Text Adventures

by
Alex McNabb
Kaleb Burdin
Bethany Colvin

Components of Code

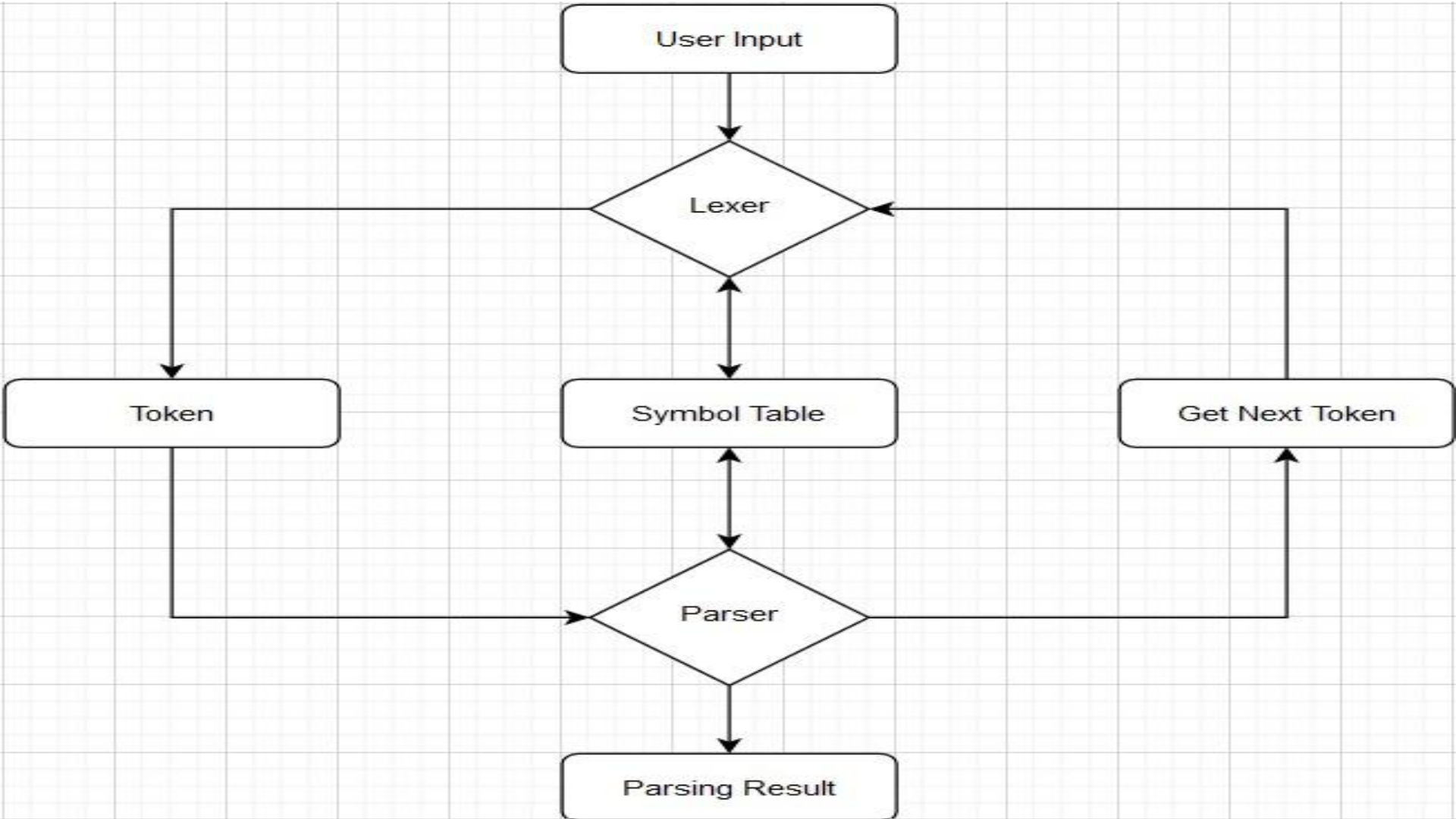
-Lexer and Parser

-NPC Dictionary

-Player Data

-Game Script

-GUI


User Input
Lexer
Token
Symbol Table
Get Next Token
Parser
Parsing Result

Lexer

```python
# Define what a word is
def t_WORD(t):
    r'[a-zA-Z]+'
    t.value = str(t.value)
    if t.value in list(nouns.values()):
        t.type = nouns.get(t.value, 'NOUN') # Convert type from a word to a noun
    if t.value in list(directions.values()):
        t.type = directions.get(t.value, 'DIRECTION')  # Convert type from a word to a direction
    if t.value in list(articles.values()):
        t.type = articles.get(t.value, 'ARTICLE')  # Convert type from a word to an article
    if t.value in list(attacking.values()):
        t.type = attacking.get(t.value, 'ATTACKING')  # Convert type from a word to an attack command
    if t.value in list(moving.values()):
        t.type = moving.get(t.value, 'MOVING')  # Convert type from a word to a movement
    if t.value in list(adjectives.values()):
        t.type = adjectives.get(t.value, 'ADJECTIVE')  # Convert type from a word to an adjective
    if t.value in list(sneaking.values()):
        t.type = sneaking.get(t.value, 'SNEAKING')  # Convert type from a word to a sneak command
    if t.value in list(grabbing.values()):
        t.type = grabbing.get(t.value, 'GRABBING')  # Convert type from a word to a grab command
    if t.value in list(dropping.values()):
        t.type = dropping.get(t.value, 'DROPPING')  # Convert type from a word to a drop command
    if t.value in list(using.values()):
        t.type = using.get(t.value, 'USING')  # Convert type from a word to a use command
    return t
```

```python
def t_error(t):
    print(f'Illegal character: {t.value[0]!r}')
    t.lexer.skip(1)

# Ignore whitespace
t_ignore = ' '

user_input = input()
lexer = lex()
lexer.input(user_input.upper())

for token in lexer:
    print(token)
    values.append(token.value) # Append token value to a usable list
    types.append(token.type) # Append token type to a usable list
```

