CPSC 2720 [Fall]

The Cube

Team Juggernaut

Jordan Harris
Mackenzie Kure
Cassandra Olfert
Scott Sonnleitner

December 2, 2020

Contents

INTRODUCTION 3
PROJECT MANAGEMENT 3
TEAM ORGANIZATION 3
RISK MANAGEMENT 4
CODE REVIEW PROCESS 6
COMMUNICATION TOOLS 7
CHANGE MANAGEMENT 7
SOFTWARE DESIGN 8
APPENDIX A: CLASS DIAGRAMS 9
APPENDIX B: SEQUENCE DIAGRAMS 13

INTRODUCTION

The Cube:
This is a solo game in which the user will traverse an ominous cube in search for the exit, all the while solving riddles, puzzles, and dealing with the fellow residents of the cube all of which aid the player in escaping. There will be a series of nine rooms of three different types all each containing their own puzzle and objects that will aid the user in escaping. If the user succeeds in completing all puzzles and collects all items necessary for escape the player will have won!
For one complete round of the game the user will have to traverse each of the nine rooms, talking to NPC's to gain hints, solving puzzles to gain information or to open doors to other rooms, and collecting objects that will all aid in the user in escaping The Cube. By solving all puzzles without failing, the user will escape, otherwise remaining trapped within the cube until their inevitable end. Once all the puzzles, one in each room, are solved, the player wins.

The team members will cycle through who is team lead, librarian, QA leader, and floater. Seeing as there are only three sections of the project in which these roles are necessary, each member will not take a turn in each role. Our team did joint risk assessment of any future problems that we suspected may occur over the duration of the project. In this document the team has laid out a developmental process in that the work will be divided evenly, in small increments with early deadlines. It was decided that no two people will work on the same area of the program, to avoid merge conflicts, unless deemed necessary by the team lead. The issue of dealing with pull requests, merge conflicts, and pushing up non-compliable code into the repo are discussed at length in accordance with the goal of not having these issues stop the progress of the project. It was decided that bug reports will be dealt with by the QA lead, in accordance with the team lead's instructions. The communication tool chosen by the team is MS Teams: this is where most all essential communication will occur, including weekly team meetings. Lastly, the design of the game was discussed and created jointly by the team in the weekly team meetings. Puzzles, rooms, and general layout of the game was a collaborative effort.

PROJECT MANAGEMENT

TEAM ORGANIZATION

Roles	Design	Implementation	Maintenance
Lead	Scott	Jordon	Mackenzie
Librarian	Mackenzie	Cassandra	Scott
QA Leader	Cassandra	Mackenzie	Jordon
Floater	Jordon	Scott	Cassandra

Role Descriptions

Leads: Coordination, Deadlines, Job delegation

Librarian: Reports, notes in weekly meetings, research
QA Lead: handles bug reports/issues, editing files, Questions to Professor/Nicole Wilson

Floater: Helps where most necessary, assigned role by team lead.

RISK MANAGEMENT

i. Requirements/Design/Estimation

1. The team Planned a project that is too large:
   If the project planned within the design stage is too big a task to complete by the project's deadline, then the team will start by eliminating the least important rooms as necessary, consequently removing all puzzles and features within them. If a feature within the room is essential to the game, it will be moved to another room.
   Another area of the project that will be reduced if the project is deemed too large will be puzzles. The puzzles associated with the removed rooms will be removed, the more complicated and time-consuming puzzle will be removed first. Any non- essential puzzles that do not depend on the game being won will be removed.

2. The team underestimated how long parts of the project would take:
   Assign the floater to assist, if not already loaded with tasks. Secondly, a deadline will be set, if the section of code is not completed by the deadline and is a non-essential feature, it will be removed from the design.
   However, if the part of the project is an essential feature it will be set a later deadline and worked on by those with the most knowledge of that section of the project and experience. If again, the section cannot be completed, at the next team meeting the discussion and change of the design will be organized, and tasks redistributed by the team lead.

3. Major changes to design are needed during implementation:
   If any major changes are needed during implementation, they will be discussed and planned out at the nearest team meeting or sooner if necessary. All members will be notified and informed of the needed changes. A new design will be planned out in one of the weekly meetings. The work for the needed changes will then be distributed by the team lead with a set deadline for their implementation. No one member will make unilateral changes without first informing their fellow teammates.

ii. People

1. Addition or loss of team member:

   In the case of an additional team member joining Juggernaut, the newest member will be updated on the purpose and stage of development of the game, from weekly meeting notes, individual progress reports, and the team members. Once sufficiently up to speed, the team lead will assign the newest member a task/work.

