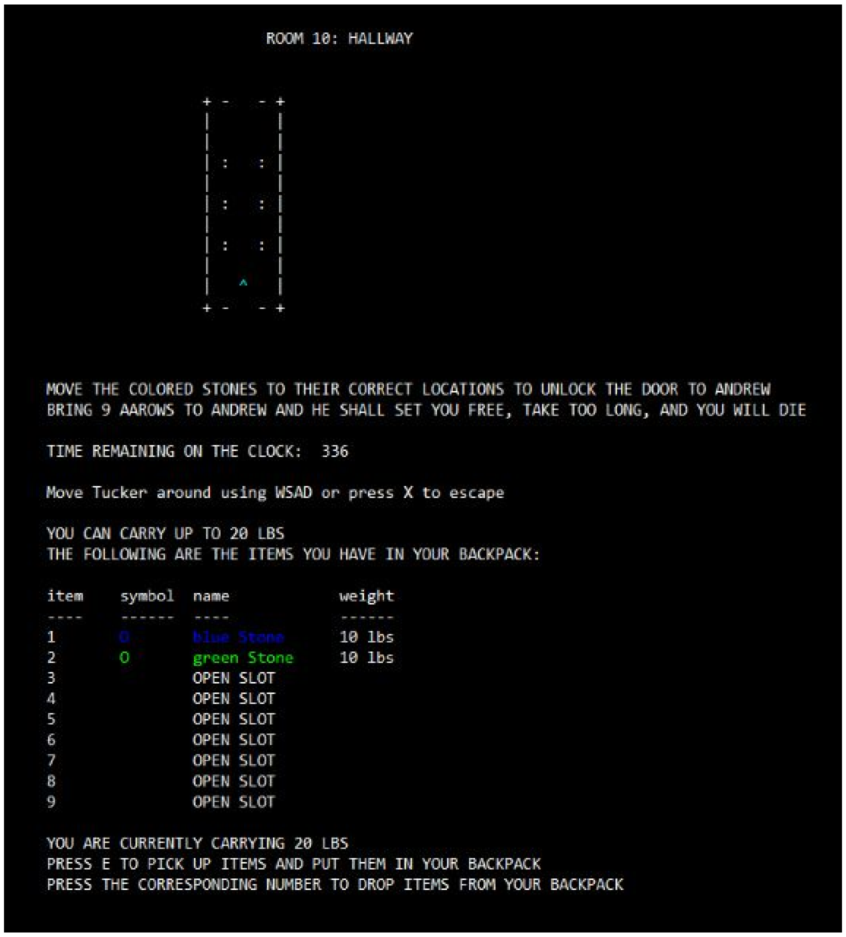
Team Rigel
Tucker Walker
Marisa Rea
David Pipitone

<Adventure Game>

15th January 2019

OVERVIEW

<Adventure Game> will be a top-down Rougelike written in C++ and run in a Linux environment. The user will be able to control a character to navigate and interact with a 2-D text-based world. Interactive windows will allow the user to switch between real-time world navigation (*figure 1a)*, selecting between menu items (*figure 1c)*, and viewing narrative text (*figure 1b)*. The setting, style, and objective of the game will be developed by the team throughout the course of the project.



```
                 ROOM 10: HALLWAY


             + -   - +
             |       |
             |       |
             | :   : |
             |       |
             | :   : |
             |       |
             | :   : |
             |       |
             |   ^   |
             + -   - +



MOVE THE COLORED STONES TO THEIR CORRECT LOCATIONS TO UNLOCK THE DOOR TO ANDREW
BRING 9 AAROWS TO ANDREW AND HE SHALL SET YOU FREE, TAKE TOO LONG, AND YOU WILL DIE

TIME REMAINING ON THE CLOCK:  336

Move Tucker around using WSAD or press X to escape

YOU CAN CARRY UP TO 20 LBS
THE FOLLOWING ARE THE ITEMS YOU HAVE IN YOUR BACKPACK:

item    symbol  name           weight
----    ------  ----           ------
1       O       blue Stone     10 lbs
2       O       green Stone    10 lbs
3               OPEN SLOT
4               OPEN SLOT
5               OPEN SLOT
6               OPEN SLOT
7               OPEN SLOT
8               OPEN SLOT
9               OPEN SLOT

YOU ARE CURRENTLY CARRYING 20 LBS
PRESS E TO PICK UP ITEMS AND PUT THEM IN YOUR BACKPACK
PRESS THE CORRESPONDING NUMBER TO DROP ITEMS FROM YOUR BACKPACK
```