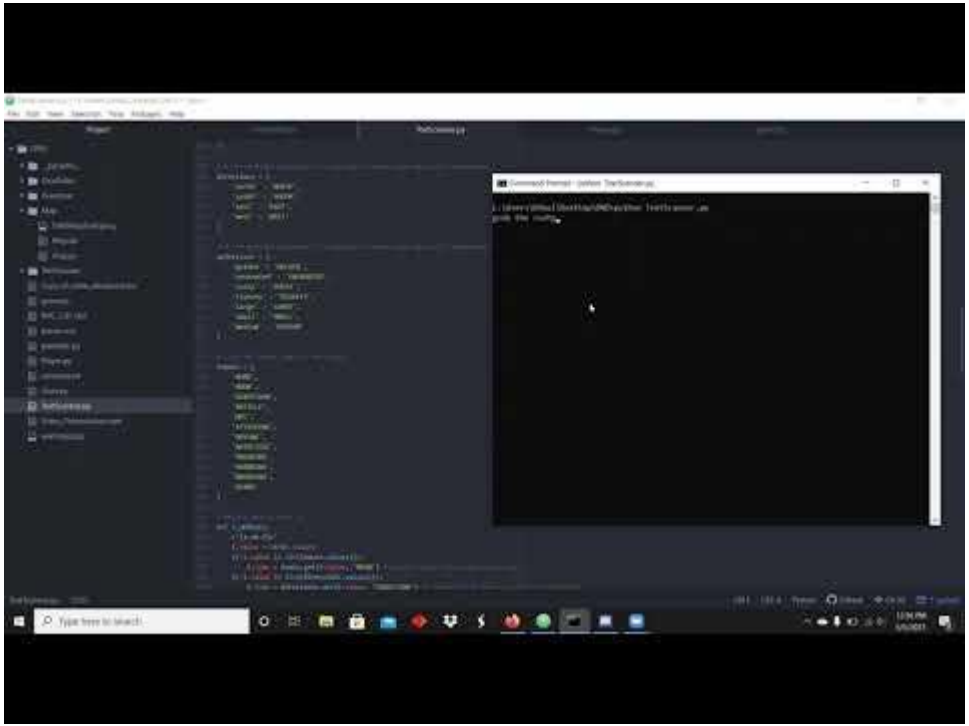
Parser

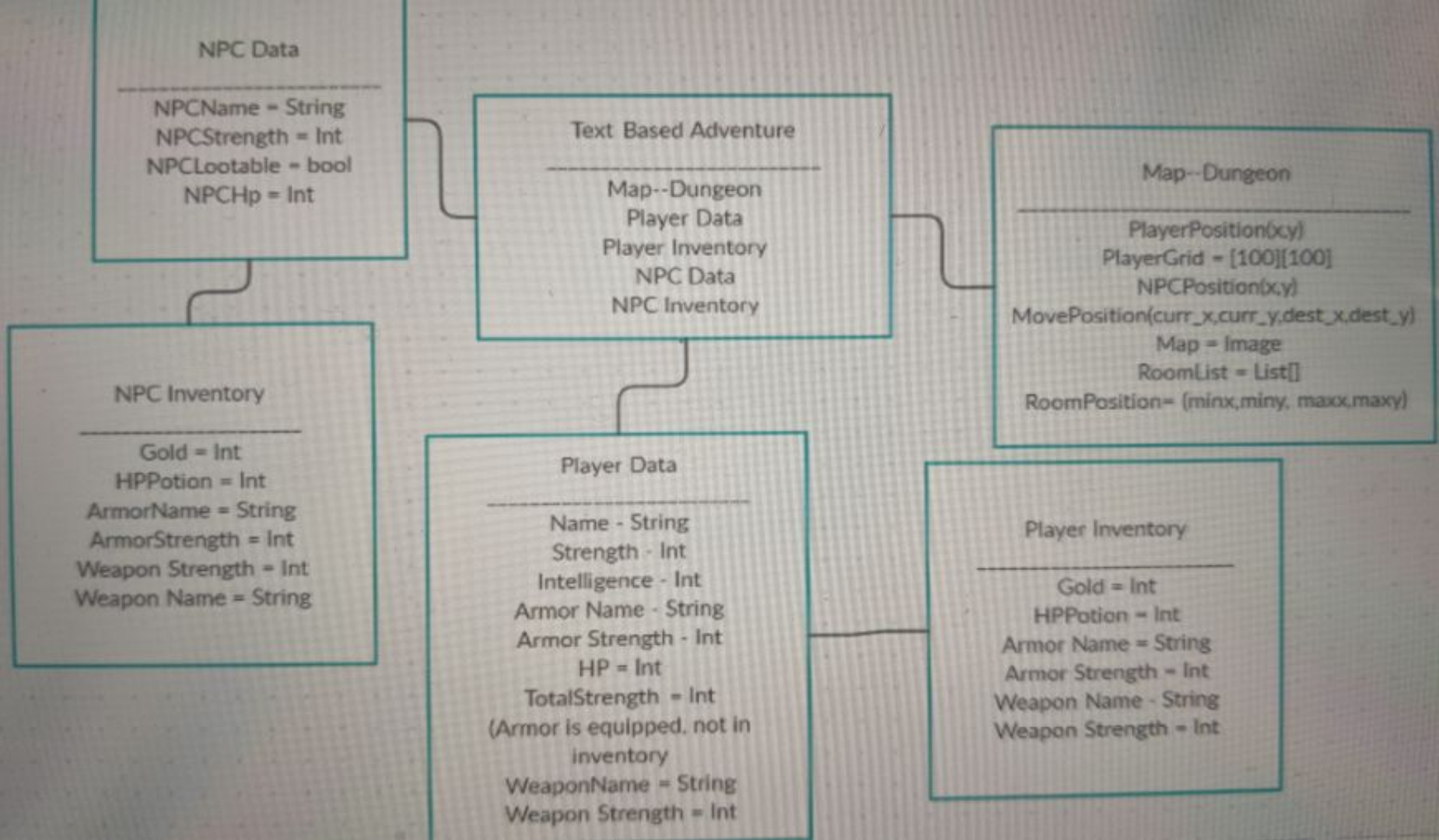
```
def p_action(p):
    '''
    command : fuller NOUN
            | fuller NPC
            | command DIRECTION
            | fuller WORD
    '''
    p[0] = p[2]

def p_fuller(p):
    '''
    fuller : full WORD
           | full ADJECTIVE
           | full ARTICLE
    '''
    p[0] = p[2]

def p_full(p):
    '''
    full : command WORD
         | command ADJECTIVE
         | command ARTICLE
    '''
    p[0] = p[2]

# Use command
def p_use(p):
    '''
    command : USING
    '''
    p[0] = {"USE", p[1]}
    print("This is a USE command!")
    global returnedValue
    returnedValue = "USE"
```

```
def p_grab(p):
    '''
    command : GRABBING
    '''
    p[0] = {"GRAB", p[1]}
    print("This is a GRAB command!")
    global returnedValue
    returnedValue = "GRAB"

# Drop command
def p_drop(p):
    '''
    command : DROPPING
    '''
    p[0] = {"DROP", p[1]}
    print("This is a DROP command!")
    global returnedValue
    returnedValue = "DROP"

# Sneak command
def p_sneak(p):
    '''
    command : SNEAKING
    '''
    p[0] = {"SNEAK", p[1]}
    print("This is a SNEAK command!")
    global returnedValue
    returnedValue = "SNEAK"

# Attack command
def p_attack(p):
    '''
    command : ATTACKING
    '''
    p[0] = {"ATTACK", p[1]}
    print("This is an ATTACK command!")
    global returnedValue
    returnedValue = "ATTACK"
```

```
# Move command
def p_move(p):
    '''
    command : MOVING
    '''
    p[0] = {"MOVE", p[1]}
    print("This is a MOVE command!")
    global returnedValue
    returnedValue = "MOVE"

def p_error(p):
    return

output = yacc().parse(user_input.upper())

def main():
    return returnedValue

if __name__ == '__main__':
    main()
```