   In the case of losing a team member, the team member that is designated as a floater for that stage of development will take on the role recently vacated. The team will then discuss if there is any need to decrease the size of the project. Once the necessary changes are made, if any, the team will then reassign the work and tasks of the previously vacated team member.

2. Unproductive team member(s)

   Set deadlines, and if the members continually fail to meet them to a point of jeopardizing the project's completion, due to a lack of effort, their work will be reassigned by the team lead and this will be reflected in the team report. This is only to be done if the team member is uncooperative and unwilling to work with their fellow team members to complete the work and communicate their progress.

3. Team member(s) lacking expected technical background:

   These issues will be brought to the nearest weekly meetings, if the issue cannot be addressed/solved there the QA will ask either Dr. Anvik or Nicole Wilson. The librarian may also do research into the subject.

   If an individual is having issues with a specific implementation or software for longer than necessary, they will post the question/problem on the group's channel or ask Nicole Wilson if she is available. If someone is having an issue, after putting a concerted effort into trying to solve the problem, get help from fellow teammates or Nicole to avoid the headache and wasted time.

   In the case of persistent and overall lack of understanding and ability the team member will shadow the more experienced programmers in the group. In addition to this, their work load in regards to where they are lacking expected technical background will be reduced and they will be reassigned tasks that better suit the individuals capabilities, whether that be writing reports, testing, fixing style errors etc..

4. Major life events:

   The same process used for the addition or loss of a member will be implemented in this case. If these major life events affect the team lead, the roles will be adjusted accordingly.

iii. Learning and Tools

1. Inexperienced with new tools:

   The team will be primarily sticking with solely using Atom and GitLab in order to minimize time spent learning to use new tools. If necessary, however, the most experienced in the group will explain the tool to the more inexperienced user. In the case that no member in the team are familiar with the tools, the QA lead will go to Dr. Anvik or Nicole Wilson to inform themselves and report to the rest of the team.
   For using unknown libraries and data types, cplusplus.com will be used.

2. Learning curve for tools steeper than expected:

   Roughly the same process as in dealing with new tools. Dedicate more time to learning. Have one individual focus, most likely the QA lead or the floater to study and learn these tools and bring back gathered intel to the rest of the group during one of the weekly meetings, or by updating their personal progress report.

3. Tools do not work together in an integrated way:
   Get help from IT. Librarian can investigate the issue as well and report back findings.

DEVEVLOPMENT PROCESS

CODE REVIEW PROCESS

i. Pull Requests:
   a. To minimize the number of pull requests, the team lead will clearly divide the workload and methods to be programmed so that no two individuals are working on the same code.
   b. If a pull request is necessary, attempt to handle them in the weekly meeting where everyone is present, so that all will be aware of the situation and updates.
   c. Otherwise, so that all team members can continue working on their section of the project, take both versions of the code(our then theirs), then comment out one version of the code noting the time and differences between the two sections of code for the team lead to review later, or to be brought up in the weekly meetings.
   d. This same process will be used to deal with merge requests

ii. S.O.L.I.D
   a. QA lead and floater will handle implementation of S.O.L.I.D principles when needed, though all members will attempt to follow such principles as they write code, in order to avoid creating more work later.

iii. Steps to Code review process:

1. Team lead assigns work/tasks to complete
    a. Small tasks and early deadlines, frequent updates
2. Set deadlines for those tasks/work
3. Write appropriate tests for methods to be programmed before working on the code
4. Complete work: no leaving it in a non-compilable condition. If leaving something in a non-compilable condition, comment it out and note that is was commented it out in the push message, before pushing non-operable work into the groups repository. Also, update this information in the individual's personal progress report.
5. Review of the code:
    a. Review your own first (style errors, etc.…)
    b. Go over progress and problems in the weekly meetings
    c. Keep personal progress reports updated in MS Teams
    d. To the best of our abilities, get previous code working before moving onto the next sections of the project.

COMMUNICATION TOOLS

The team will be using MS Teams as their primary method of communication. The team will meet three times a week, Monday and Friday at 15h00 and on Wednesday's at 13h00, setting aside an hour each time, whether that whole hour is necessary will be determined on a case-by-case basis.

In the testing stage, all issues will be handled and reported using GitLab's issue tracker.

Every developer on the project will keep a file in the team's MS Team channel and keep an updated progress report that the other members of the team may view to inform themselves of the progress done and of what needs to be done, in addition to any issues one developer may be faced with.

