SoPra of the dead

Playing with fate

Single phase

Table of contents

1 Organizational matters

Note

For the course of the individual phase, please also note the information in the document "Organizational matters", which was made available to you in the CMS at the beginning of the software internship.

Once you have successfully made it through the group phase and, together with your group you have designed, implemented and tested the game, you now need to alone show that you have understood and can implement the concepts of the lecture. In the individual phase, you will design an extension of the game from the group phase, implement and test.

You will receive a new *git* repository in the chair's own *GitLab*[1] , which only you can access. have access on their own. In it, we provide you with an implementation of the game of group phase on which you should build. (By providing our implementation, we ensure that all participants have the same starting point for implementation. mentation, regardless of the group's previous success). We recommend that you use the provided implementation of the group stage game and its design for the group phase. your single-phase project. However, you are of course free to use this draft and the associated implementation as you wish[2] . To make it easier for you to familiarize into our implementation, we have included a description of the design and the implementation and a class diagram, which you can find in the CMS.

The individual phase is conducted entirely online. During the individual phase there are *no* Compulsory attendance, i.e. you can organize your own time and must also be available for not be reachable by anyone. **Group work is prohibited during the individual phase, each part The software system (i.e., design, implementation, testing) must be completely independently.** However, the tutors are still available to you in Office Hours[3] is available to help you with any questions you may have.

The final submission of the individual phase must be made by Friday, 15.10.2021, 23:59 at the latest, into your single-phase repository.

[1] https://sopra.se.cs.uni-saarland.de/

[2] Plagiarism is of course still prohibited, you may only use your own ideas or concepts of your own. Group from the group phase.

[3] from Monday, 04.10.2021, 14:00 hrs. For more details on the office hours, please refer to the terminkalender in the CMS.

1.1 Task

To pass the individual phase, you must create a draft for the following player's route.
the concepts of the lecture and, at the end of the individual phase, prepare a
executable implementation that corresponds to your design. To do this, you must also
Create a document that explains how the concepts from the lecture can be incorporated into your design.
have been incorporated. We provide you with a template in the CMS that allows you to
should serve as a template and example for the form of this document. You must also
write meaningful unit tests and system tests for your single-phase implementation and
pass. Simply passing all the tests is not enough, your code must also pass well.
be structured and follow the concepts of the lecture.

1.2 Draft

You must create a class diagram for your design. You can use the
class diagram provided. In your class diagram, you must
all the classes that you add in your own design, and also all the
classes that you can change or extend yourself. All other classes from the available
design you do not need to include in your class diagram, as long as you have this
unchanged. However, when adding new classes, make sure that
Relationships between classes must be included in the class diagram even if
an unchanged class is affected. (For example, if you have a new class that is affected by a
existing class, which you leave unchanged, the class from which your
inherits a new class will be included in your diagram, even though it is unchanged, to
inheritance relationship in the class diagram).

For the design submission on Friday, 08.10.2021, your design (in the form of a UML-
class diagram in PDF format) must be uploaded to the CMS by 23:59. Likewise
Your project description uploaded separately in the CMS by Friday, 08.10.2021 at 23:59
be.

1.3 Disposal

At the end of the individual phase (Friday, 15.10.2021), your final implementation (incl.
Tests[4]) on the master `branch` of the repository provided for you.
The last commit on the `master branch` that is made by 15.10.2021, 23:59 at the latest counts.
CEST in the repository. An updated version of your
Project description to be uploaded to the CMS. This updated version is intended to improve the
which you need to consider in your final implementation compared to the draft.
have undertaken.

[4]Further information on the expected tests can be found in Tests.

It is your responsibility to check that the push has arrived in the repository and everything is up to date in good time. Your delivery must be registered on our reference platform (*Debian 10* with *OpenJDK 16*).

1.4 Overview

The player extension contains all the details about the player extension that you can find in the individual phase must be implemented. Technical details and notes on implementation in the individual phases can be found in Technical details. Finally, here are some tips on testing your Implementation in tests.

Note

Please note that all information from the group phase specification document is still valid (as long as it does not conflict with the changes in the following chapters).

2 The player extension

Four weeks have passed since we barricaded ourselves in at the university. We unfortunately had to realize that life in this post-apocalyptic world is not quite so easy. is linear, as we originally thought...

2.1 Fate cards

Each round, a new fate card is drawn for each player when it is their turn. Each depending on which actions are carried out in this round, fate is triggered and takes its course. For example, food may disappear, new survivors may appear or zombies spawn, even though you've just felt so safe.
Sometimes, however, players are also spoiled for choice when it comes to fate of the group...

2.1.1 Trigger

One of the following actions can trigger the fate (*Crossroads Event*):

- Creating equipment
- Changing the waste pile
- Searching a (optionally: specific) location
- Move (optional: to a specific location)
- Barricade (optional: a specific location)

The first of these actions that a player performs in an action triggers the round, the fate card. However, this does not happen immediately, but only after the corresponding action of the player is completed. More about the new Commands and events can be found in section 3.3.

