Camelopardalis: Project Plan

Team Members:

Aaron Boutin
Babatunde Ogunsaju
Jessica Spokoyny

Introduction:

Command Line Interface Adventure Games, a form of Interactive Fiction, have been around since personal computers became popular. These games provide a text-based interaction, where the player goes on an adventure by using text as input and the game responds in kind with text indicating the result of their action. Over the course of an eight week period, we plan to design, implement and test a command line adventure game. Our game theme is a murder mystery that will allow the player to navigate and investigate a small town in search of a killer, at large, using natural language. This project will allow us to refresh our knowledge of C++ and learn to implement new functionality, such as language parsing and implementing a game engine.

User Perspective:

The user is presented with a Command Line based Murder/Mystery style adventure game. The user will use a series of approved verbs, in naturally structured sentences, to navigate the game. He/she will be able to move between locations and around rooms, make observations about the surroundings, speak with characters, and gather evidence. When the user enters text the game responds with text describing the results of their actions. The game starts out in Small Town USA, where the user is a private investigator from the city who has been called in to investigate a murder at the local bar. To win the game, the user will need to follow the evidence and arrest the killer.

Structure:

Our perceived projection of the structure of our software will be such that it allows for easy maintenance and modification. Every Room, Object, and Character is instantiated from their respective classes. Each will contain verb objects that can be set to perform the appropriate action requested by the player specific to that room, object, or character.

The game engine will contain a game class that deals with calling upon load and save functionality, as well as initializing and running a game. Running the game involves prompting the player for a sentence. The engine calls the parser and then sends that input to the appropriate object. At a minimum this will repeat.

The Command Parser will take a sentence from the command line as input, extract the intent from verbs and nouns, and output them as arguments to be used on the intended room, object, or character. The parser will also keep an miniature thesaurus for the approved verbs to allow variation in used sentences and allow the player to feel more comfortable in writing naturally.

The game data will be saved in such a way that each room will be written to a file of its own, containing the state it was in when saved, including the objects and characters that were in it. The Player state will be saved in its own file. The data will be written using readable text.

Characters

Beth: Love triangle's center, used to date Sam, currently dates Bobby, best friends with Claire

- Originally located at Beth's house

Bobby: Beth's new boyfriend/Vic

- Originally located at Bar (dead)

Claire: Beth's psycho best friend, used to date Trent

- Originally located at Beth's house

Sam: Beth's old boyfriend, might still be in love with Beth

- Originally located in Moose's house

Frank: Medical Examiner

- Originally located in Morgue

Detective: the user

- Originally located in Bar

Trent: Bartender, used to date Claire

- Originally located in Bar

Trevor: Underage Drinker who saw the fight!, Trent's younger brother

- Originally located in Trent's house

Moose: Loan shark, owed money by Trent

- Originally located in Moose's house

Abe: Police Captain, Knows about bunker in Claire's house her father built during cold war

- Originally located in Police Station

Rory: Pharmacist

- Originally located in Pharmacy

**Other Bar patrons to be added as decoys for misdirection (time permitting)

Locations

Small town USA, everything is on Main St.

The Bar (2)
- Hidden Room

Police Station (1)

Morgue (1)

Pharmacy (1)

Bobby's House (4)
- Living Room
- Office
- Bedroom
- Front Porch

Claire's House (4)
- Bedroom
- Living Room
- Hidden Bunker
- Front Porch

Beth's House (3)
- Bedroom
- Living Room
- Front Porch

Sam's House (3)
- Bedroom
- Living Room
- Front Porch

Trent's house (3)
- Bedroom
- Living Room
- Front Porch

Moose's house (3)
- Bedroom
- Living Room
- Front Porch

Total Rooms: 19

Verbs

Go	(lets the player navigate spaces/rooms)
Look	(informs the player about what they see in the current room)
Look at	(lets the player examine a specific object/character)
Take	(allows the player to add object to inventory)
Help	(displays a list of approved verbs)
Hint	(displays relevant hint to player on what they should do next)
Inventory	(displays a list of the player's inventory)
Talk/ask	(allows the player to engage in dialogue with characters)
Arrest	(accuse character to end game)
Use	(interact with an object)
Enter	(go into house/building)
Knock	(request entry into building)
Break In	(force entry into building)
Drop	(put item down at current room/location)
Save game	(saves the state of the game to a file)
Load game	(loads the game state from the file)