CHANGE MANAGEMENT

Issue and bug reporting will be done on GitLab, the QA Lead is responsible for managing the issues and notifying their fellow teammates. Only the QA Lead and the Team Lead will be allowed to close issues.

Once a bug is reported, the QA lead will investigate the issue and inform the team lead, additionally the QA lead will inform the developer whose code contains the issue which needs to be corrected. If the issue is between two developers code, the issue will be documented and reported at the next team meeting where the issue may be discussed and resolved. In all bug reports, there is to be a specification of what the problem is, what was expected, what happened, and how to recreate the issue. Add any additional notes necessary, i.e what sections of the program this bug will affect and possible solutions.

The Team reports will be discussed in the meetings, the librarian will take notes during these meetings that will be used to complete the reports. All team members will look over the report before final submission.

SOFTWARE DESIGN

DESIGN

The design changed a lot during the Implementation phase, as we realized we had too much dependency between the classes and circular dependency. We decided to separate the classes more, add a few classes (Inventory, PlayerStat, Coordinate, Item) and remove a couple classes (Door and Player). The Area, Room and Item started out as pure virtual classes, but we realized later on that they needed to be abstract classes, that the other classes could inherit from, and use some of the same methods, but also allowed for variation within the derived classes. This allowed us to follow DRY principles and allowed us to write the basic algorithm (below) and just reuse the process for all classes used during the user's turn. This allows us to use the same execution for the Puzzles, Object or NPC interaction (eg. Positive outcome = pick up the object, negative = leave the object on the ground).

Basic algorithm that will be run by Room class:

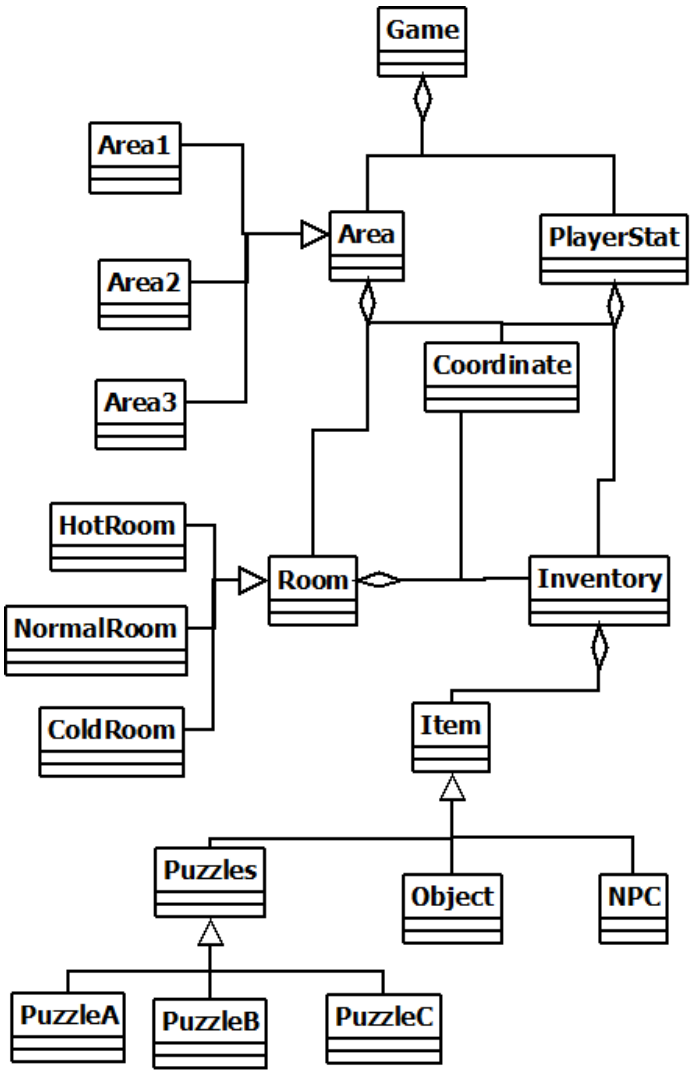
1. Print necessary description
2. Asks the user for input
3. Checks the user's input.
4. Modify any affected variables.
5. Print the outcome
6. Exit the loop Design Rational:

The rational behind the design is the D.R.Y principle as well as the use of abstraction and polymorphism to lower the amount of programming necessary, as well as following the design principles presented in this course.