Figure 1 - a prototype of <Adventure Game>

(a) *The user can control a player, represented here by a cyan carrot symbol, around a 2-d text based room. This takes place in an interactive 'world' window at the top of the screen.*

(b) *The user is able to read information about the game. This includes game instructions and room descriptions. This is shown in the middle of the screen.*

(c) *The user is able to select between various menu items shown in an inventory window at the bottom of the screen.*

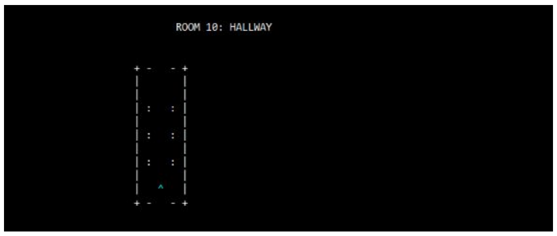
USER PERSPECTIVE

<Adventure Game>'s plot and variously themed rooms will resonate most strongly with the target user: lifelong fans of video games. The game will transport the user into a familiar and humorous stroll through variously themed rooms inspired by and paying homage to their favorite video games. Classic tropes, mechanics, and inside jokes will evoke a sense of nostalgia and joy for the user. The top-down Roguelike game display and mechanics will feel intuitive to the target user. The simplified graphics and world interaction will aid in the vintage feel and nostalgic experience that the game aims to provide, facilitating the immersion of the user into the vibrant memories of their gaming past rather than an excessively graphic, modern day clone.

GOALS

The following is a list of the major goals that <Adventure Game> seeks to accomplish:

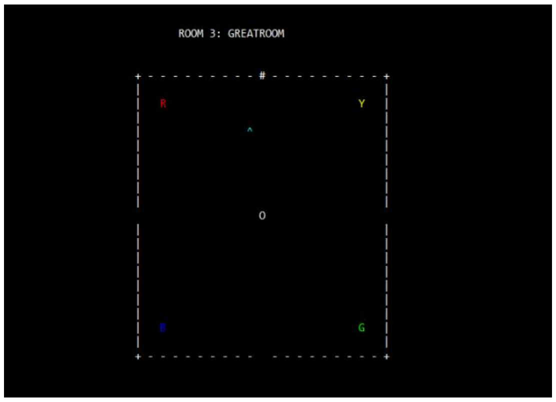
1. **Windows** - The user can switch context between various interactive *Window*s that affect the state and/or rendering of <Adventure Game>. At minimum, these *Windows* include a *World*, *Inventory*, *Status*, and a *Narrative Window*. Context switching between these *Window*s will be triggered by user input. See *figure 1a-c* for prototype depictions of these *Window*s.
2. **World** - Within the context of the *World Window* (*figure 1a)*, the user can control a *Player* to navigate a static *World* consisting of *Space*s. In real-time, the *Player* is able to move between *Spaces* within the *World* by various means such as doors.
3. **Space** - A *Space* is a static set of *Tiles* within the *World* displayed in the *World Window*. Only one *Space* can be displayed in the *World Window* at a time. A *Space* can hold various *Game Objects* and *World Objects* within its *Tile*set, to include the *Player*.



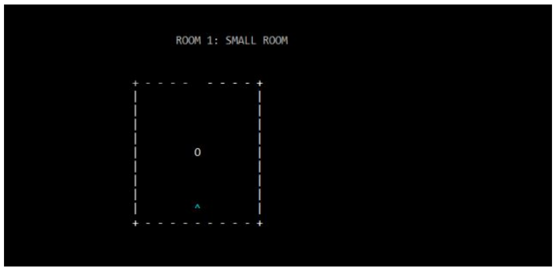
```
        ROOM 10: HALLWAY

+-  -+
|   |
|   |
|: :|
|   |
|: :|
|   |
|: :|
|   |
| ^ |
+-  -+
```