Total Verbs: 14

Carriable Items

Trevor's Wallet
- Originally located in hidden room of bar

Vial Evidence
- Originally located in bar (trash)

Sam's Love Note Evidence
- Originally located in Sam's house

Autopsy Report Evidence
- Originally located in Morgue

Pharmacy Pickup Receipt Evidence
- Originally located in Pharmacy

Pill Bottle
- Originally located in Claire's bunker

Engagement Ring
- Originally located in Bobby's house

Bobby Phone's
- Originally located in bar (on Bobby's body)

Lockpick

- Originally located in detective's bag

Flashlight

- Originally located in detective's bag

Suspect List

- Originally located in detective's bag

Software/Systems Required:

We will build our software using the C++ programming language. We will use standard libraries(tentative) and the software will compile and run in a Linux environment. For development tools we will use Cloud9 (https://c9.io) as a collaboration tool, and the OSU Engineering Server to run our game.

Team Member Tasks:

Aaron: Command Parsing Developer

Week & Associated Tasks	**Estimated Time in Hours**
Week 3: Develop Text Parser Module Submit video update	15
Week 4: Continue Text Parser Module Work Unit Test Text Parser Create Verb Class	15
Week 5: Unit Test Verb Class Integration Test Parser w/ Verb Objects	10
Week 6: Integration Test Modules Create functional demo	13
Week 7: Develop Character Dialogue Submit video update	13
Week 8: Develop Character Dialogue Develop Object/Room Attributes	13
Week 9: Improve User Interaction	13
Week 10: Final System Tests Final Code review/ Refactor Submit Project	13
Total Hours	105

Jessica: Engine Developer

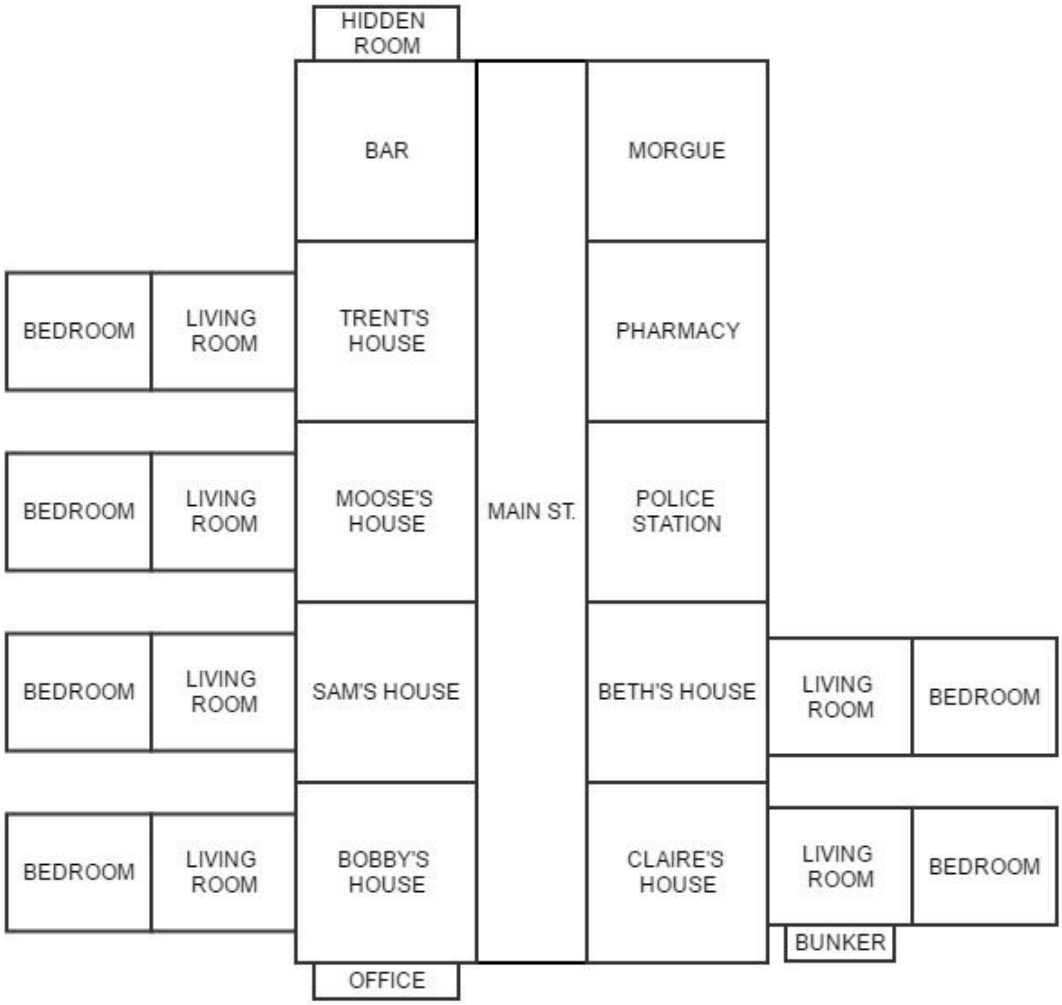
Week & Associated Tasks	**Estimated Time in Hours**
Week 3: Create player class Create room class Create object/item class Create game class	15
Week 4: Create player subclasses Create object/item subclasses Create room subclasses	15
Week 5: Create code to initialize new game Create main function Unit test classes Submit video update	13
Week 6: Integration Test Modules Create functional demo Submit Mid-Point Documentation	13
Week 7: Develop Character Dialogue	11
Week 8: Develop Object/Room Attributes Perform tests and fix bugs	13
Week 9: Improve User Interaction Submit Final Report	13
Week 10:Final System Tests Final Code review/ Refactor Submit Project	13
Total Hours	106

