Disaster on the Good Ship Lethbridge

Team Calrissien

Team C

Max Niu and Steven Deutekom

due February 20, 2019

TABLE OF CONTENTS

Introduction----------3
Project Management----------4
    Team Organization----------4
    Risk Management----------4
Development Process----------5
Software Design----------5
    Design----------5
    Design Rationa1----------5
Appendices----------6
    Appendix A----------6

Introduction

This is a student game project developed by Max Niu & Steven Deutekom, The following are the purposes of this project:

I. Practice advance object-oriented design and techniques

II. Implement test driven development, error detection and handling

III. Develop team management skill on collaborative software project

IV. Practice UNIX programming tools and scripting language

Disaster on the spaceship Lethbridge is a text based adventure game developed using c++.

The player will start his/her's journey as the captain of the space ship named "Lethbridge" and work their way to solve the puzzles and avoid all the death traps and eventually save the crew and the ship.

There is no scoring. The goal is to reach the end of the game where you save your ship from disaster. Each turn will consist of entering an action and, if necessary, an item to do the action to. The game will process this command and inform the player of the result. This process is repeated until the end is reached or the player is killed.

Player actions

Action 1: Drop — remove an item from players inventory
Action 2: Get — get an item from the environment
Action 3: Use — use an item in the environment or players inventory
Action 4: Talk — interact with an NPC
Action 5: Look — get details about a location or item
Action 6: Exit — move to another location
Action 7: Inventory — display the players inventory

In game commands

Command 1: Save — save the game
Command 2: Help — display some help documentation
Command 3: Load — load a previously saved game
Command 4: Quit — Leave the game

Project Management

Team Organization

Max Niu - Design Lead, Story writer, Documentation assistant , Software Developer, Software Tester

Steven Deutekom — Documentation, Quality assurance Lead, Software Developer, Software Tester

Shared Responsibilities - Merge

Risk Management

1. Requirements/Deign/Estimation

Over Estimation: eliminate some functionality, or get basic game working and add what we can in the time we have.

Changes: we do the best we can and save as much as possible but simplify.

Addition: do our best to integrate them and give them tasks that will help finish the game. Could even add some function if having an extra member means we are done quickly.

Technical issues: work together to help understand necessary technical stuff.

Tools all issues: stick to what we know, if things are tough seek help from experienced individuals (teachers/team members).

2.People

Human Casualties: if its close to deadline just finish up what you can, earlier join another group since we are team of two. Press F to pay respect and the name will appear in game credits.

Unproductive team members: team leader will be speak to this individual about the current situation and figure out what caused the loss of productiveness, and put him back on track. note this member must bring snacks to the next team meeting.

3.Tools

Unfamiliar Tools: will not use tool that is not taught in class, ask for help from other member and seek help from instructors.

Development Process

Code Review Process:

this happens every Monday,Wednesday and Friday after cpsc2720 class the two member will meet and discuses the progress they made, if there is any uncertainty on the quality of the code the team lead will decide if we add or reject the code.

Communication Tools:

verbal & telepathic communication as well as text though cell phone.

Change Management:

1.if bug report is filed, the QA lead will find the team member who wrote the bug code and assist them fix the code.

2. if bugs is from two member QA lead will identify the problem and reassign member for bug fix.

3. QA lead will replay to the bug report appropriately.

Software Design

Design:

We tried to use inheritance where possible to encapsulate common behaviors. We also tried to work as many pieces of the game into their own classes with well defined behavior. This also allowed us to separate some components out to their own classes and then build other classes from them. Hopefully, this strategy of inheritance and composition will allow us to write less code and to be able to add functionality where it is needed with ease.

Design Rationale:

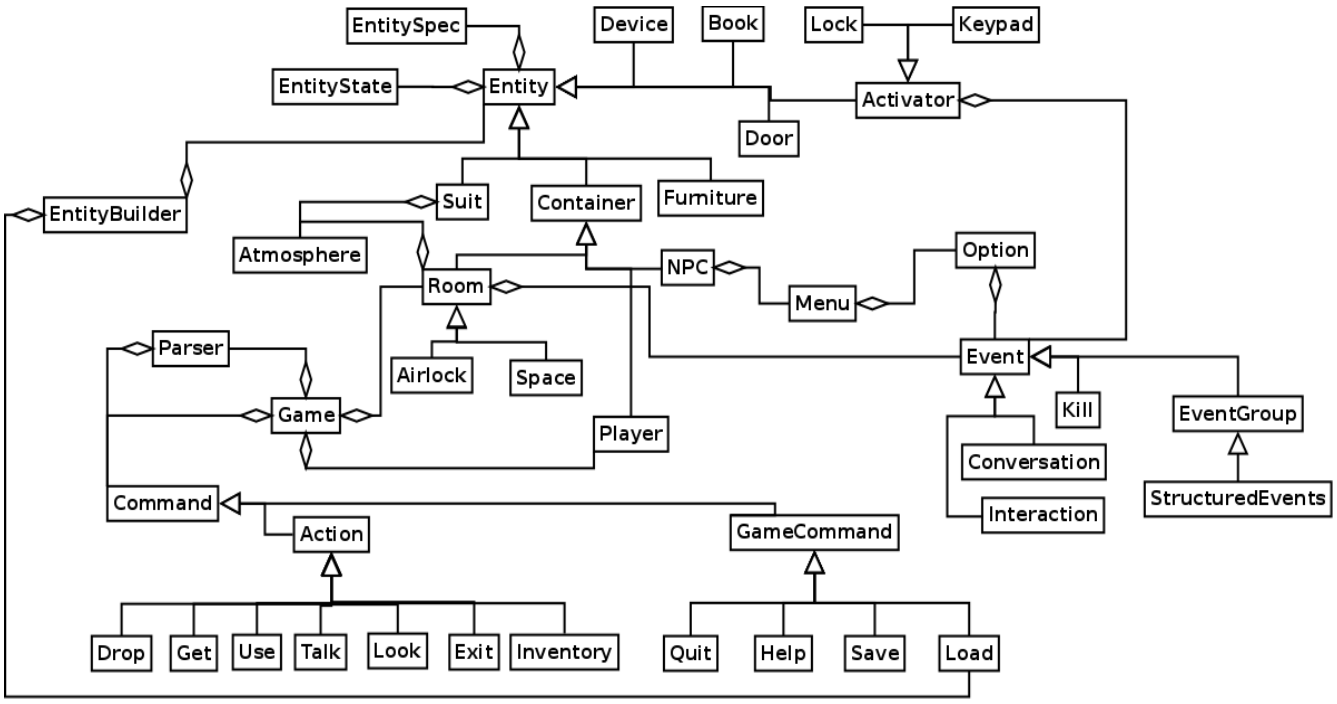
The goal of this project is to learn advanced design principals, and to prove we understand these design principals, as well as practice test driven development. We have attempted as stated above to use good judgment in applying inheritance and composition. We are also trying to use the Single responsibility principle where possible. But we recognize that in the projects early stages some of these principals will be violated in order to deliver a product on time. Hopefully, the things we learn in class will help us to improve the project as we go

and gain experience from the mistakes we will inevitably make in the early stages.

