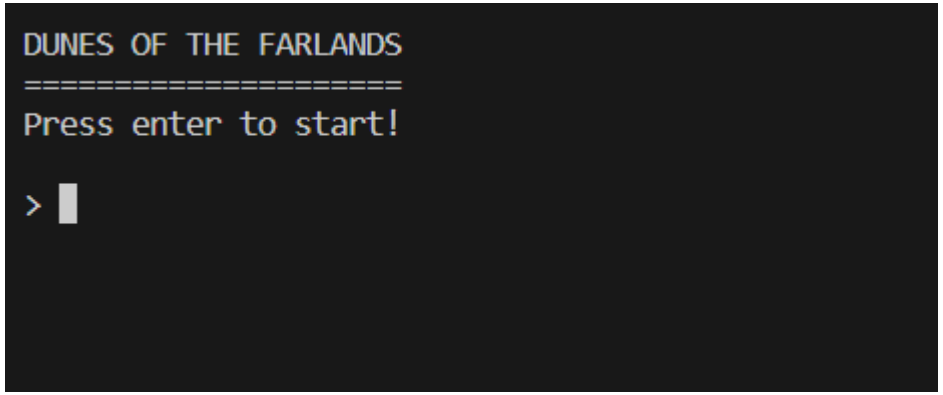
Group 1

Connor Morris, Lane Durst, Shawn Russel, Logan Remondet, Maureen Sanchez, Jack (Yu Joo)

What is our Project?

Our project is a simple Text-Based adventure game called “Dunes of the Farlands”.

The main parts of this program are the user interface, a text input parser, a system for storing and modifying game objects, and a system for outputting gameplay events depending on user choices in the game.



Connor:

Leader / Text Parser Designer

```cpp
int main()
{
    std::string inputText;
    std::string outputText;
    std::tuple<std::string, game_object> parserOutput;
    game_object emptyGameObj;
    player_info player;

    // Main program loop
    while(true)
    {
        // Setting up all the game objects & player character
        initialize_game_objects();
        player = player_info("new");

        // Title card
        std::cout << std::endl << "DUNES OF THE FARLANDS" << std::e
        std::cout << "Press enter to start!" << std::endl << std::e

        // Make the program wait until the user inputs any characte
        get_input();

        // Intro text
        narrator("You awake in a sandy desert. Your head is throbbi
        " What you do know, however, is that your name is Vir Khaba
        " When your vision starts to come back to you, you sit up s
        " You spot a town that appears to be 'abandoned' in the nor
        " but also see what appears to be an oasis nearby.");

        // Main gameplay loop; if player dies, break loop to restar
        while(true)
        {
            // Get the player's current input
            inputText = get_input();

            // Send input to parser
            parserOutput = game_input_parser(inputText);
```

```cpp
std::tuple<std::string, game_object> game_input_parser(std::string input)
{
    std::string returnStr;
    game_object returnObj;
    int iterVal;

    if (input.empty())
    {
        return std::tuple<std::string, game_object>{returnStr, returnObj};
    }

    if (input.size() > 4 && input.substr(0, 4) == "use ")
    {
        returnStr = "use";
        iterVal = 4;
    }
    else if (input.size() > 5 && input.substr(0, 5) == "take ")
    {
```

```cpp
class player_info
{
    private:
        game_object currentLocation;
        std::vector<game_object> inventory;
        std::vector<std::string> flags;
        //std::vector<game_object>::iterator invIter;
        //std::vector<std::string>::iterator flagIter;
        bool isAlive;

    public:
        // Public default constructor
        player_info() {}
        // Public constructor called by main
        player_info(std::string str)
        {
            currentLocation = *mainObjects.begin();
            inventory = {};
            flags = {};
            isAlive = true;
        }
        // Class methods
        void set_location(game_object location)
        {
```

Lane:
Vice-Leader /
UI Designer

I was in charge of creating the basic game loop in the main as well as handling how input and output are read too and from the system.

The main loop simply operates by waiting for any input to begin, and then entering an infinte loop that persists until an exit sequence is entered. This loop simply outputs the current story event, takes user input and parses it, and then outputs the next story event.

The user_input function takes simply prompts the user for input with a '>' then reads the whole next line to a string. This string is then fully capitalized, for easier matching, and returned by the function.

```
// Method for prompting and reading user input
std::string get_input()
{
    std::string temp;
    // Prompts for user input without skipping a line
    std::cout << "> ";
    // Reads user input into temp until a newline character is reached
    getline(std::cin, temp);
    // Simply converts input into uppercase for easy matching
    std::transform(temp.begin(), temp.end(), temp.begin(), ::toupper);
    // Returns the player input
    return temp;
}
```

There are additionally two 'dialogue' functions, which simply take a string and output it to the screen. The first of these is the narrator function which puts the text within brackets []. The other is called npc text which outputs to the screen in the format npcName+format+dialogue, with 'format' being an optional field default set to ': '

```
// Function for returning npc dialogue to the screen
void npc_text(std::string dialogue, std::string npcName, std::string format = ": ")
{
    int wrap = 100;
    // Default formatting takes the style [Name: dialogue], however the format field was included
    // as an optional argument in case specific NPC entities should have different dialogue style
    if (dialogue.length() > wrap)
    {
        // unimplemented
    }
    std::string output = npcName + format + dialogue;
    std::cout << output << std::endl;
}

// Method for returning all non-dialogue information to the screen
void narrator(std::string text)
{
    std::cout << "[" + text + "]" << std::endl;
}
```

Shawn: Story Lead / Gameplay Implementation

Prologue

Description: You awake in a sandy desert. Your head is throbbing, and you don't remember much. What you do know, however, is that your name is Vir Khabar, a human. When your vision starts to come back to you, you sit up slowly to check if anything is around you. You spot a town that appears to be "abandoned" in the north.