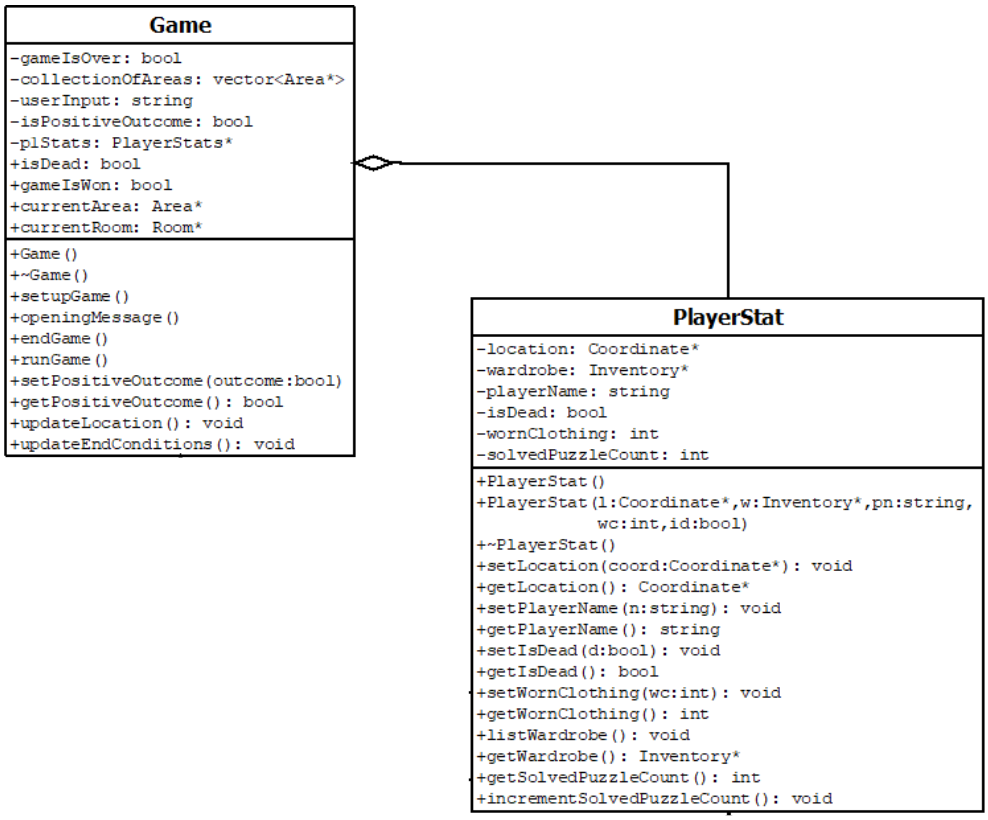
APPENDICES

APPENDIX A: CLASS DIAGRAMS

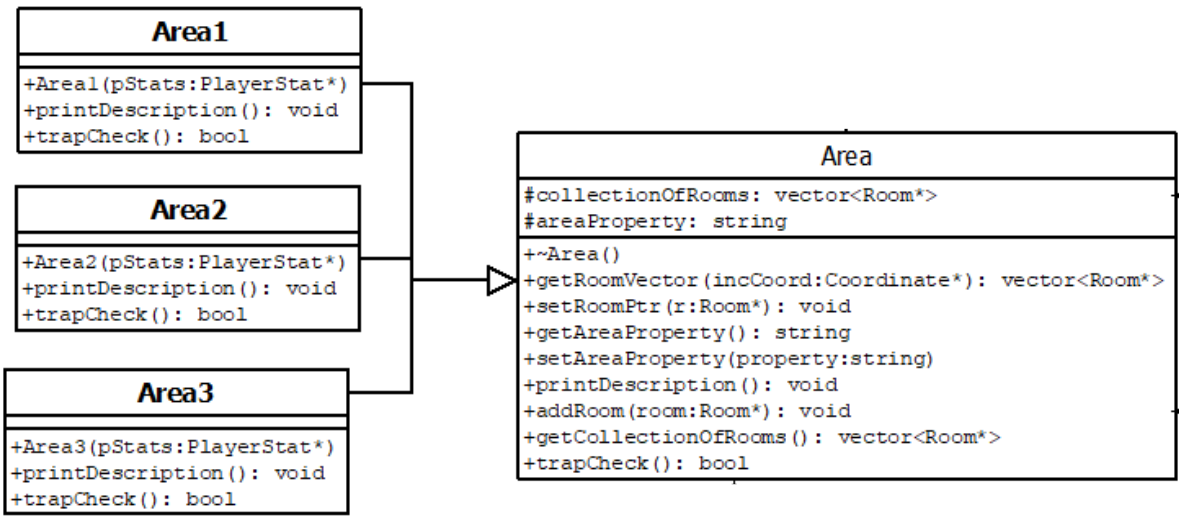
1. Overview (includes all the classes other than the Exceptions classes which are housed together in Exceptions.h and used by most classes).