Tunde: Data Format Developer

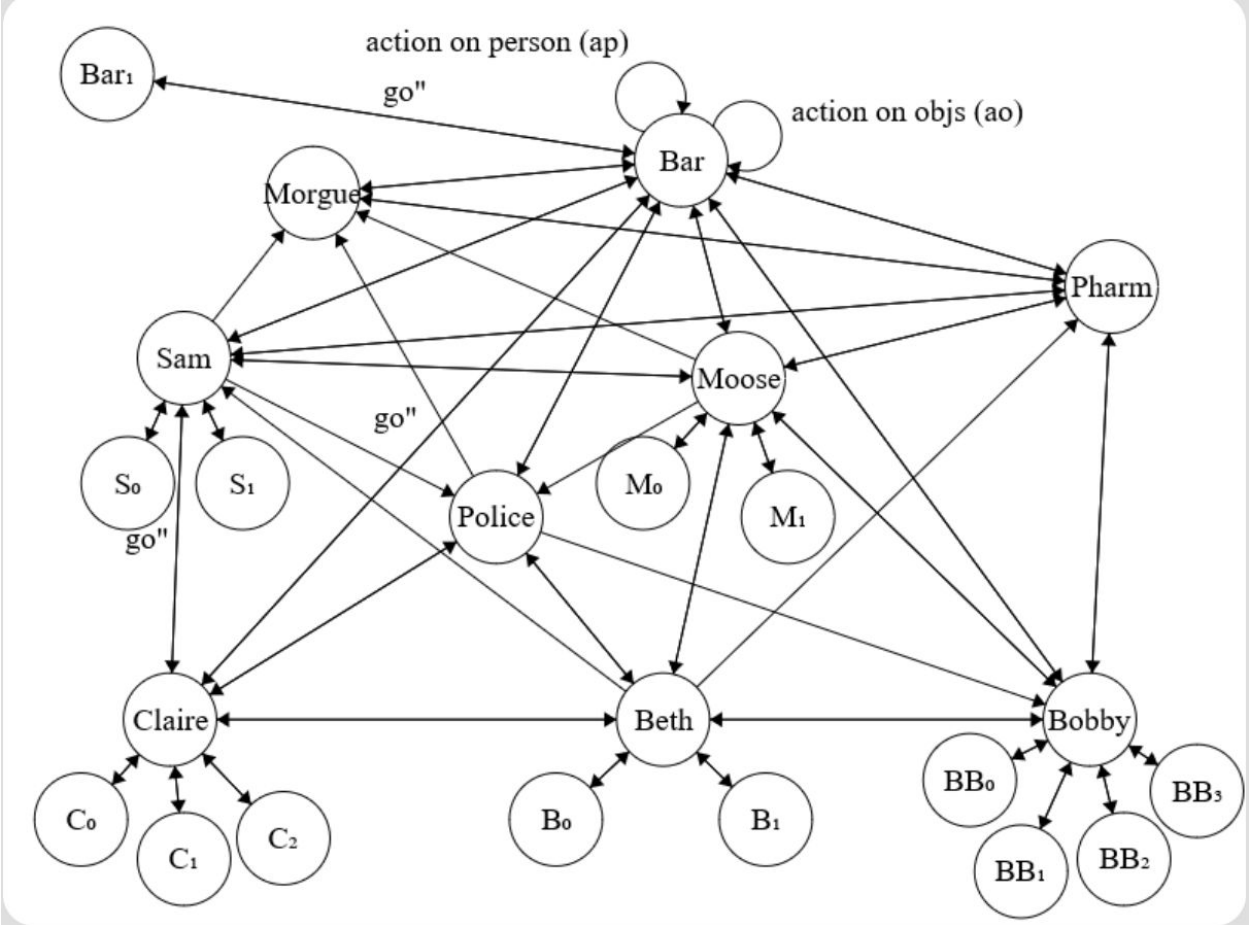
Week & Associated Tasks	**Estimated Time in Hours**
Week 3: Design data format for Room objects Identify activities that require APIs	13
Week 4: Create data format APIs Submit video update	13
Week 5: Unit Test APIs	13
Week 6: Integration Test Modules Create functional demo	13
Week 7: Develop Character Dialogue	13
Week 8: Character development Submit video update	13
Week 9: Improve User Interaction Submit demo	13
Week 10:Final System Tests Submit Project	13
Total Hours	104

