Dungeon Crawler

By Ethan Jackson, Madelyn Kempka, Michael Plumlee, and Tylar Wolff

Project Overview & Objectives

- Overview: Develop a text-based dungeon crawler game with random dungeon generation and combat.

- Objectives:
  1) Use classes for the player and monsters.
  2) Implement random dungeon generation with rooms that may contain monsters or treasures.
  3) Allow combat with random damage values.
  4) Display the player’s stats (HP, inventory, and score) on demand.

Key Features & Functionalities

- Classes: Room, monsters, items, player, shop
- Two major gameplay loops:
  - Combat Loop
  - Exploration Loop
- Ascii Art and Lore

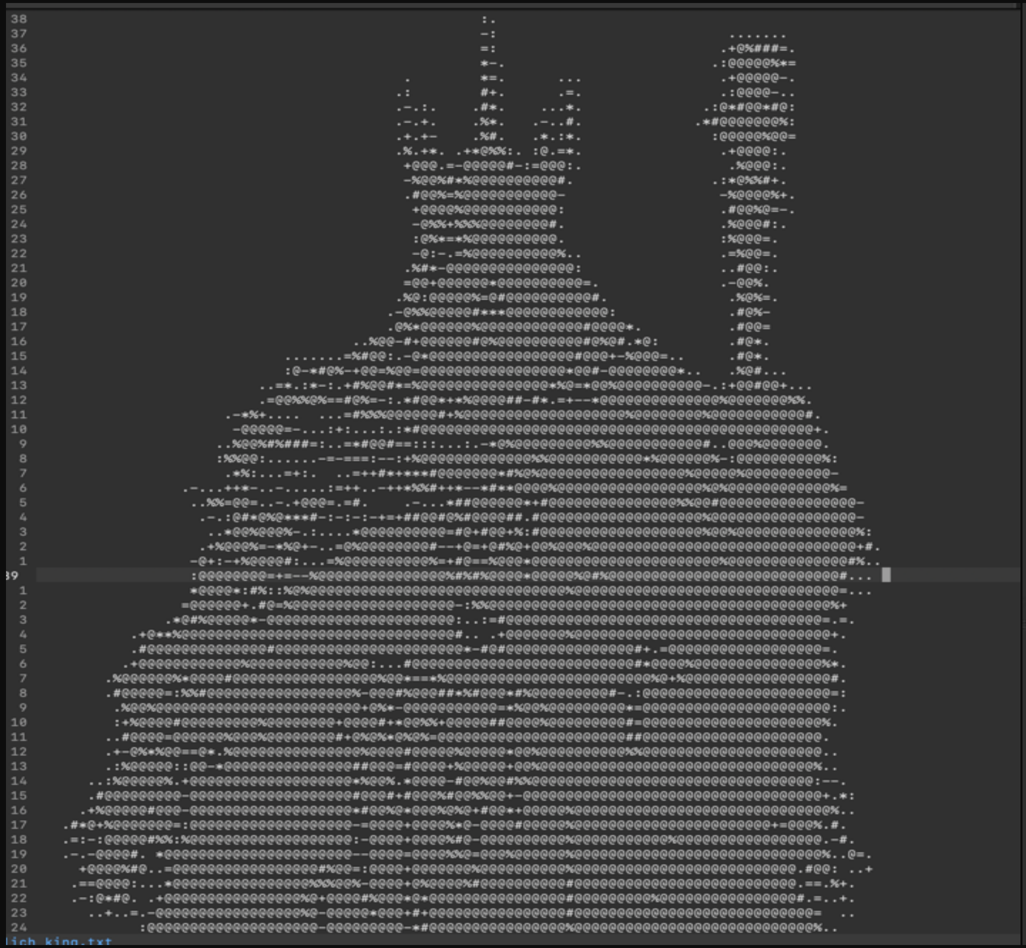
Room Class

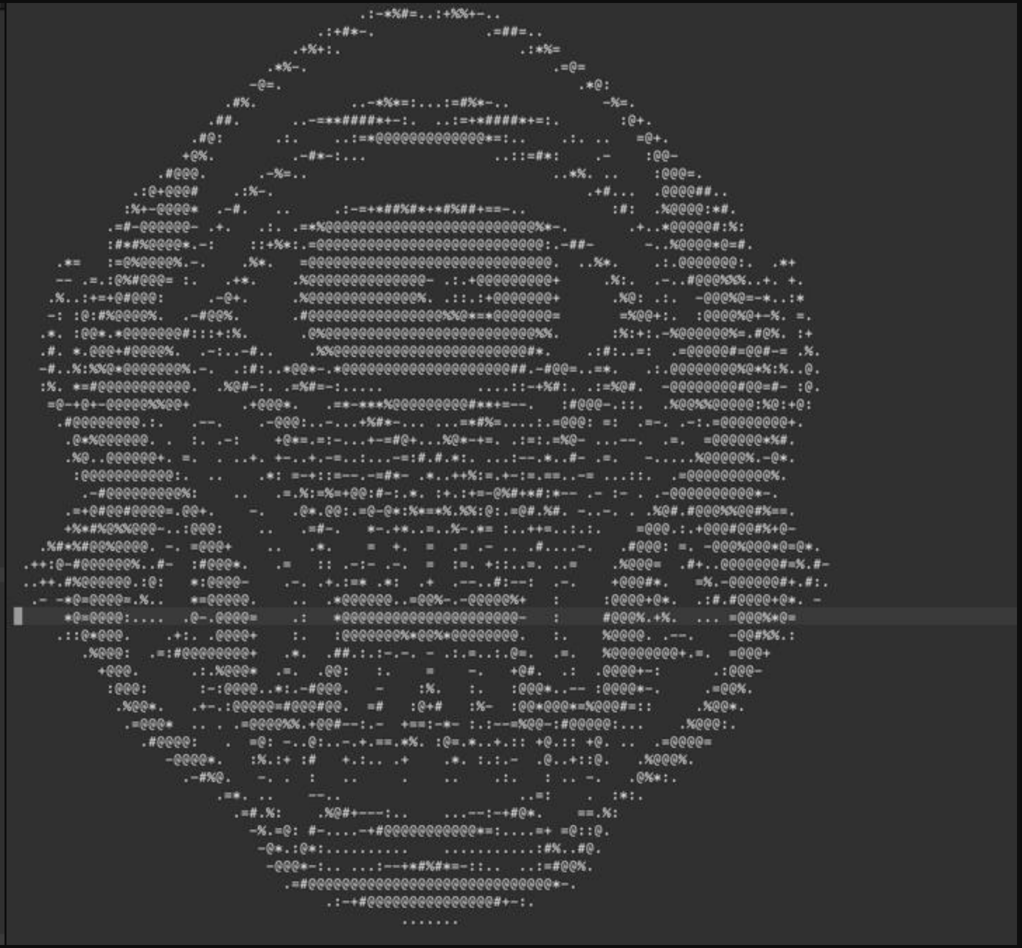
- The room class models a room with features a typical room would have in a dungeon game: rooms, treasure, and monsters
    - Uses class specific boolean values such as Room.isMonster and Room.isTreasure
- Constructors were added to allow the developers to fully customize the room to be empty or full of whatever attributes they choose
    - Boolean logic for the rooms are 50/50 chance using coinflip logic.
- The **describe()** function thoroughly describes the room and it’s features to immerse the user fully into the game
- By using a room class, we can build rooms in this game without hard coding each individual one.

Monster Class

- The monster class establishes the various attributes of the monsters, those being HP, Attack Bonus, Armor Class, gold, minimum Spawn Depth, Spawn Weight, damage roll data, Ascii Art, and a Biography.
- The data can be edited inside of a csv file, easily allowing for the creation and balancing of the monsters.
- The examine command allows for some ascii art, as well as a humorous biography for the monster to pop up.
- The monsters can be loaded into the game, they're health decreasing with every attack landed. A certain amount of gold is earned upon their death. They also have the ability to roll for an attack against the player every turn.