Figure 2 - Spaces

(a) *a narrow hallway space consisting of four walls, game objects represented as : symbols, open doors to the North and South, and the player represented by a ^*

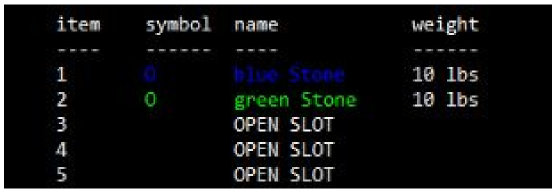


(b) *a large greatroom with four walls, world objects represented by colored letters, a locked door to the North, and open doors to the South, East, and West.*



(c) *a small room containing a single game object represented by the letter 'O' and an open door to the North.*

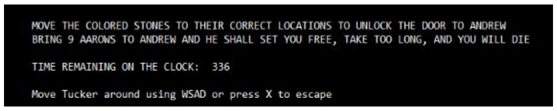
4. **Tile** - A *Tile* is any x/y coordinate within the *World Window*. A *Tile* may be empty or it may not, instead holding a value that symbolizes various *World Objects* such as concrete walls, water, and grass. Each *Tile* type may or may not have an effect on how the *Player* behaves. For example, a *Player* may not be able to move onto a wall *Tile* where it could move onto a grass *Tile*. Various *Game Objects* such as the *Player* and *Item*s may exist on and be rendered on top of a *Tile*. *Tiles* will be represented and rendered as a combination of symbols and colors. For example, grass might be represented as a green background and desert by a yellow background. Furthermore, walls might be represented as black characters such as +, |, -, and / while sand ripples in a desert *Tile* might be represented by black ~ symbols.
5. **Status** - The *Status Window*, a thin window at the top of the game screen, will communicate to the user various *Status* information about the game. The *Status* information displayed will communicate *Player* information such as : health, mana, and/or selected item. It will also display the *current Space* rendered by the Game.
6. **Inventory** - Within the context of the *Inventory Window* (*figure 1c*), the user will be able to view and select between various *Items* collected in the *Player's* inventory. By selecting an *Item*, the user will be able to read a description related to that item in the *Narrative Window*. The user will also be able to augment the *Player*'s interactions in the *World* when certain *Items* are selected. For example, the *Player* may be able to open a door within the context of the *World* when a Key *Item* is selected in the *Player*'s *Inventory*.



```
item    symbol  name           weight
----    ------  ----           ------
1       O       blue Stone     10 lbs
2       O       green Stone    10 lbs
3               OPEN SLOT
4               OPEN SLOT
5               OPEN SLOT
```

Figure 3 - Inventory
Displayed here is an example of what a Player's inventory might look like, to include inventory Item characteristics such as weight, color, and graphical symbol

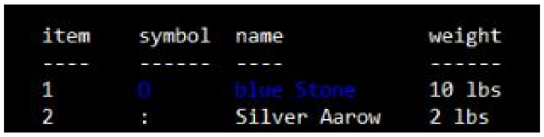
7. **Narrative** - The *Narrative Window* (*figure 1b*) will be a read-only window displayed to the user. The contents of this *Window* will contain a *Narrative* that is updated based off of user-input, game logic, and inventory selection. For example, in the *World*, if the player moves onto a *Tile* that represents a trap, the *Narrative* displayed might be "*you sprung a trap!"*. If the player elects to use a special function provided by a selected *Item*, such as a key, a different *Narrative* might be displayed such as *"you unlocked a door"* or "*that item does not work here"*. Furthermore, when the user selects an item in the *Inventory Window*, a description of the *Item* will be displayed in the *Narrative*. For example, *"This is a key, it is used to lock and unlock doors"*.



```
MOVE THE COLORED STONES TO THEIR CORRECT LOCATIONS TO UNLOCK THE DOOR TO ANDREW
BRING 9 AAROWS TO ANDREW AND HE SHALL SET YOU FREE, TAKE TOO LONG, AND YOU WILL DIE

TIME REMAINING ON THE CLOCK:  336

Move Tucker around using WSAD or press X to escape
```