Parser Demo




NPC Data
NPCName = String
NPCStrength = Int
NPCLootable = bool
NPCHp = Int
Text Based Adventure
Map--Dungeon
Player Data
Player Inventory
NPC Data
NPC Inventory
Map--Dungeon
PlayerPosition(x,y)
PlayerGrid = [100][100]
NPCPosition(x,y)
MovePosition(curr_x,curr_y,dest_x,dest_y)
Map = Image
RoomList = List[]
RoomPosition= (minx,miny, maxx,maxy)
NPC Inventory
Gold = Int
HPPotion = Int
ArmorName = String
ArmorStrength = Int
Weapon Strength = Int
Weapon Name = String
Player Data
Name - String
Strength - Int
Intelligence - Int
Armor Name - String
Armor Strength - Int
HP = Int
TotalStrength = Int
(Armor is equipped, not in inventory
WeaponName = String
Weapon Strength = Int
Player Inventory
Gold = Int
HPPotion = Int
Armor Name = String
Armor Strength = Int
Weapon Name - String
Weapon Strength = Int

NPC Dictionary

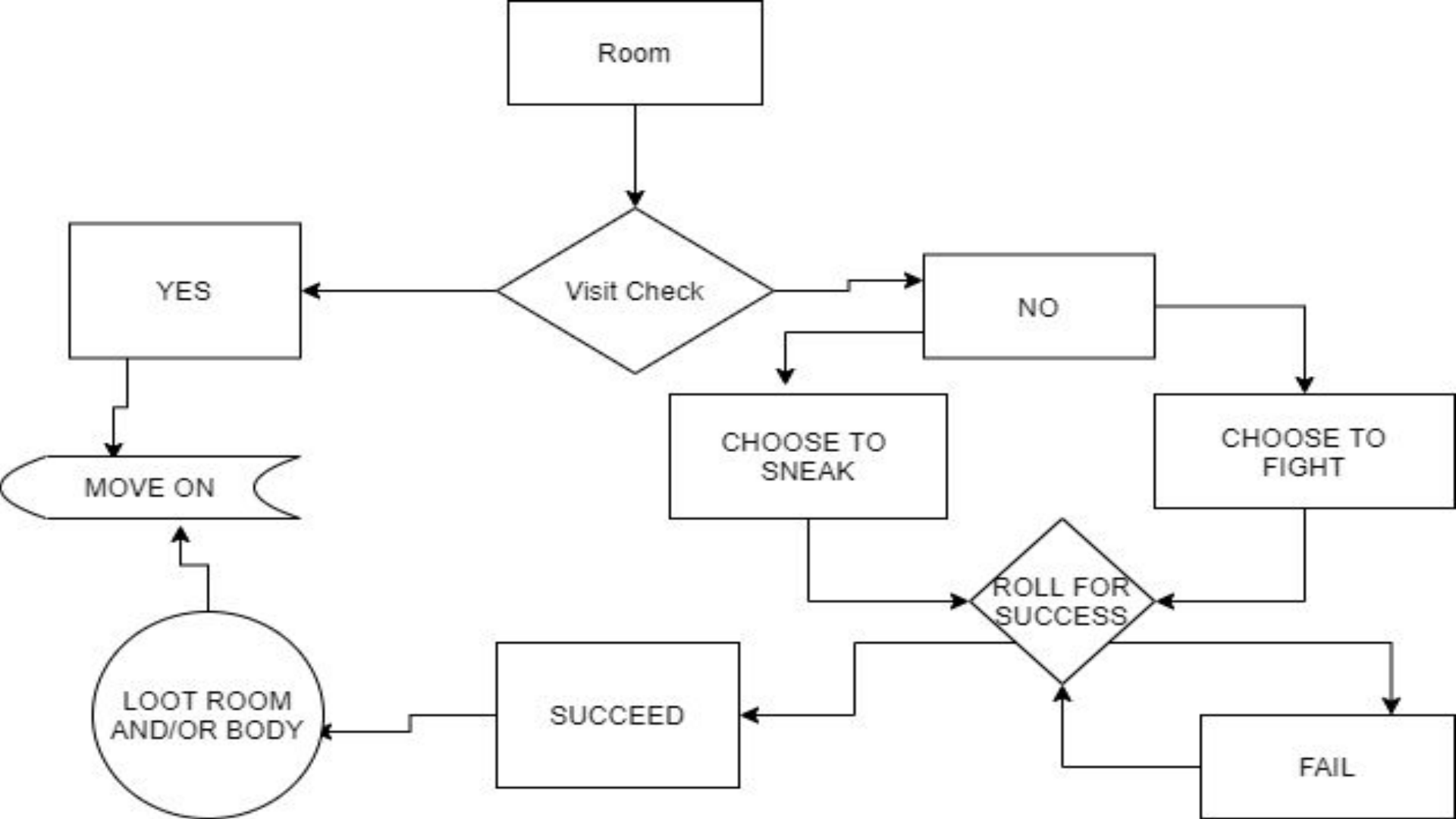
```
"npcName" : "Jareth",
"npcPosition" : 4,
"npcGold" : 10,
"npcHP" : 5,
"npcStrength" : 5,
"npcInventory" : {
    "cloth" : 0,
    "potions" : 1,
    "sword" : {
        "count" : 0,
        "strength" : 0
    },
    "chestplate" : {
        "count" : 0,
        "strength" : 0
    },
```

```
"grieves" : {
    "count" : 0,
    "strength" : 0
},
"helmet" : {
    "count" : 0,
    "strength" : 0
},
"matches" : 0,
"mirror" : 0,
"ring" : {
    "count" : 0,
    "strength" : 0
```

```
"ring" : {
    "count" : 0,
    "strength" : 0
},
"scarf" : {
    "count" : 0,
    "strength" : 0
},
"dagger" : {
    "count" : 0,
    "strength" : 0
}
```

Player Data

```
import random

room = 1
health = 100
strength = random.randrange(11)
intelligence = random.randrange(11)
gold = 2000

class Player:
    def __init__(self, name, strength, intelligence, health, gold, room):
        self.name = name
        self.strength = strength
        self.intelligence = intelligence
        self.health = health
        self.gold = gold
        self.room = room

    @classmethod
    def from_input(cls):
        return cls(
            input("What is your name, adventurer? "),
            strength,
            intelligence,
            health,
            gold,
            room
        )
```