Chapter 1: Awakening in the Desert Keep

- You can either venture out to the desert oasis, or you can see what the "abandoned" town may bring you. Do you…
  - Travel along the oasis
    - If they choose this, a poisonous desert frog lands on their foot and kills them within 10 seconds.
  - Or make your way to town?

```
std::string talk_to(game_object &obj, player_info &playerChar)
{
    if (obj.get_object_name() == "barkeep" )
    {
        if (playerChar.get_location() == tavern)
        {
        return "Welcome to the Sand Dune Saloon. I'm the barkeep. Here's a drink, it's on the house; I can tell you're good people."
        " I must warn you though: there is nothing worth staying for at this town, that is, unless you want to get yourself killed by"
        " one of the locals. If I were you, I'd recommend heading east of here, to the Farlands. It's quite the dangerous area in its"
        " own right, but you'll find more to do over in that region.";
        }
    }
    else if (obj.get_object_name() == "old lady")
    {
        if (playerChar.get_location() == abandonedTown)
        {
            playerChar.set_player_state(false);
```

Logan: Story Lead / Gameplay Implementation

- You have a scimitar and a small shield. You have the option to either attack head on or apologize and try to sort things out. Do you…
  - Attack the thugs head on.
    - If the player chooses this, you will unsheathe your sword and kill the first guy in front of you out of reflex. The bandits back off slowly and leave the saloon. None of the friendlier locals even batted an eye.
  - Or apologize profusely (“I’m so sorry, please forgive me”)
    - If the player chooses this, you will attempt to apologi
      escape, he gut punches you and then stabs you in the

```
// Read things like books, notes, etc.
std::string read(game_object &obj, player_info &playerChar)
{
    if (obj.get_object_type() == "item")
    {
        if (obj.get_object_flag("at_location") == "at_location")
        {
            return obj.get_object_description();
        }
    }

    return "There's nothing to read over here.";
}

// This function will take in an action as a string and the current
// game object being called upon; it will then check the name of the
// action via a set of if statements doing string comparisons, and once
// it finds a match, it will send the game object through to a
// dedicated function for the action being called upon.
std::string main_action(std::string act, game_object &obj, player_info &playerChar)
{
    std::string result = "";
```

Maureen: Story Implementer / General Code Helper

```
// This function initializes all game objects at the start of the program's runtime.
void initialize_game_objects() {

    // Check if game has already started; if mainObjects is not empty then
    // clear mainObjects & re-add objects in their default state to restart game.
    if (!mainObjects.empty()) {
        mainObjects.clear();
    }

    // Initializing items (objects of type "item")
    sword = game_object("item", "sword", "You look upon an ordinary sword; "
    "it's not pretty, but it gets the job done.");
    shield = game_object("item", "shield", "You look upon an ordinary shield; "
    "it may be made out of wood, but it'll protect you well enough. Maybe.");
    chestkey = game_object("item", "chestkey", "This is, almost certainly, "
    "the key to the chest. The engraving on the side says 'chest key';"
    " I'd be surprised if it was for anything else.");
    chest = game_object("item", "chest", "You look at the chest and see "
    "that it is... a chest. What, did you expect a mimic or something?");
    note = game_object("item", "note", "The note reads: 'January 18th. "
    "Seen some bandits around here recently. Trying to stay out of sight. I know it's part of"
    " my job to keep this chest protected, but I won't make it out here much longer.'");
    drink = game_object("item", "drink", "It's the drink the barkeep gave you at the tavern."
    " You get the feeling it'd be nice to take a drink within the tavern.", {"at_location"});

    mainObjects.insert(mainObjects.end(), sword);
    mainObjects.insert(mainObjects.end(), shield);
    mainObjects.insert(mainObjects.end(), chestkey);
```

Jack: Story Implementer / General Code Helper

```
// This class contains information about any object within the game
class game_object
{
    private:
        std::string objectType;
        std::string objectName;
        std::string objectDescription;
        std::vector<std::string> objectFlags;
        //std::vector<std::string>::iterator flagIter;

    public:
        // Public default constructor
        game_object() {}
        // Public constructor which sets a game_object's type, name & description only.
        game_object(std::string oType, std::string oName, std::string oDesc)
```

```
// The master list of all objects in the game; add objects to this vector after creation.
std::vector<game_object> mainObjects = {};

game_object gameStart;
game_object abandonedTown;
game_object oasis;
```