Figure 4 - Narrative
Displayed here is narrative text explaining a goal of <Adventure Game>, a timer, and instructions on how to move the Player Object

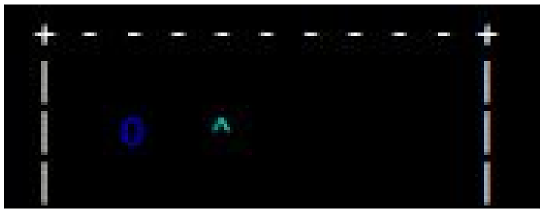
8. **Game Objects** - A *Game Object* is any non-static entity that exists within the context of the *World*. *Game Objects* may be positioned on a *Tile* or in the *Player*'s *Inventory*. A *Game Object* may have other dynamic properties outside of its position such as *"locked"* or *"unlocked"* in the case of a door or chest. There are many potential types of *Game Objects*. At minimum, <Adventure Game> has at least one *Player* and various *Item* and *World Objects*.



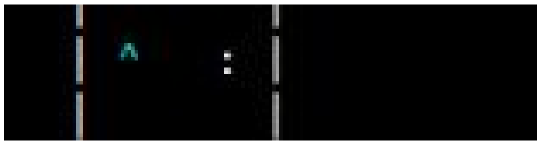
```
item    symbol  name           weight
----    ------  ----           ------
1       O       blue Stone     10 lbs
2       :       Silver Aarow   2 lbs
```

Figure 5 - Game Objects

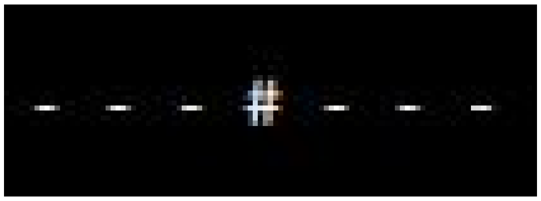
(a) *Items in a player's Inventory, specifically a blue stone and a Silver Arrow (looks like Arrow is spelled wrong!)*



(b) *the Player and a blue stone shown from Figure 5a represented in the World.*



(c) *a silver arrow from the inventory in Figure 5a represented in the world next to the player.*



***(d)** a locked door represented by the # symbol.*

9. **Player Object** - The *Player* is the single most important *Game Object*. The *Player* is a *Game Object* which the user directly manipulates and uses to interact with the *World* and other *Game Objects*. For example, the user can move the *Player* around within the *World* and augment the *Player*'s abilities within the *World* by selecting various *Items*. See *5b* and *5c* for graphical representations of the *Player* within the context of the *World*
10. **Item Objects** - *Item*s are *Game Objects* that are not directly controlled by the user, but controlled indirectly by means of the *Player* and the *Game* itself. *Item*s may take up space in the *World* on a *Tile* or in the *Player*'s *Inventory*. *Items* in the *World* can be picked up into the *Player's Inventory*, and *Items* in the *Player's Inventory* can be dropped into the *World*. *Item*s may or may not have additional properties such as a weight, special functionalities (like lock/unlock door), and specific colors/symbols used to represent them graphically. See *Figure 5a-c* for representations of item objects, such as stones and arrows.
11. **World Objects -** *World Objects* are objects within the *World* that have dynamic properties, but cannot be added to the *Player's Inventory*. For example, *World Objects* might consist of doors that lock/unlock, movable blocks, and switches that trigger various narratives and/or change something in <Adventure Game>. See Figure 5d for an example of a locked door world object.