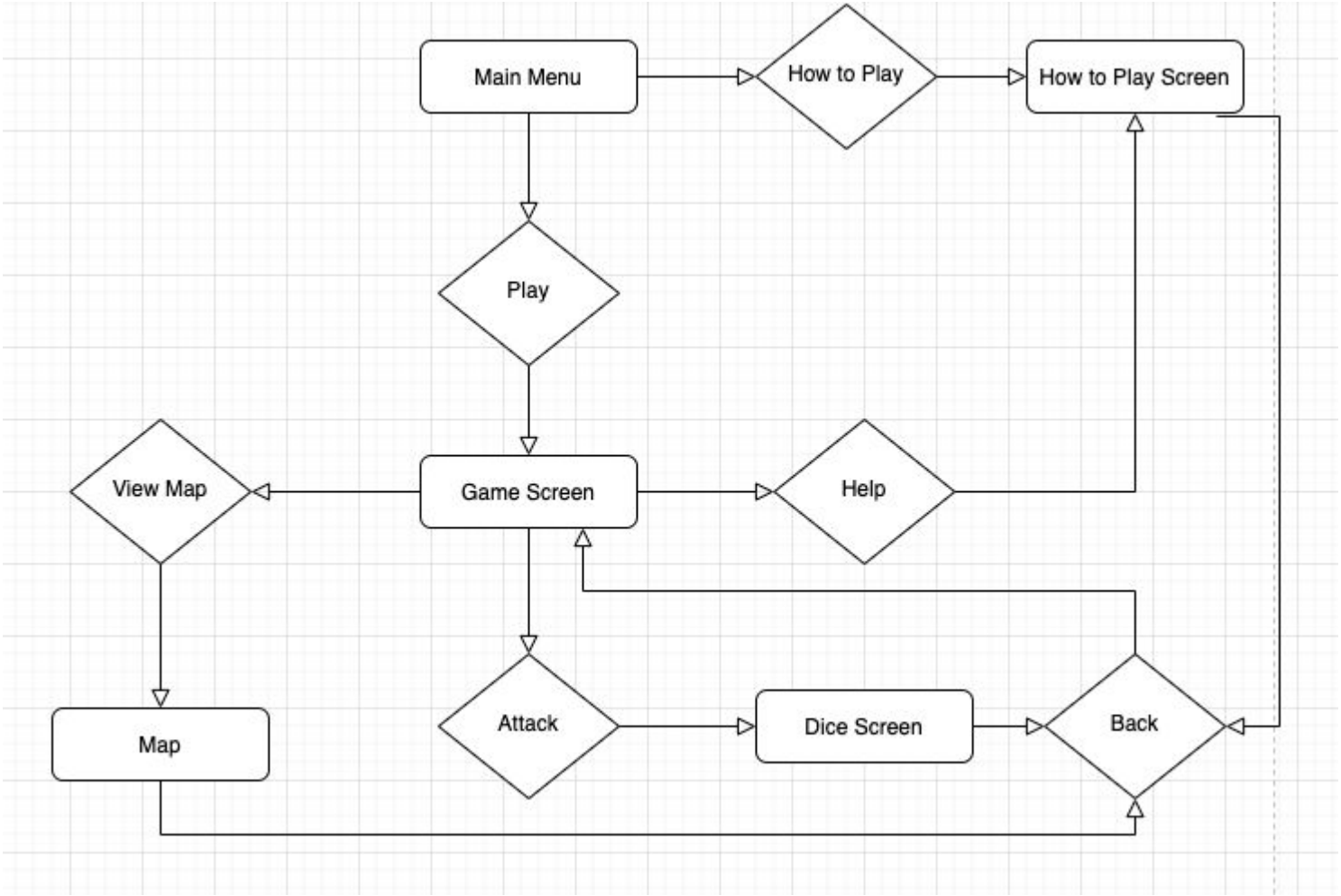

Room
Visit Check
YES
NO
MOVE ON
CHOOSE TO SNEAK
CHOOSE TO FIGHT
ROLL FOR SUCCESS
LOOT ROOM AND/OR BODY
SUCCEED
FAIL

Game Script

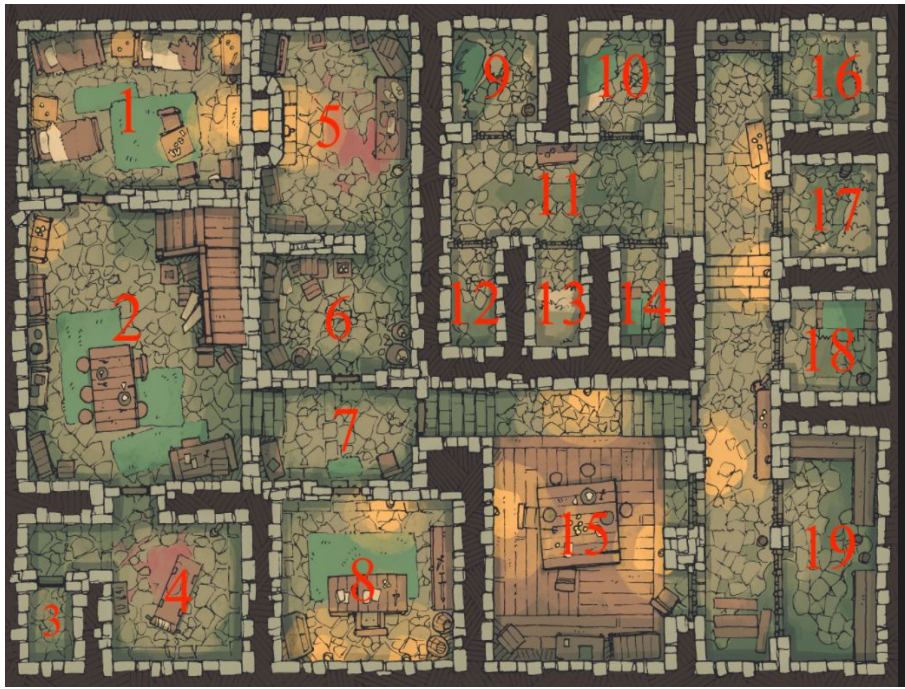
```
def roomCheck(roomNumber):
    with open("savedata.txt", "r") as file:
        lines = file.readlines()
        if lines[roomNumber - 1] == "0":
            return True
        else:
            return False

def roomWrite(roomNumber):
    newLine = "1"
    with open("savedata.txt", "r") as file:
        lines = file.readlines()
        lines[roomNumber - 1] = newLine
    with open("savedata.txt", "w") as file:
        file.writelines(lines)
```

```
def sneakChecker(npcValue):
    rolled = roll()
    if (user.intelligence + rolled) > (npcValue * 2):
        return True
    else:
        return False

def attackChecker(npcValue):
    rolled = roll()
    if (user.strength + rolled) > npcValue:
        return True
    else:
        return False

def attackSequence(npcValue):
    if attackChecker(npcValue):
        print("The attack was successful!")
        return True

    else:
        print("The attack was not successful!")
        user.health = user.health - npcValue
        return False
```

GUI


Main Menu
How to Play
How to Play Screen
Play
View Map
Game Screen
Help
Map
