C++ Test - Simple *Zork* Game Assignment

Overview

To test our knowledge on C/C++ we will code a simple text mode conversational game. We should be aiming for a well organized code (efficiency is not important right now) where we explore C++ capabilities like polymorphism.

Submission rules

The delivery must be a link into **github.org** repository inside the folder *"Zork":*

1. A build in the "Release" section of your repository, with a **zip** containing the code compiled in Release with the minimal amount of files required to play.
2. In the repository, your code with well commented and small **commits** on your changes over time.
3. There must be a text file called "README.md" containing info about the game, authors, a link to the github repository and *license*. Should also contain a **guide in how to play and finish the game**.

The **link** must be submitted in **seven days** after receiving this assignment.

Grading Criteria

To accept a submission for grading, it must comply with:

1. It followed the submission rules stated above.
2. The code is created on Visual Studio 2019 Community with minimal level 3 warnings.
3. The game does not crash on the main player path.
4. The game content is at least:
    a. Three connected rooms (aim for 5+)
    b. Items in the rooms that can be picked up
    c. Being able to drop items
    d. Items can be placed inside other items
    e. Extra feature (e. g. creatures that walk around, RPG stats in player and items, exits that can be closed and only open with keys, real-time combat, NPC dialog, etc. be creative!)
5. The example code can be used as a guide that implements the proposed class structure but you must do your own code. Please, don't copy the example.
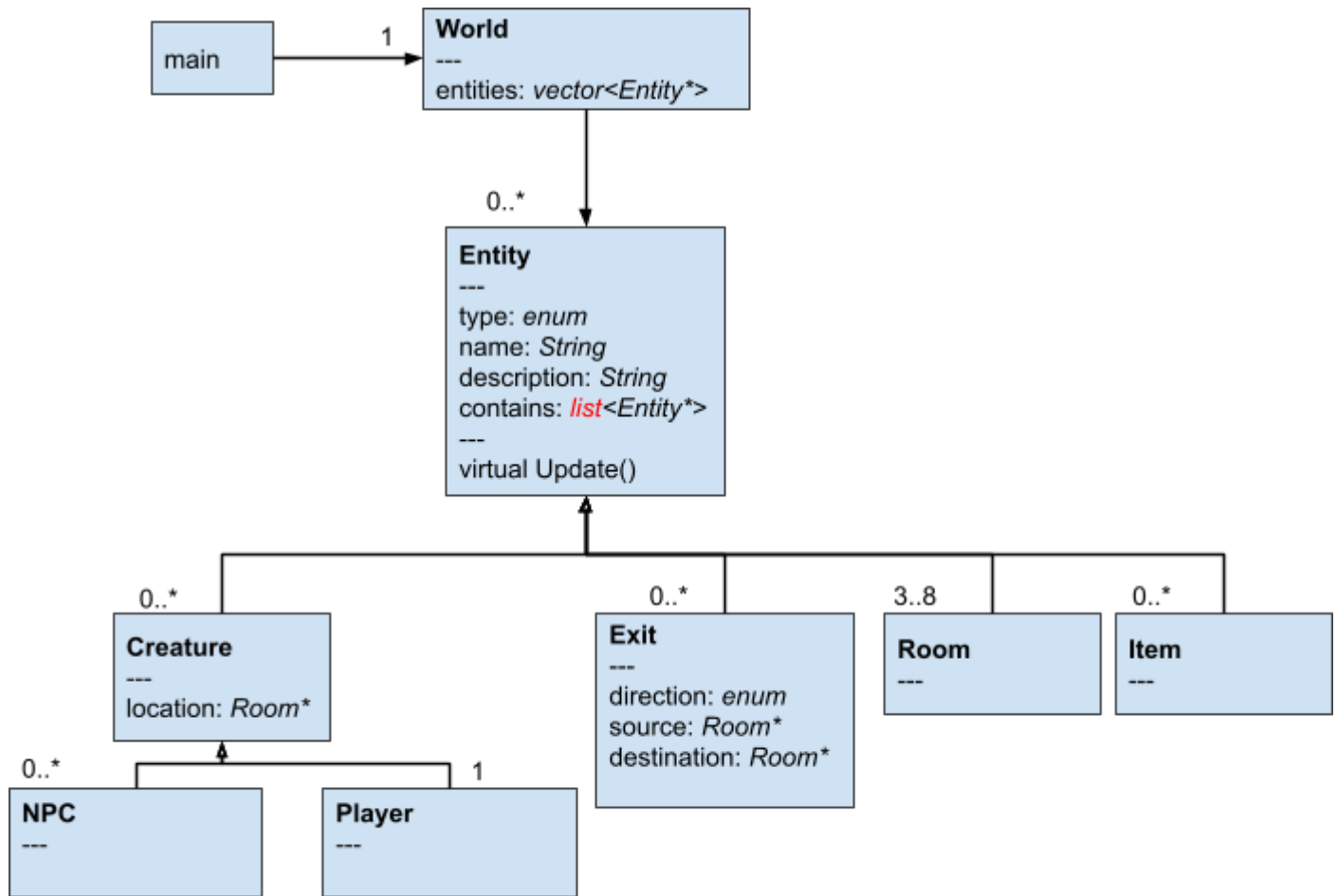
Once accepted the criteria is as follows:

- 30%: Const-correctness, correct use of references.
- 20%: Coding style consistency, healthy commits.
- 50%: Simple but well structured C++ code. Correct use of containers.

References

- Play original Zork! Here (try killing the troll)
- Zork! map here (have it open while play)
- Example code following this structure can be found here

Structure Guide

```
┌─────────┐   1    ┌──────────────────────────────┐
│  main   │───────▶│ World                        │
└─────────┘        │ ---                          │
                   │ entities: vector<Entity*>    │
                   └──────────────────────────────┘
                                  │
                          0..*    │
                                  ▼
                   ┌──────────────────────────────┐
                   │ Entity                       │
                   │ ---                          │
                   │ type: enum                   │
                   │ name: String                 │
                   │ description: String          │
                   │ contains: list<Entity*>      │
                   │ ---                          │
                   │ virtual Update()             │
                   └──────────────────────────────┘
```

World
---
entities: *vector<Entity*>*

Entity
---
type: *enum*
name: *String*
description: *String*
contains: *list<Entity*>*
---
virtual Update()

0..*

Creature
---
location: *Room**

0..*

NPC

1

Player

0..*

Exit
---
direction: *enum*
source: *Room**
destination: *Room**

3..8

Room

0..*

Item

In the README feel free to explain anything you feel relevant for us to know, like what you struggled with or how you approached your unique feature.

Remember to have fun!