SPECIFICATIONS

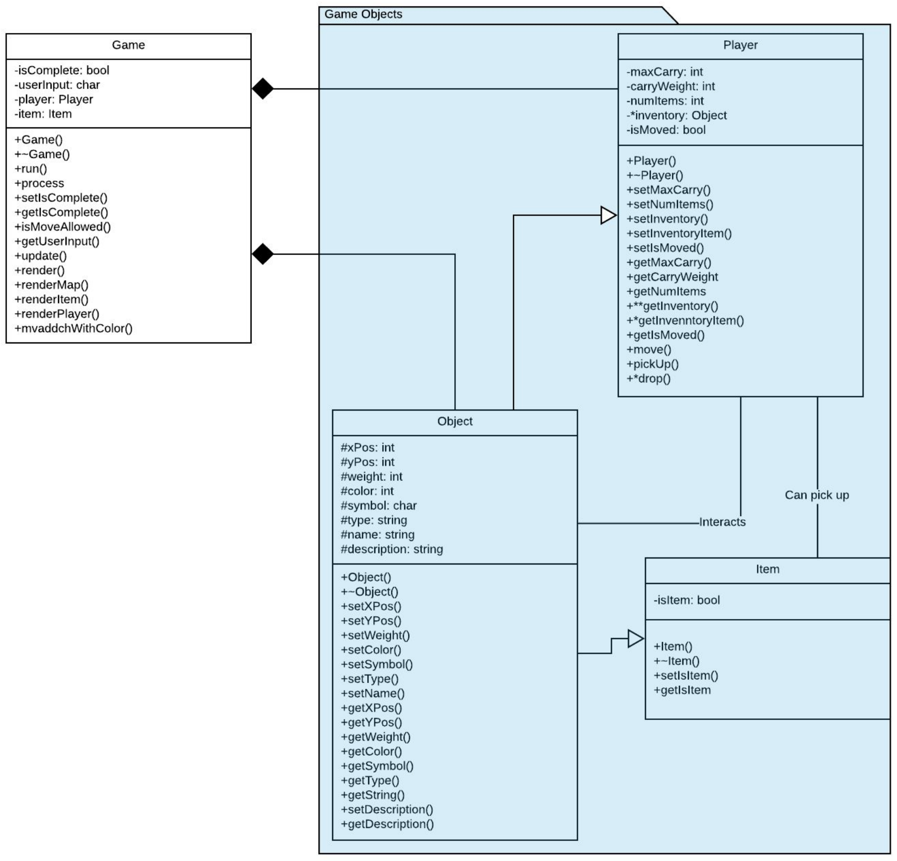
Abstract Classes

<Adventure Game> will be written in C++ and will include but not be limited to the following abstract classes.

1. **Window** - An abstract class that has a height, width, position within the terminal, and is capable of rendering game state. Concrete subclasses include:
    a. Inventory_Window
    b. World_Window
    c. Narrative_Window
    d. Status_Window
2. **World** - A concrete class which consists of a collection of Spaces and Game Objects and which may be rendered in a Window.

3. **Space** - an abstract class which holds shape and tile information about a space. Potential concrete subclasses of a generic space might be
    a. Square Rooms
    b. Circular Rooms
    c. Desert Spaces
    d. Grassland Spaces
4. **Inventory** - A concrete class which serves as a container for Item objects
5. **Narrative** - A concrete class which contains messages that may be displayed to the user in the Narrative window. The narrative class might be manipulated to display text slowly, fast, or instantly.
6. **Game_Object** an abstract class defining parameters common to all game objects, such as location (*Space* and x/y coordinate of a *Tile*) or game location (*Inventory* or *World*) examples of potential subclasses are:
    a. Player_Object
    b. Item_Object
    c. Enemy_Object

UML Diagram



TECHNOLOGY REQUIREMENTS

<Adventure Game> will be written in **C++11** and compiled in **Linux** using **g++**. Specifically, it will be capable of compiling and running on one of OSU's three flip servers. It will rely heavily on the ***ncurses* library** in order to render graphical data to the user, taking advantage of various built-in functionalities such as *Windows*, *Colors*, and *Cursor Manipulation*. **This Linux Journal article** will serve as a starting point for programming with *ncurses*.

MILESTONES

Project Planning

User Story: As a developer on *Team Rigel*, I want a shared plan on what <Adventure Game> consists of, its technology requirements, and how we as a team are going to build it.

Acceptance Criteria: *Team Rigel* has generated a robust Project Proposal for <Adventure Game> outlining Overview, Goals, Specifications, Technology Requirements, Milestones, and Resources.

User Story: As a developer on *Team Rigel*, I want to set up my coding environment so that I can begin development on <Adventure Game> with shared access to our team's Planning / Coding repositories.

