Something about a Diary...

Nate Jones, Kelley Kelley, Anton Machkasov
Ryan McGarvey, Aryan Regmi, Alex Southard

Project Management Method

Overall development followed the Waterfall process

- Defined all desired functionality in the first several weeks with a basic skeleton of what everything would do
- Created a logical order for development of all features
- Each individual component of the project would then implement one feature of that component at a time, be tested, then continue following an agile methodology

Tools

Project Tracker: Asana with Instagantt
Repository: GitHub
Database: PostgreSQL through a QT API (to work with c++)
Testing Methods and Tools:

- QT Debugger (5 out of 5)--to ensure all values set correctly in the software and to go line by line when something went wrong
- Testing:
    - Regression and Integration testing
        - Program comprised of many different classes and interactions
    - System Testing
        - Ensuring that the QT, c++, and PostgreSQL components work well together
    - Since the program doesn't work with significant amounts of data, most testing could be performed by hand.
- Deployment: LocalHost

IDE: QT Creator

Asana with Instagantt

Rating: 2

Wasn’t very useful

Unintuitive

Github

Rating: 5

Made sharing work extremely easy

PostgreSQL

Rating: 2

Hard to use and not particularly useful

QT Creator

Rating: 5

Had really easy to use implementation of useful tools

How did we test?

Rating: 5/5 (for entertainment)

- Play-testing, trying to break the program
- QT Debugger
  - Traces functions to see where errors happen
  - Make sure values are appropriately set

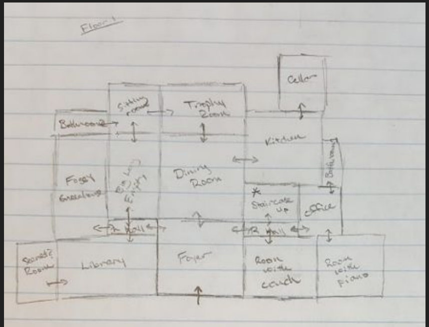
Deployment Environment

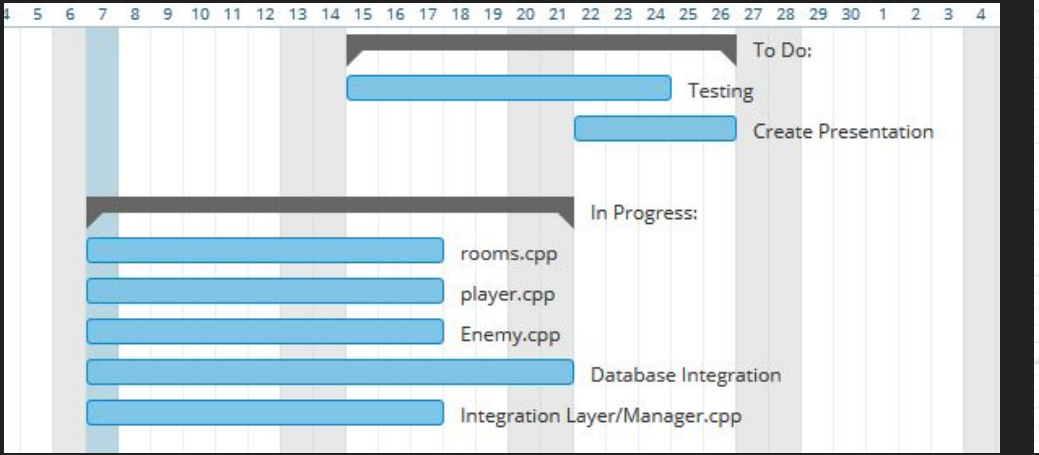
Rating: 4/5
--Convenient, but hard to work with

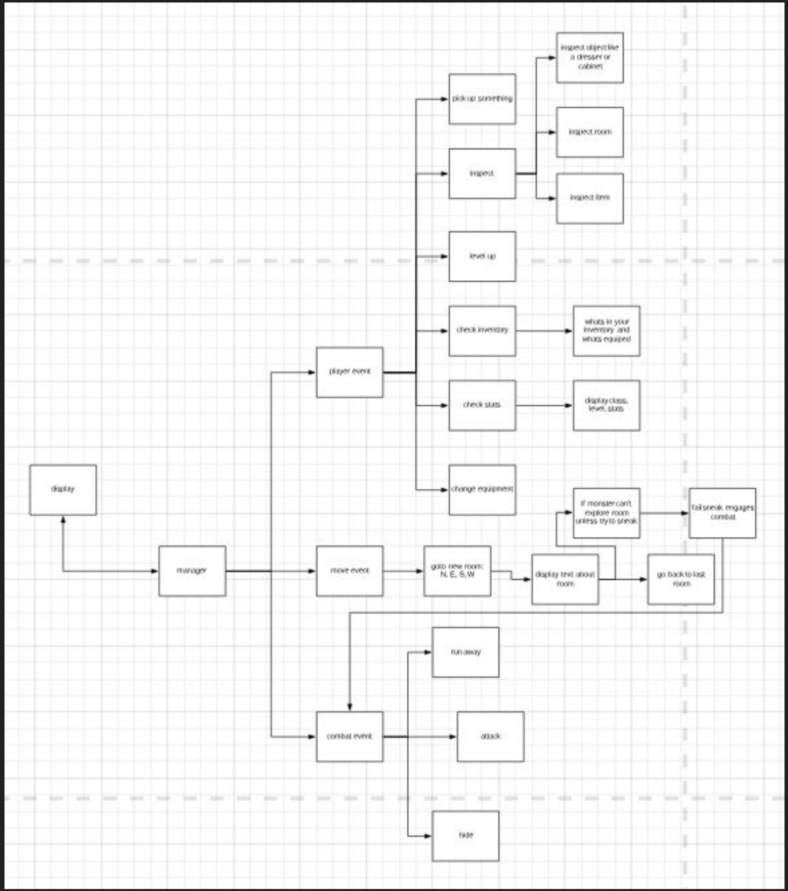
- Using Localhost to deploy
- Postgres database on Heroku