Design Diagrams:

Town Layout Representation:

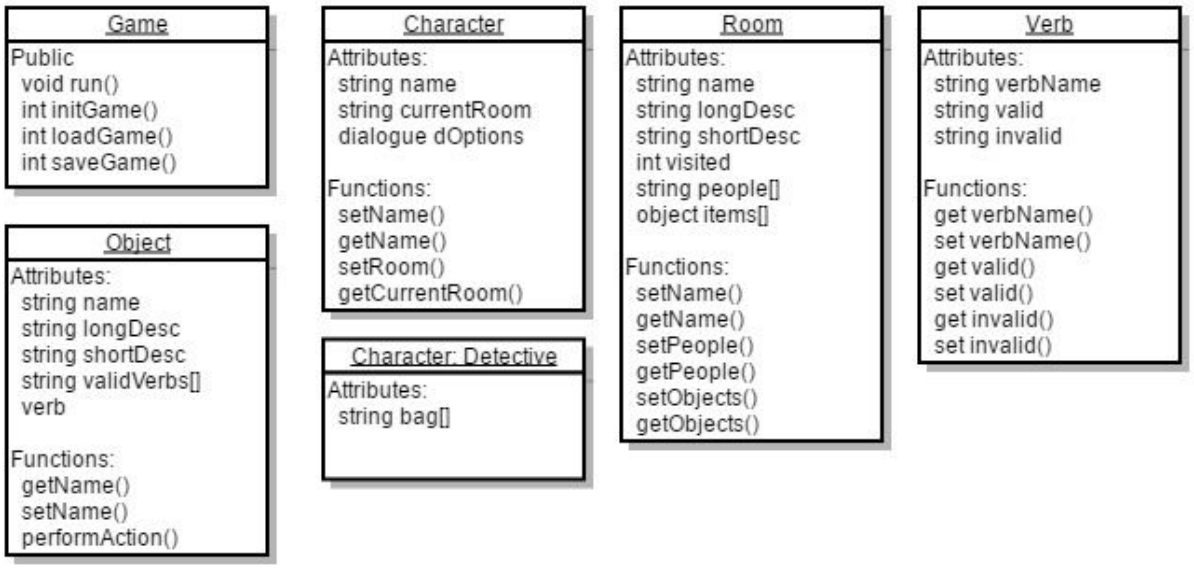


Game State Machine Diagram:



This diagram shows various rooms and paths to those rooms via the “go” action. Actions on characters and objects in a room keep the player in that room (state), while actions to relocate will transition the player to another room.

Class Diagram:



Conclusion:

Our team will spend at least 300 hours over eight weeks developing our murder mystery game. The structure of the game will allow us to provide a murder mystery style adventure game for the command line interface that is highly modular and will allow for easy modification. The parser will allow the use of natural language to be used to interact with the game environment. Our use of C++ will provide control and efficiency in the implementation of the game.