Acceptance Criteria:

- *Team Rigel* has a shared GitHub repository for its code base
- *Team Rigel* has a shared Google Docs repository for it's Planning documents
- I have a development environment set up that allows me to compile and run C++11 code from the <Adventure Game> repository using g++ in Linux
- I have access to *Team Rigel's* <Adventure Game> code repository and have tested the following commands.
    - git pull origin <branch>
    - git push origin <branch>
    - git rebase <branch>
    - Generating a pull request on a branch off of <Adventure Game>'s master
    - Conducting Code Review with my team on GitHub
    - Merging Code via GitHub.

Windows

User Story: As a User, when I run <Adventure Game>, I want to be able to see at least four different windows on my screen, switch between the windows, and manipulate them in some way.

Acceptance Criteria: When run, <Adventure Game>:

- Generates and renders a World Window
- Generates and renders a Narrative Window
- Generates and renders an Inventory Window
- Generates and renders a Status Window
- Allows the user to switch context between the World and Inventory Window
- Allows the user to manipulate what is rendered within the World and/or Inventory Windows based off of which Window is currently selected.
- Narrative and Status windows are updated based off of what happens in the World and/or Inventory Windows.

Base Game

User Story: As a user, I want to be able to move a playable character around in a single space, interact with objects within that space, and see information about my interactions.

Acceptance Criteria: The game generates

- A world window that renders a single space, a player, one item object, and one world object.
    - The space contains tiles that the player can move into and tiles that the player cannot move into
    - The space contains at least one world object that the player can interact with
    - The player is able to pick up and drop at least one item object in the space.
        - When the player picks up an item from the space, it is no longer rendered on any tile in that space.
        - When the player drops an item into the space, it is rendered on a tile in that space.
- An inventory window
    - When the player picks up an item from the world, it is shown in the inventory window
    - When the player drops an item from the inventory into the world, it is removed from the inventory window

- A narrative window
    - The narrative window displays textual information about the room when the player is moving about
    - The narrative window displays textual information about the current item selected when an item is selected in the inventory menu
    - The narrative window displays special information when the player interacts with a world object.
- A status window
    - The currently selected inventory item is displayed in the Status window; this is updated when the player selects a different item in the inventory.
    - The current space is named in the Status window.

Subclass Generation

User Story: As a user, I want to be able to move between various spaces in <Adventure Game> and interact with different types of game objects.

Acceptance Criteria: <Adventure Game> allows the user to...

- Move into and out of various spaces by means of doors (a world object) which point to a tile in another space.
- Pick up and drop at least 8 unique Item Objects
- Interact with at least 3 unique World Objects

Puzzles

User Story: As a user, I want to have a specific goal I need to fulfil within <Adventure Game>, and solve puzzles to achieve that goal.

Acceptance Criteria: <Adventure Game>...

- Restricts the player's inventory to a certain number of slots and/or weight
- Requires that the Player solve at least 3 puzzles that involve
    - Moveable blocks
    - Locked Doors
    - Switches
- Each of the 3 puzzles are restricted to a single space
- Contains Items with special functions which augment the Player's interaction within the world. (examples: *bomb, key*)

Expanded Puzzles

User Story: As a user, I want to solve a number of puzzles throughout <Adventure Game> in order to achieve the goal.

Acceptance Criteria: <Adventure Game>...

- At least 1 puzzle is not restricted to a single space but must be solved by interacting with multiple spaces throughout the world.
- 3 additional puzzles are generated which stand in the way of the player achieving the goal.

Independent Object Movement

User Story: As a user, I want to be able to interact with various world and game objects that move in real-time.

Acceptance Criteria: <Adventure Game> contains...

- A timer which displays the current time and is updated even when the player isn't moving.
- A world object that moves independently of the player.

Enemies

User Story: As a user, I want to be able to fight my way through enemies in order to achieve the goal of the game

Acceptance Criteria: <Adventure Game> contains...

- At least one enemy type that can damage and/or limit the player's ability to achieve the game's goal.
- The Player has a health buffer that can be affected by enemies.

Game Manual

User Story: As a user, I want to be able to be able to play the game, accomplish the goal (beat the game) and understand the controls.

Acceptance Criteria: <Adventure Game> manual contains...

- A compile and running guide.
- A description of the goal of the game.