127.0.0.1

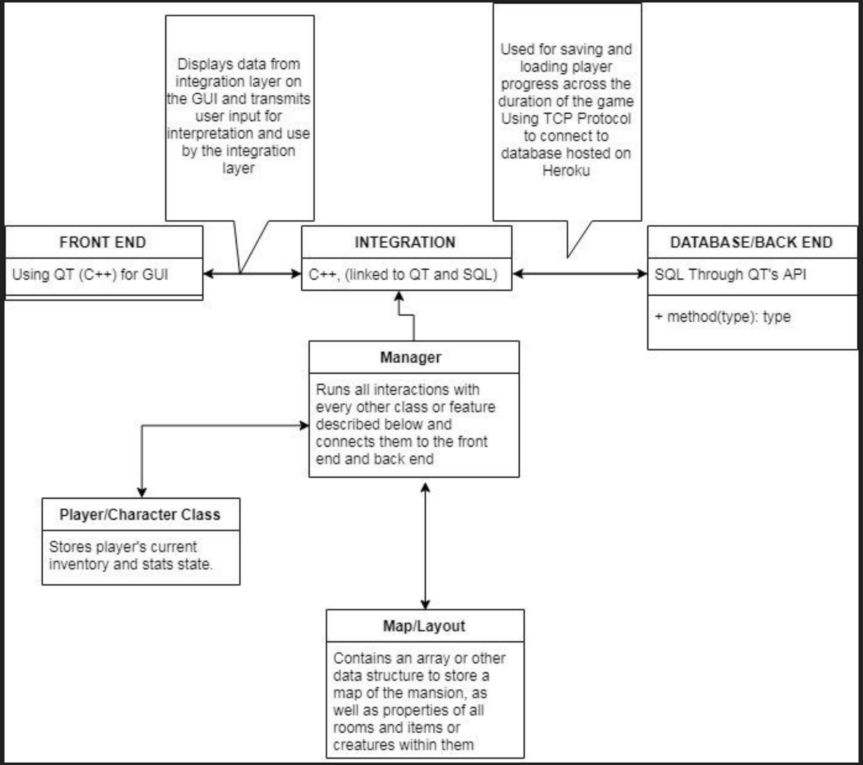
Some Planning Images






display
manager
player event
pick up something
inspect
inspect object like a dresser or cabinet
inspect room
inspect item
level up
check inventory
whats in your inventory and whats equiped
check stats
display class, level, stats
change equipment
move event
goto new room: N, E, S, W
display text about room
if monster can't explore room unless try to sneak
fail sneak engages combat
go back to last room
combat event
run away
attack
hide

The Stack Overview



Displays data from integration layer on the GUI and transmits user input for interpretation and use by the integration layer

Used for saving and loading player progress across the duration of the game Using TCP Protocol to connect to database hosted on Heroku

FRONT END

Using QT (C++) for GUI

INTEGRATION

C++, (linked to QT and SQL)

DATABASE/BACK END

SQL Through QT's API

+ method(type): type

Manager

Runs all interactions with every other class or feature described below and connects them to the front end and back end

Player/Character Class

Stores player's current inventory and stats state.

Map/Layout

Contains an array or other data structure to store a map of the mansion, as well as properties of all rooms and items or creatures within them

What Went Wrong?

- Trying to merge branches in GitHub to add SQL functionality, causing a “catastrophic failure”
- Conflicting coding styles and conventions
  - Expecting methods to be written differently or have a certain return type
- Complex dependencies across many c++ classes and files
  - Communicating and organizing work across all of these
- Very large code base
  - (3200+ lines of code)

How We Fixed Wrongs

- Someone who has an older version uploaded their version
- Most work was completed at our weekly meetings
    - Allowed Collaboration and more efficient project design
    - Eased difficulties caused by complex class hierarchy
    - Whenever conflicting functionality or someone needed a function could just ask for it from the other team member present

Overall end result wasn’t affected by challenges