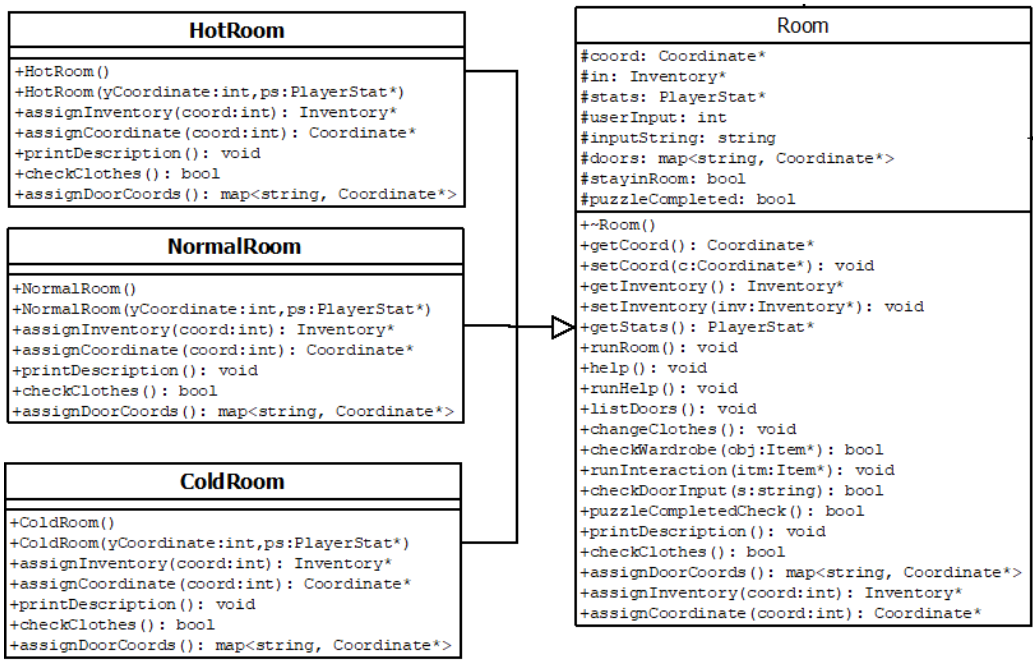
2. Game and PlayerStat Classes



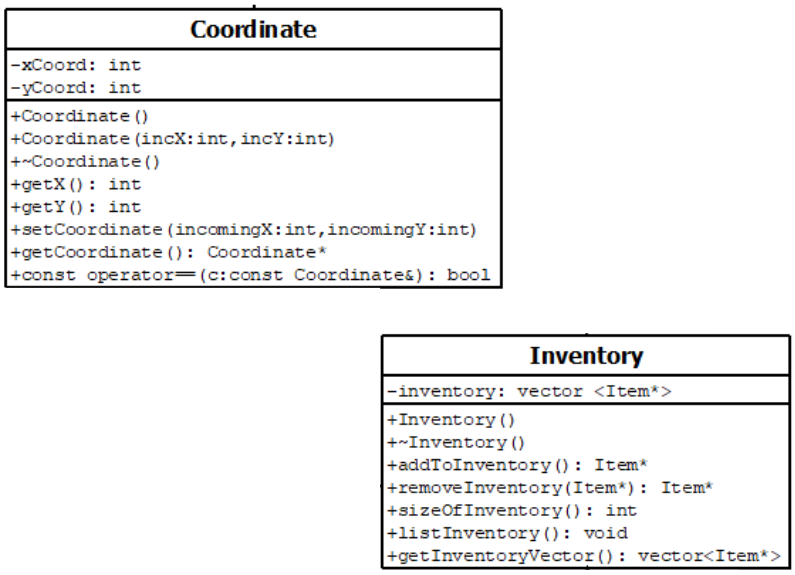
3. Area Classes



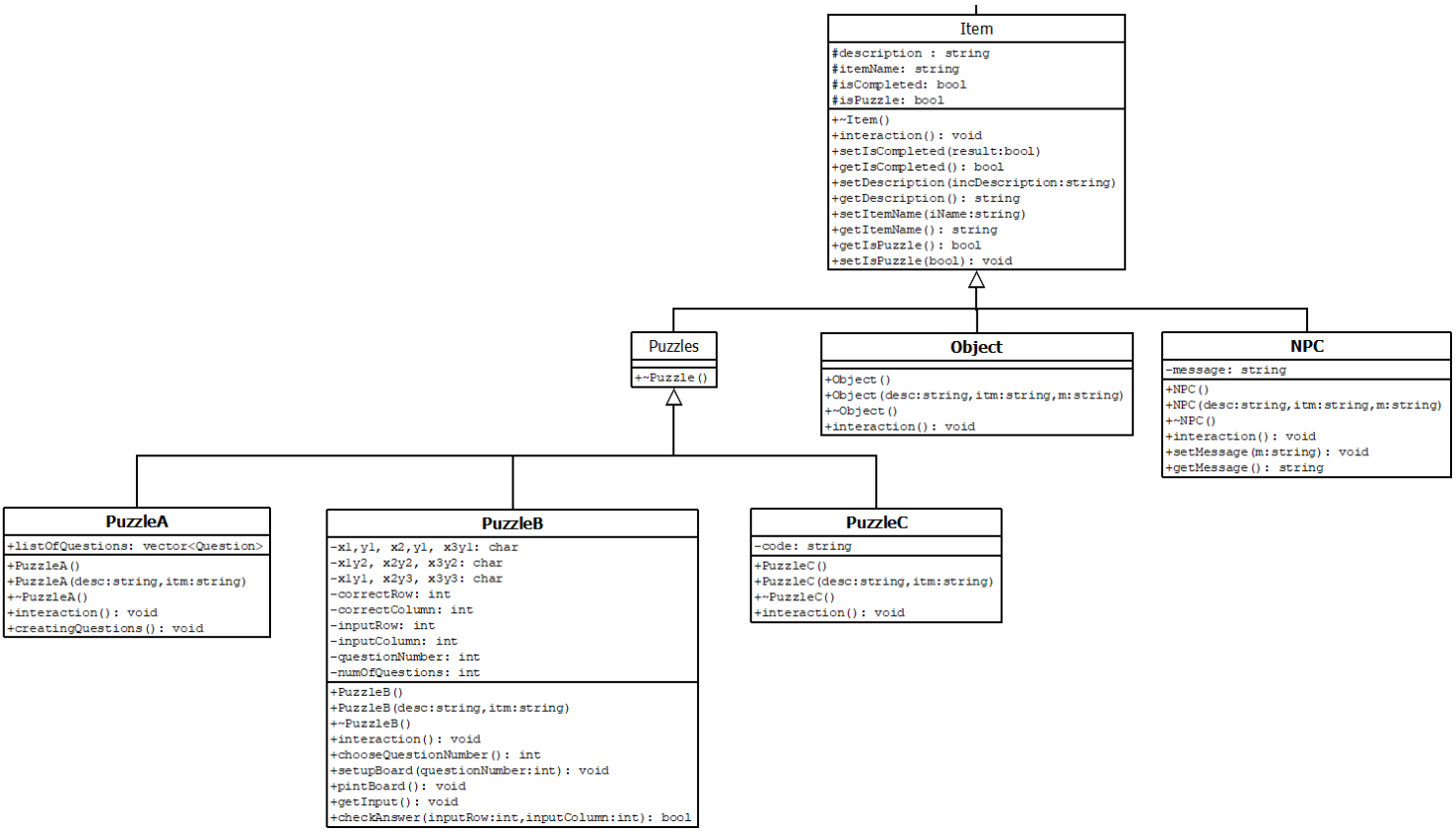
4. Room Classes



5. Coordinate, Inventory Classes