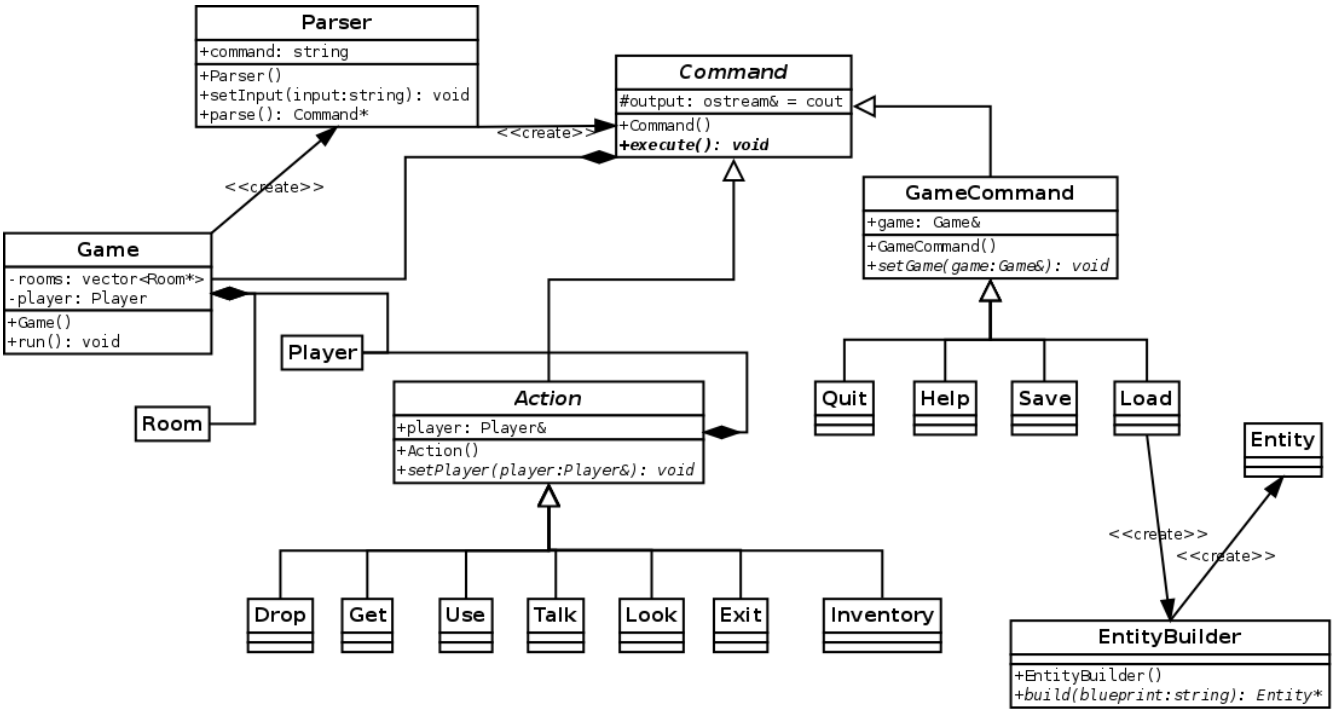
Appendices

Appendix A: Figures and Diagrams

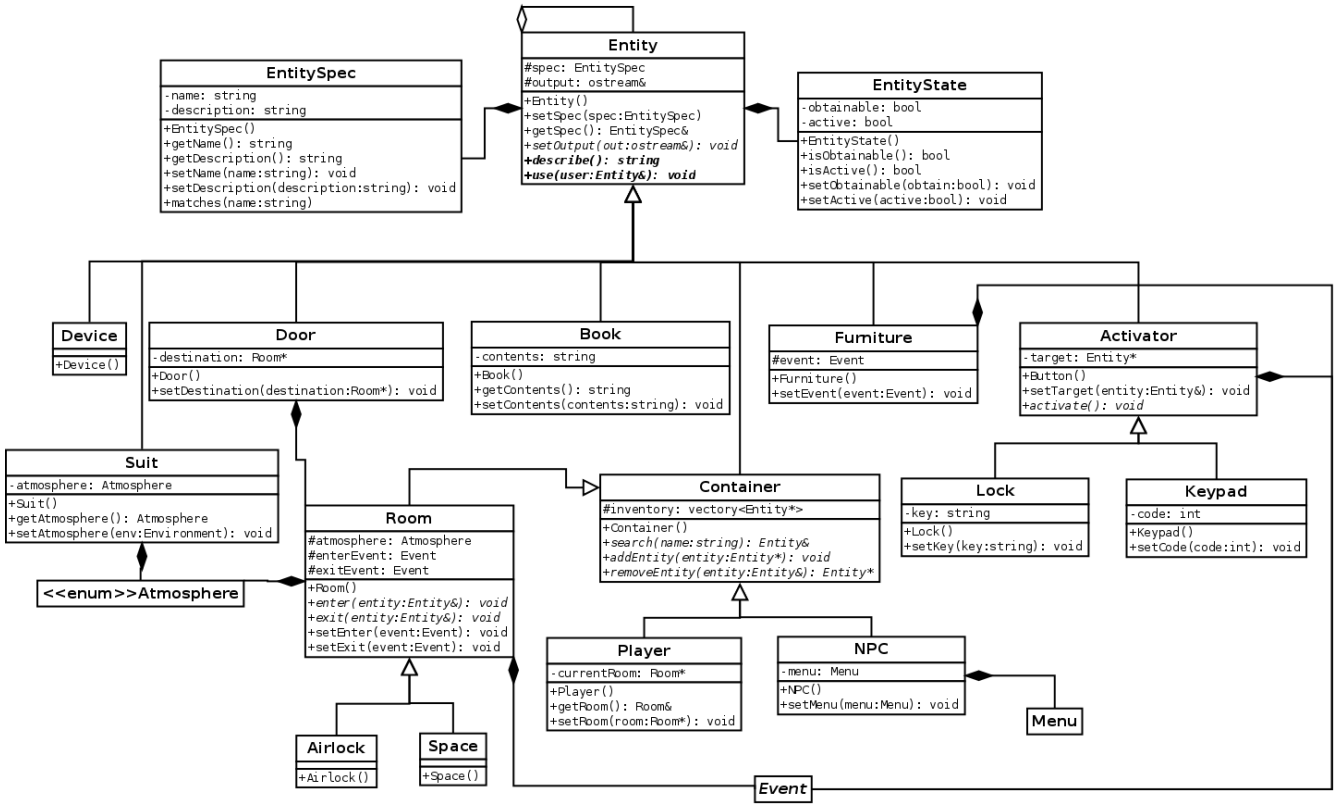
1. Overview (includes all classes currently in the project)