Attack
Dice Screen
Back

Map




```python
def map():

    conn = sqlite3.connect("Map.db")
    c = conn.cursor()

    plt.figure()
    img = mpimg.imread('FinalMap.jpg')
    plt.imshow(img)

    roomNumber = Map.location
    c.execute('SELECT xCoord FROM Maps2 WHERE roomNumber = ?', (roomNumber,))
    rows = c.fetchone()
    for row in rows:
        xCoord = rows

    c.execute('SELECT yCoord FROM Maps2 WHERE roomNumber = ?', (roomNumber,))
    rows = c.fetchone()
    for row in rows:
        yCoord = row
    plt.scatter(xCoord, yCoord, s=250, c='red', marker='o')
    plt.axis('off')
    plt.savefig('latestMap.png', bbox_inches='tight', pad_inches = 0)
```

Dice

```
class DiceScreen(Screen):
    click = 0
    def roll(self):
        if (self.click == 0):
            num = randomNum(False)
            self.ids.diceroll.background_normal = f"Dice/side_{num}.png"
            self.click = 1
        else:
            self.click = 0
            self.ids.diceroll.background_normal = f"Dice/side_1.png"
            self.manager.current = 'play'
```

Connecting All Three Components

```
elif(self.messageCount == 7):
    value = parse1(self.userInput)
    if (value == 'NORTH'):
        self.messageCount = 8
        Map.getLocation(value)
    self.message += game.room4_go_north(value)
```

Game Demo Video