannst endgültig aufhören mit dem Befehl \u001B[47m\u001B[30mwin\u001B[0m
735 ""

```
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]
```

```
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 bad"
82
```

```

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 = ""...

```

Chapter 4.:

The files are released into the internet for anyone to read, people are quick to analyse through every single tiny detail, some immediate details come to light:

- Sylvasta was researching ethically questionable areas of science, in particular, they were running hundreds of tests daily on a variety of beastman test subjects. But nobody could prove anyone was there against their will.
- However, the research did also get immediately picked up by foreign powers who quickly discovered that the beastman society living in the city is a far greater threat than they initially anticipated.
- The city, with Sylvasta's public advice, immediately ordered evacuation of all citizens to any area they could find fearing a potentially deadly conflict on 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."

```

[entities]

```
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
```



```

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 helph 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 55NAN0966PA 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.
```

```
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
```



```

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 GAGAATAAACTAGTATTCTTCTGGTCCCCACAGACTCAGAGAGAACCCGCCACCATGTTCGTGTTCTGGTGCTGCTGCC

```

```

571 TCTGGTGTCCAGCCAGTGTGTGAACCTGACCACCAGAACACAGCTGCCTCCAGCCTACACCAACAGCTTTACCAGAGGCG
572 TGTACTACCCCGACAAGGTGTTTCAGATCCAGCGTGCTGCACTCTACCCAGGACCTGTTCTCCTGCTTTCTTCAGCAACGTG
573 ACCTGGTTCCACGCCATCCACGTGTCCGGCACCAATGGCACCAAGAGATTCGACAACCCCGTGCTGCCCTTCAACGACGG
574 GGTGTACTTTGCCAGCACCAGAGAAGTCCAACATCATCAGAGGCTGGATCTTCGGCACCACTGGACAGCAAGACCCAGA
575 GCCTGCTGATCGTGAACAACGCCACCAACGTGGTCATCAAAGTGTGCGAGTTCCAGTTCTGCAACGACCCCTTCCTGGGC
576 GTCTACTACCACAAGAACAACAAGAGCTGGATGGAAGCGAGTTCCGGGTGTACAGCAGCGCCAACAACCTGCACCTTCGA
577 GTACGTGTCCAGCCTTTCTGATGGACCTGGAAGGCAAGCAGGGCAACTTCAAGAACCTGCGCGAGTTTCGTGTTTAAGA
578 ACATCGACGGCTACTTCAAGATCTACAGCAAGCACACCCCTATCAACCTCGTGCGGGATCTGCCTCAGGGCTTCTCTGCT
579 CTGGAACCCCTGGTGGATCTGCCCATCGGCATCAACATCACCCGGTTTTAGACACTGCTGGCCCTGCACAGAAGCTACCT
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_l = "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_l = "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_l = "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..."

```

```

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 ""

```

```

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      ] }

```

```

42     protestors = { description = "protestors.dialog", options = [
43         { description = "protestors.option_1", to = "leaflet" },
44         { description = "small_talk.option_q", to = "small_talk", mustExit = true }
45     ] }
46     enquire = { description = "enquire.dialog", options = [
47         { description = "protestors.option_1", to = "leaflet", },
48         { description = "small_talk.option_q", to = "small_talk", mustExit = true }
49     ] }
50     leaflet = { description = "leaflet.dialog", options = [
51         { description = "leaflet.option_1", to= "small_talk", mustExit = true }
52     ] }
53     #recon = { description = "testing", options = [
54     #     { description = "sususus!", to = "recon", mustExit = true }
55     #] }
56
57 # 3. Old Man located in Forest
58 [npc_old_man]
59     _prefix = "forest.old_man."
60     _start = "small_talk"
61     small_talk = { description = "small_talk.dialog", options = [
62         { description = "small_talk.option_1", to = "request" },
63         { description = "small_talk.option_q", to = "small_talk", mustExit = true }
64     ] }
65     request = { description = "request.dialog", options = [
66         { description = "request.option_1", to = "small_talk", mustExit = true },
67         { description = "request.option_q", to = "small_talk", mustExit = true }
68     ] }
69     praise = { description = "praise.dialog", options = [
70         { description = "praise.option_q", to = "praise", mustExit = true }
71     ] }
72
73 # 4. Protestors located on Street
74 [npc_protestors]
75     _prefix = "street.protestors."
76     _start = "small_talk"
77     small_talk = { description = "small_talk.dialog", options = [
78         { description = "small_talk.option_1", to = "enquire" },
79         { description = "small_talk.option_2", to = "small_talk", mustExit = true }
80     ] }
81     enquire = { description = "enquire.dialog", options = [
82         { description = "enquire.option_1", to = "leaks" },

```



```

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 = [

```

```

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 ] }
134
135 # 8. TV located in Apartments
136 [home_tv]
137     _prefix = "home.tv."
138     _start = "first_on"
139     first_on = { description = "first_on.dialog", options = [
140         { description = "keep_watching", to = "dialog_a" },
141         { description = "off", to = "first_on", mustExit = true }
142     ] }
143     dialog_a = { description = "first_on.dialog_a", options = [
144         { description = "keep_watching", to = "dialog_b" },
145         { description = "off", to = "first_on", mustExit = true }
146     ] }
147     dialog_b = { description = "first_on.dialog_b", options = [
148         { description = "keep_watching", to = "dialog_c" },
149         { description = "off", to = "first_on", mustExit = true }
150     ] }
151     dialog_c = { description = "first_on.dialog_c", options = [
152         { description = "off", to = "first_on", mustExit = true }
153     ] }
154
155 # 9. Communicator device with Marie
156 [comms_marie]
157     _prefix = "marie.comms."
158     _start = "orientation"
159
160     orientation = { description = "orientation.dialog", options = [
161         { description = "orientation.option_1", to = "directions" },
162         { description = "orientation.option_2", to = "hear" },
163         { description = "orientation.option_q", to = "orientation", mustExit = true }
164     ] }

```

```

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 = [

```

```
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   ] }
```

```
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"
```