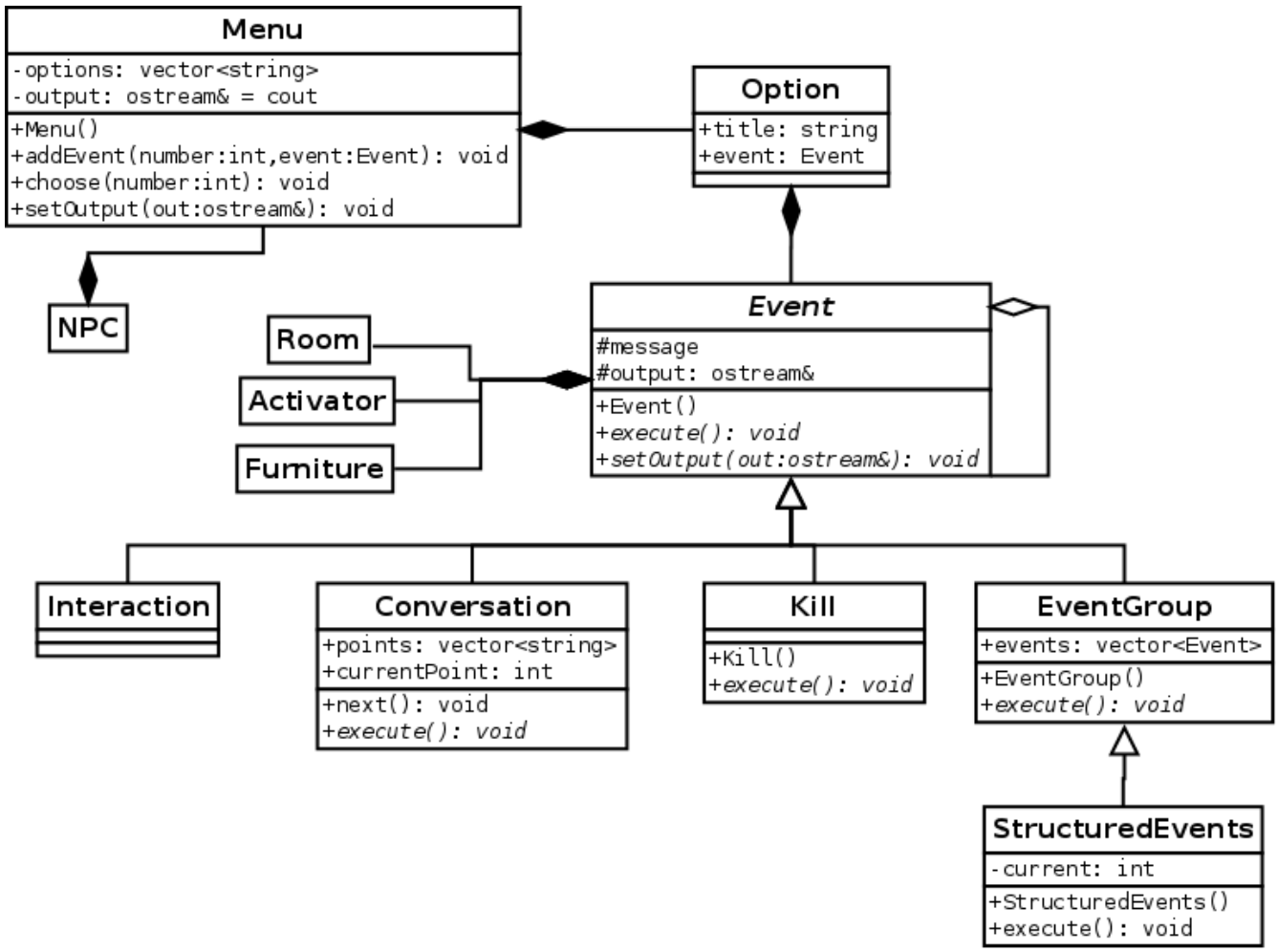
2. Game system components (Room, Player, and Entity on next page)



3. Entity components (Menu and Event on the next page)



4. Menu and Event components



5. Sequence diagram