- A controls guide that explains how to control the player character and how the character interacts with objects within the game.
- A walkthrough.

TASK SCHEDULING AND TEAM CONTRIBUTIONS

Strategy

Weekly Sprints: In order to facilitate creativity and allow for the opportunity to pursue various stretch goals, Rigel has elected to follow the below schedule with details and assignees determined at weekly meetings.

Schedule

Week 2 - January 13 - 19: Finalize Plot

- Overall plot
- Beginning
- Predetermined plot points from beginning to end
- End
- **Project Plan Submission (**Submitted by all**)**

Week 3 January 20 - 26: Generate Base Game

- Screen Layout
  - World Window
  - Narration Window
  - Inventory Window
  - Status Window
  - Item desc
  - Text interaction
- Player
- HP
- Room Mechanics
- Item Mechanics
  - World Items
  - Obtainable Items
- Inventory Mechanics
  - Weight Limit
  - Item equipment
- ncurses Mechanics
  - Player Movement
  - ColorR

Week 4 January 27 - February 02: Build Expanded Game

- Rooms (2)
    - Layout (how the rooms are connected)
    - Three new rooms completed
- Items
    - Create a least one unique item
- Documentation Updated with Changes

Optional: Explore stretch goals

Week 5 February 03 - 09: Build Expanded Game

- Rooms (5)
    - Three new rooms completed
- Items
    - Create a least one unique item
- Documentation Updated with Changes

Optional: Explore stretch goals

Week 6 February 10 - 16: Build Expanded Game

- Rooms (8)
    - Three new rooms completed
- Items
    - Create a least one unique item
- Documentation Updated with Changes
- **Mid-Point Project Check Submitted (**Submitted by Marisa**)**

Optional: Explore stretch goals

Week 7 February 17 - 23: Complete Expanded Game and Pursue Stretch Goals

- Progress Check
    - Will Base Game be completed by deadline?
    - If yes, what stretch goals will be implemented
- Rooms
    - Three new rooms completed
- Items
    - Create a least one unique item
- Stretch Goal
    - Develop selected stretch goal(s)
- Documentation Updated with Changes

Week 7 February 24 - March - 02: Complete Expanded Game and Pursue Stretch Goals

- Rooms
  - Three new rooms completed
- Items
  - Create a least one unique item
- Stretch Goal
  - Develop selected stretch goal(s)
- Documentation Updated with Changes

Week 8 March 03 - 09: Complete Expanded Game and Pursue Stretch Goals

- Rooms
  - Three new rooms completed
- Items
  - Create a least one unique item
- Stretch Goal
  - Develop selected stretch goal(s)
- Documentation Updated with Changes

Week 9 March 10 - 16: Expanded Game Complete / Finalize Stretch Goals

- Stretch Goal
  - Finalize selected stretch goal(s)
- Plan Project Poster
- Documentation Updated with Changes
- Test and Polish Final Product

Week 10 March 17 - 23: Finalize Complete Game

- Test and Polish Final Product
- Documentation Updated with Changes
- Complete Project Poster
- **Project Poster, Final Report, and Project Demonstration (**Submitted by David**)**

Stretch Goals

- Save Feature
- Game Menu
  - New Game
  - Continue
  - Controls
  - About
- Enemies
  - AI
- Player

- Classes

Time Expectations

Within each weekly sprint, each team member is expected to work in excess of 10 hours on their assigned tasks which work toward completion of Milestone goals. Beyond these minimum 10 hours, or when a task is completed early, team members are encouraged to experiment and pursue the development of stretch goals, including new stretch goals that result from experimentation. Those which develop successfully and prove to compliment the Milestone version of the game will be examined after the Mid-Point Check and implemented in the final version of the game.

CONCLUSION

<Adventure Game> will provide a joyful and nostalgic gaming experience for the target customer. With the schedule outlined, Team Rigel will successfully be able to complete the base and expanded game, as well as develop and execute a number of added features as pursued through stretch goals within a minimum of 300 hours and before the deadline of March 17th, 2019.

RESOURCES

ncurses

ncurses **library**

Creating an Adventure Game with ncurses

C++11

C++11

Makefiles

g++