2.1.2 Consequences

Each fate card has a consequence. Either exactly one follows through the fate card, or it must be played by all players.

players can vote on two available options. Each player and each player exactly one vote.

Possible consequences are:

(1) The food supply changes (changeFood). If more food is added to the food supply than is available, a hunger marker is placed instead. added.

(2) The morale changes (`changeMoral`). If the morale drops to 0 as a result, the Game *immediately* lost as usual.

(3) Spawn new survivors ()`spawnSurvivors`) with optional field `children`, if as many children as there are new survivors to spawn.

(4) Spawn new zombies (`spawnZombies`). If the optional `locationId` is not given zombies will spawn at *each* location in the `amount` specified.

(5) Vote on two consequences (`choice`).

For all consequences (with the exception of `choice`), the configuration file specifies how the corresponding value changes (`amount`).

For example, the consequence `"changeFood"` with `"amount: 2"` means that the close food supply, two food markers are added, whereas with `"amount: -2"` two food markers would be removed.

The rule is that a player's action is always processed *completely* by the server first, before fate is handled. In particular, if a character is killed by dies when performing an action, but the fate is still handled normally afterwards. If the consequence indicated on the fate card cannot be realized, the card will not be played. nothing.

The new commands and events that can be sent and received via the fate cards are specified in 3.3.

***Example:* Triggering a fate card with foodChange**

There are still three food markers in the food supply. A fate card is drawn for the player Jonathan, which has the trigger *Create equipment* with the consequence `changefood` at -2.

Of course, Jonathan has no way of knowing this, so he unsuspectingly plays an equipment card first and puts `snowboots` on his survivor Carla. The action of *putting on equipment* is thus completed and fate takes its course.

→ The server sends the Crossroad `event` to the player and the food supply is reduced by 2, the corresponding events are sent and then the player waits for the next command from Jonathan.

3 Technical details

This chapter describes the technical details required for your implementation.
are relevant.

3.1 Coincidence

The fate cards are shuffled after the crisis cards at the start of the game.
This is done with the following call:

```
Collections.shuffle(crossroadCards, random);
```

The first card from the shuffled pile is always drawn during the game. If
a player does not perform an action in a round that would trigger his fate card,
it simply expires.

3.2 Configuration file

Sufficient fate cards must be defined in the configuration file:

$$\#CrossroadsCards \geq maxPlayers * numRounds$$

Incomplete example configuration file:

```
{
  ...
  "crossroads":[
    {
      "barricaded":{
        "identifier":1,
        "locationId":42,
        "consequence":{
          "change Food":{
            "amount":-2
          }
        }
```

```
      }
    }
,
    { "moved":{
        "identifier":3
        ,"locationId"
        :1,"consequence
        ":{
          "spawnSurvivors"
            :{"amount":2,
            "children":true
          }
        }
      }
    }
,
    { "searched":{
        "identifier":6
        ,"locationId"
        :2,"consequence
        ":{
          "spawn Zombies":
            {"amount":1,
            "locationId":42
          }
        }
    } }
,
    {
      "waste Changed":{
        "identifier":7
        ,"amount":2,
        "consequence":{
          "choice":{
            "consequence 1":{
              "spawn Zombies":
                {"amount":2,
                "locationId":42
              }
            },
            "consequence 2":
              {"change Food"
              :{
                "amount":-3
              }
            }
```

```
}
```

```
            }
          }
        },
        {
          "equip":{
            "identifier":8,
            "consequence":{
              "change Moral":{
                "amount":-1
              }
            }
          }
        },
        {
          "barricaded":{
            "identifier":10,
            "consequence":{
              "change Moral":{
                "amount":2
              }
            }
          }
        }
      ],
      ...
}
```

3.3 New/changed commands and events

Table 1: New events

New events in the individual phase
`Crossroad(int crossroadId)` With this event, the server informs the client that the fate has been triggered. The `crossroadId` specifies the ID of the fate card. This event is sent after the action that triggered the fate card has been fully processed by the server.
`VoteNow()` With this event, the server informs the client that it must vote for one of the consequences. The VoteNow() is to be understood like an Act- Now(): it is always sent to the player (in ascending order of `playerId`) from whom a vote is expected next.
`VoteResult(boolean answer)` With this event, the server informs the client which of the two consequences has won the vote. The selection of the first consequence offered corresponds to the parameter `answer = True`. If there is a tie, the first consequence offered is always selected.

Table 2: New commands

New commands
`Vote(boolean anwswer)` With this command, the player informs the server which of the two offered consequences is selected.

4 tests

This chapter deals with the tests required in the individual phase of the software development process.
Internship.

- You must test your own implementation in a meaningful way and pass your own tests.
  tion. Write meaningful unit and integration tests for the relevant parts
  your implementation of the individual phase. We will also (similar to the
  Group phase) `provide` a framework for system testing that allows you to
  can create system tests.
- Your implementation must pass system tests created by us (which we will provide in re-
  at regular intervals on your implementation).