Monster Ascii Showcase





Art to Ascii generator, Raw literals

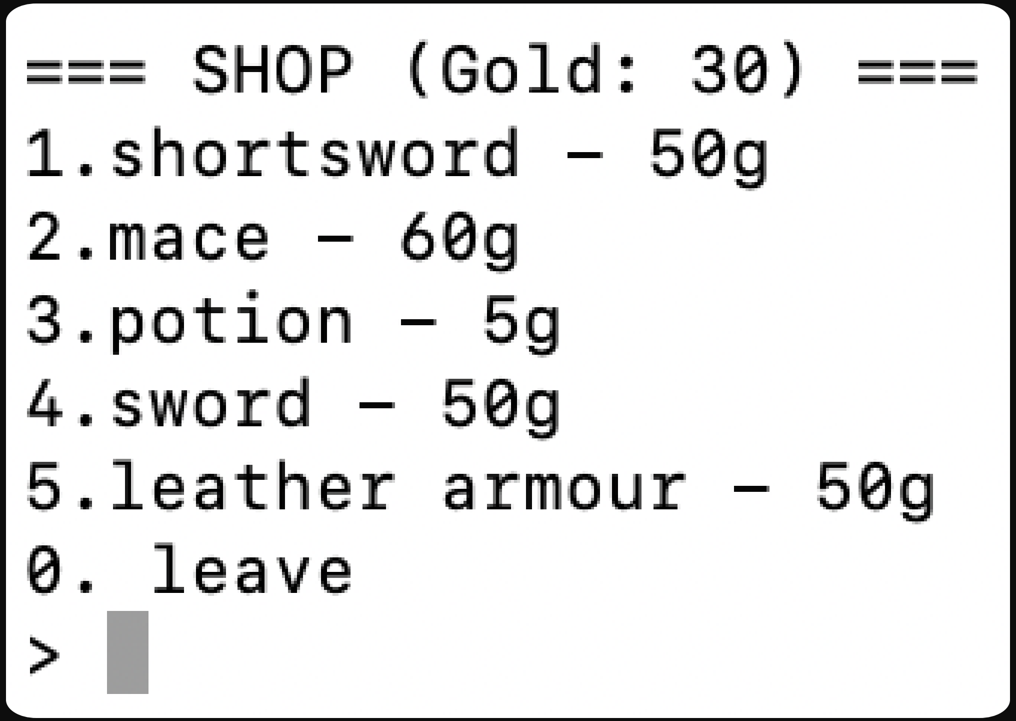
Item Class

- The Item Class was developed to handle all Item logic for the game.
- Items contain attributes such as
  - Damage roll
  - Healing value
  - Attack bonus
  - Defensive (AC) bonus
- There are then Getter and Setter functions to get these attributes for use in functions such as player.damageRoll() and player.attackRoll()
- Items are sideloaded into the game via CSV and parsed using ItemLoader.cpp

Player Class

- The Player class was designed to handle all player logic
- Players contain attributes such as
  - Base AC
  - MaxHP
  - Weaponslot/Armourslot
  - An inventory vector<Items>
  - Gold
- There are then Getter and Setter functions to get these attributes for use in gameplay, as well as some behaviour functions such as damageRoll().

Shop Class



```
=== SHOP (Gold: 30) ===
1.shortsword - 50g
2.mace - 60g
3.potion - 5g
4.sword - 50g
5.leather armour - 50g
0. leave
> 
```

- Allows the player to buy items every 5 floors.
- Randomized available items.
- Items get more powerful as you make it further in the game.
    - Uses Item rarity value and compares that to the value ofcurrentDepth to make rarer items more likely to appear.

Challenges & Solutions

- During development, there was a lot of back and forth with learning how to implement classes.
  - Learning about how public and private work to prevent mutation of items/monsters
- We also learned how to use pointers at a basic level.
  - Pointers to our arrays of Monsters and Items allow us to create copies from the prototype list to create monsters or items for our player.
- Developing the combat loop and main gameplay loop helped in reinforcing how to navigate the logic and debugging infinite loop issues as well as fall-through logic.

Time for a demonstration!