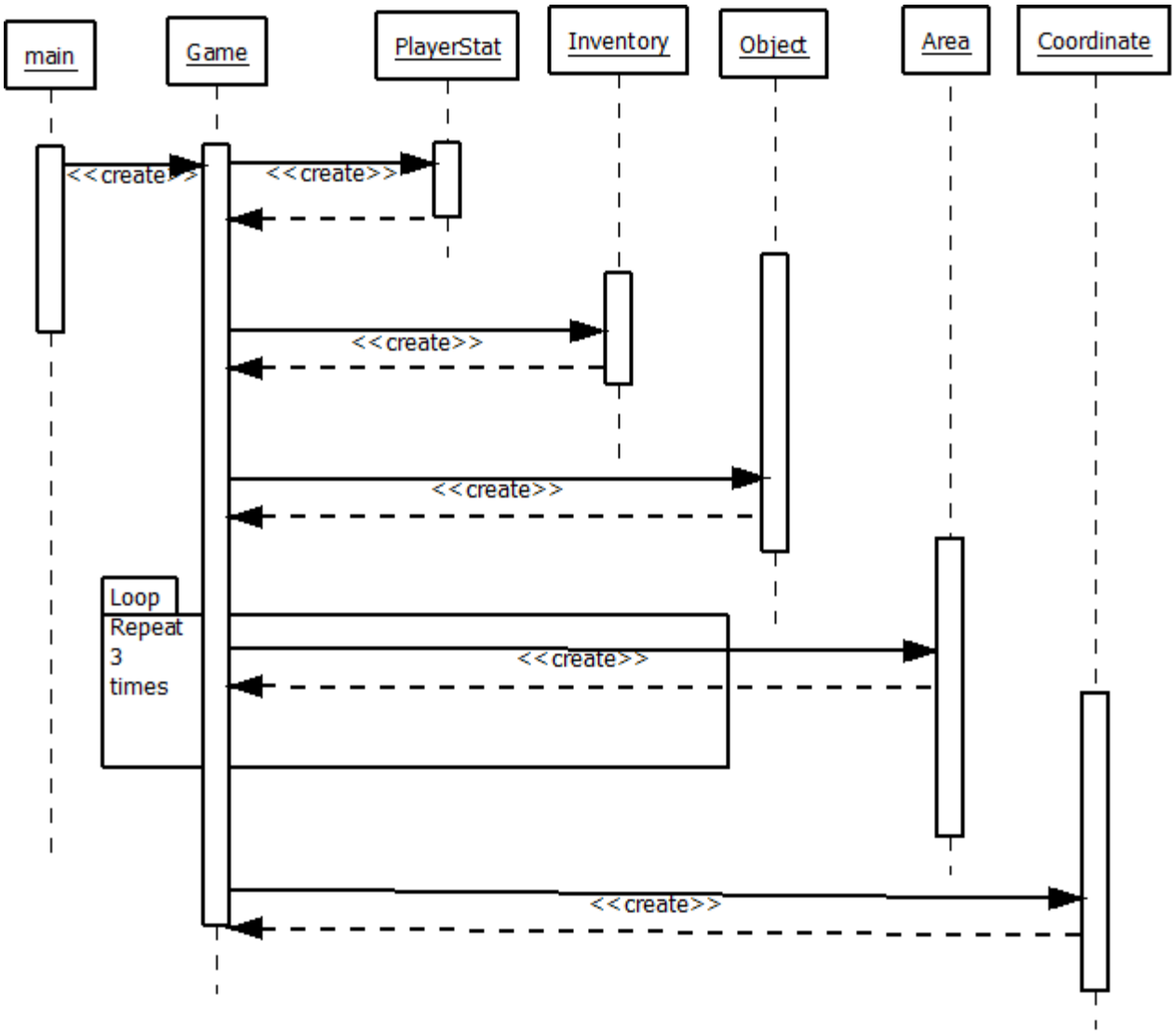


6. Item, Puzzle, Object and NPC classes

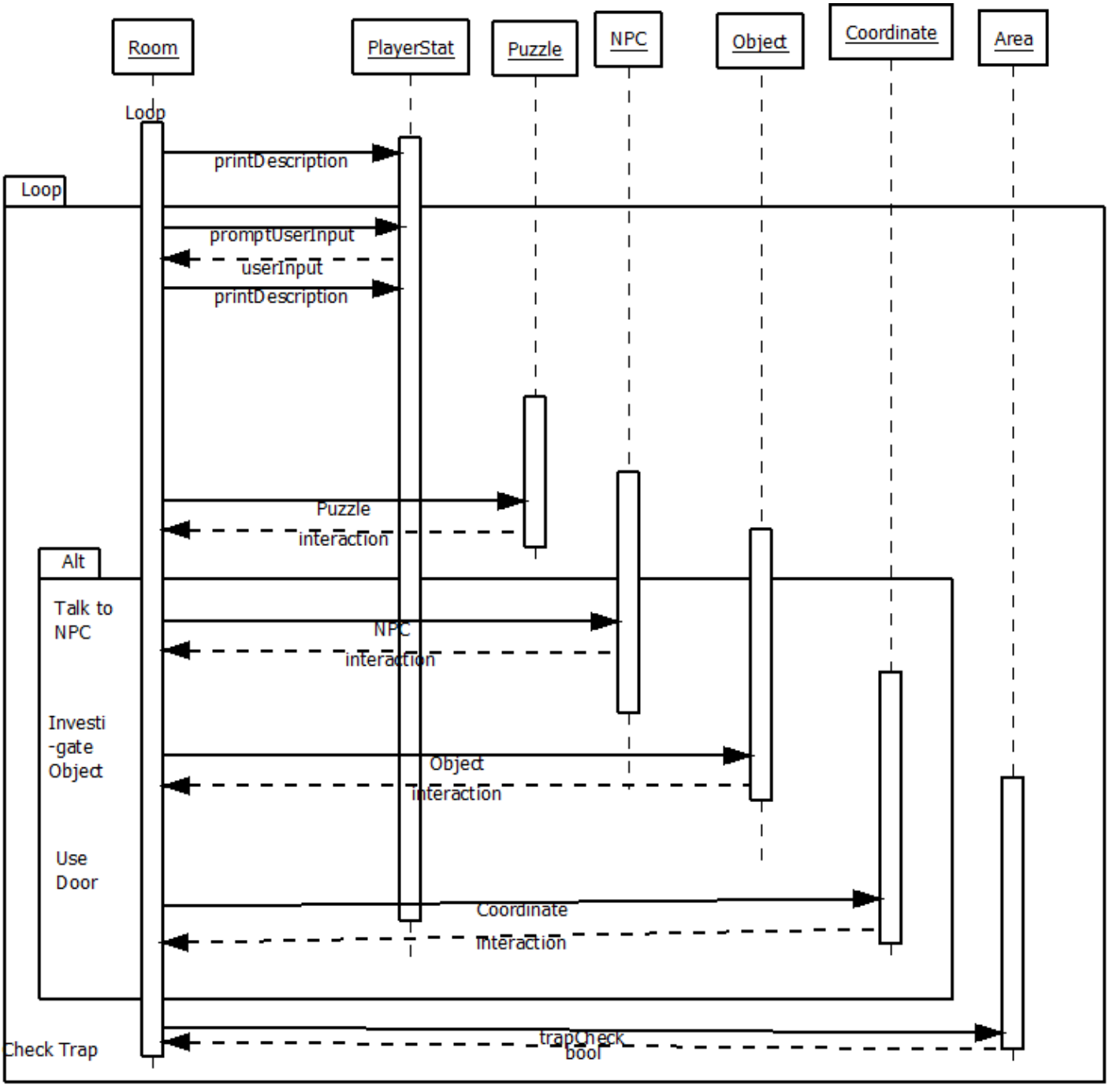


APPENDIX B: SEQUENCE DIAGRAMS

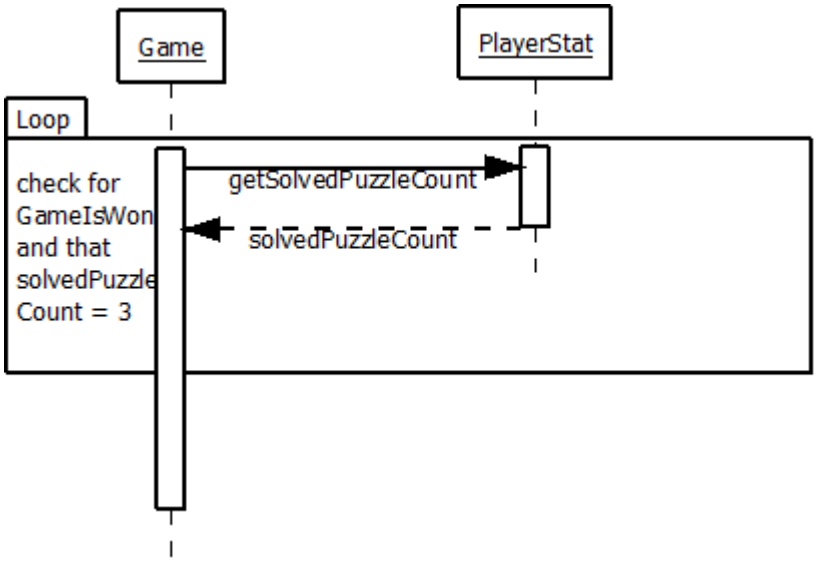
1. Game Setup



2. One Turn


Room
PlayerStat
Puzzle
NPC
Object
Coordinate
Area
Loop
printDescription
Loop
promptUserInput
userInput
printDescription
Puzzle
interaction
Alt
Talk to
NPC
NPC
interaction
Investi
-gate
Object
Object
interaction
Use
Door
Coordinate
interaction
Check Trap
trapCheck
bool

3. End Game


Game
PlayerStat
Loop
check for GameIsWon and that solvedPuzzle Count = 3
getSolvedPuzzleCount
solvedPuzzleCount