Project 2

Dungeon Crawl Game

Andrew Reid East

CIS-5 - 40718

February 12, 2015

Table of Contents

Introduction

The Dungeon Crawl Game is a simple role playing game loosely based around the concept of the classic computer game Rogue. In this long-standing game, a maze-like series of rooms is presented as ASCII characters in a terminal, and the player is represented by a single character that they challenge a multi-level dungeon filled with monsters, also shown by a single character each. This game was known for being complex and difficult to beat, and has challenged many computer techs and programmers over the years. I wanted to try to emulate some of the controls and displays of such a game, and felt I could create a playable version to fulfil the requirements of this project.

Game Play and Rules

In my implementation, I chose to have two modes. You start in a dungeon crawl mode in which you move your player, the "Q" character, about the screen with the letter keys WASD. You can view a brief help screen or drink a potion to heal, but the main goal of this mode is to explore and find monsters to fight! You interact with another game asset by running into it.

The second mode starts when you move upon a monster. The fight screen is based on the dice rolling of a tabletop role playing game. You can choose to attack, and then roll a virtual dice to create random chance, and add your character's skill modifier to get a chance to hit past the monster's defense. You can choose to defend, and get a bonus to deflect the monster's attack. After you attack or defend, the monster get a chance to do the same. Turns continue until one combatant reaches zero hit points.

The game as it stands is a single dungeon level with a few monsters, and a harder boss monster. Since all game data is loaded from files, it could be modified and expanded.

Development Summary

Lines of Code Metrics

Total Lines of Code: 1337
Physical Executable Lines of Code: 1011
Logical Executable Lines of Code: 654
Number of Methods: 27

Programming Methods

This game presented two challenges I have not before overcome in previous programming: generalized saving to a file of a game world, and the complete stateful operation of a dynamic game. I also wanted to make my game run smoothly in the console more like the original Rogue game, which required using a Windows-specific, non-portable method to gather keyboard input without waiting for the Enter key being pressed. To ensure the program could be tested on any platform, I kept all such input routines under a flag that will only be turned on if Windows is tested for.

Creating a generalized save game state that interface from programming structures and could be saved and loaded to text files was actually the more challenging portion of the project for me. I had a general idea of the data structure required—a two-dimensional array holding the characters that represent the game board, and structured data for each character "asset". As I worked with the code that created, loaded, and then interacted with the objects, my definitions of what was required for the objects changed quite a bit, and even more so as I approached how to retrieve that data modularly from disk. I ended up with two main types of files saved: one master "map" file that has the map grid and a reference list of all character assets to load, and a collection of single files identified by a numeric ID for each asset in game. Saving and loading these files was mainly challenging in keeping the data consistent for read versus write—especially those tricky newline characters. In the end, I think the experience I gained by working through this problem will be valuable for working with flat files in the future, perhaps with configuration files or other common tasks for a desktop programmer.

The more fun challenge was to control the flow of the game in response to the user's input, and centrally manage the game state using a primitive structure and game state flag. I used a single loop to control the flow of the program, which tested the game state flag each time to see if the user had requested a change in the game's operation. By centralizing the logic, I not only made my program smaller, but much easier to understand. Before, if the user requested something like to save the game or to exit, the procedure was handled as it was inputted. But, when I simply changed the game state and let the main loop dispatch the save or exit routine, the program is much more readable and was easier to modify.

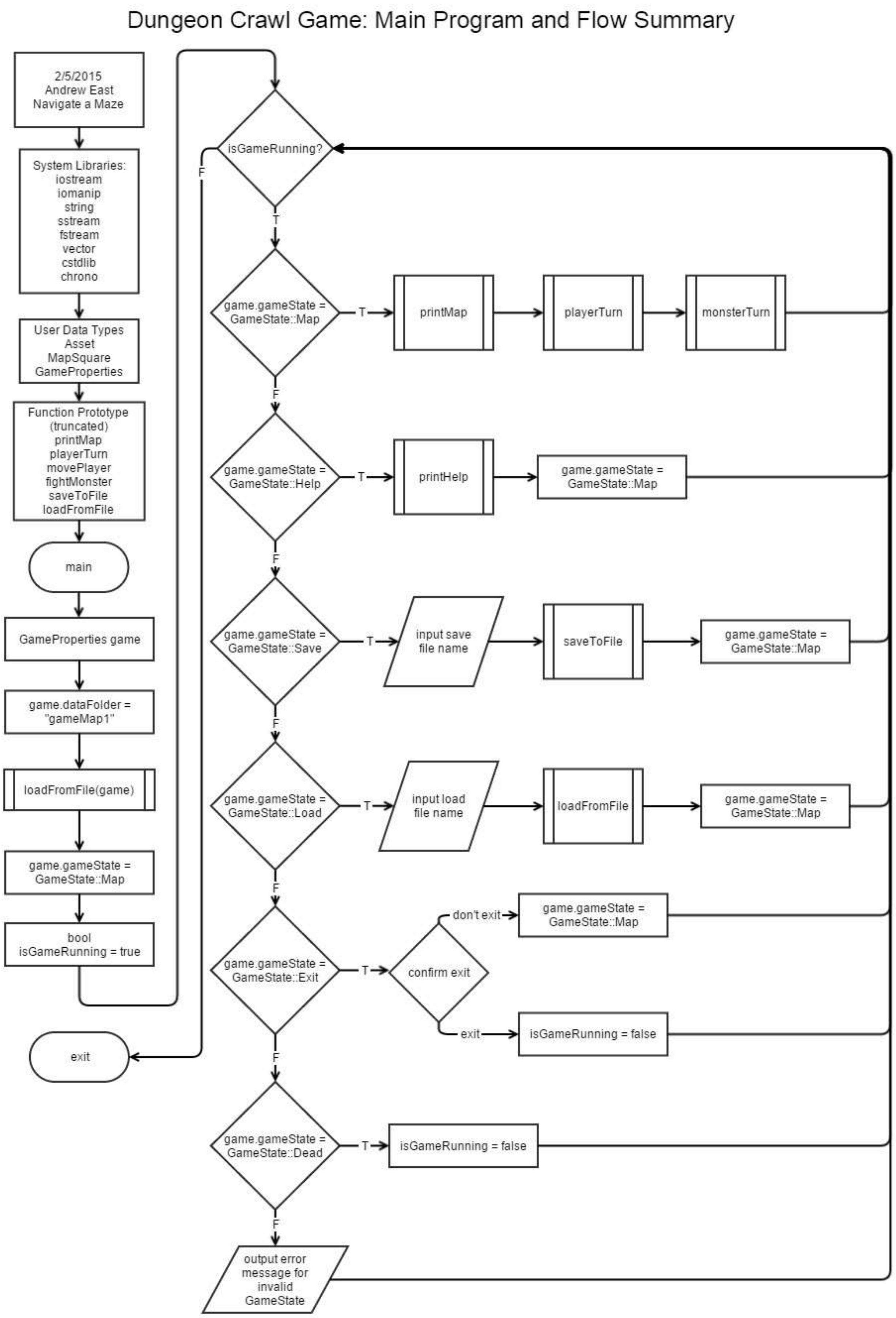
I estimate I spent 24 hours building and testing this project. Most of that time was spent in the iterative process of planning a new feature, implementing the code, finding the right method or programming technique to make it work, and then testing that change.

Specifications

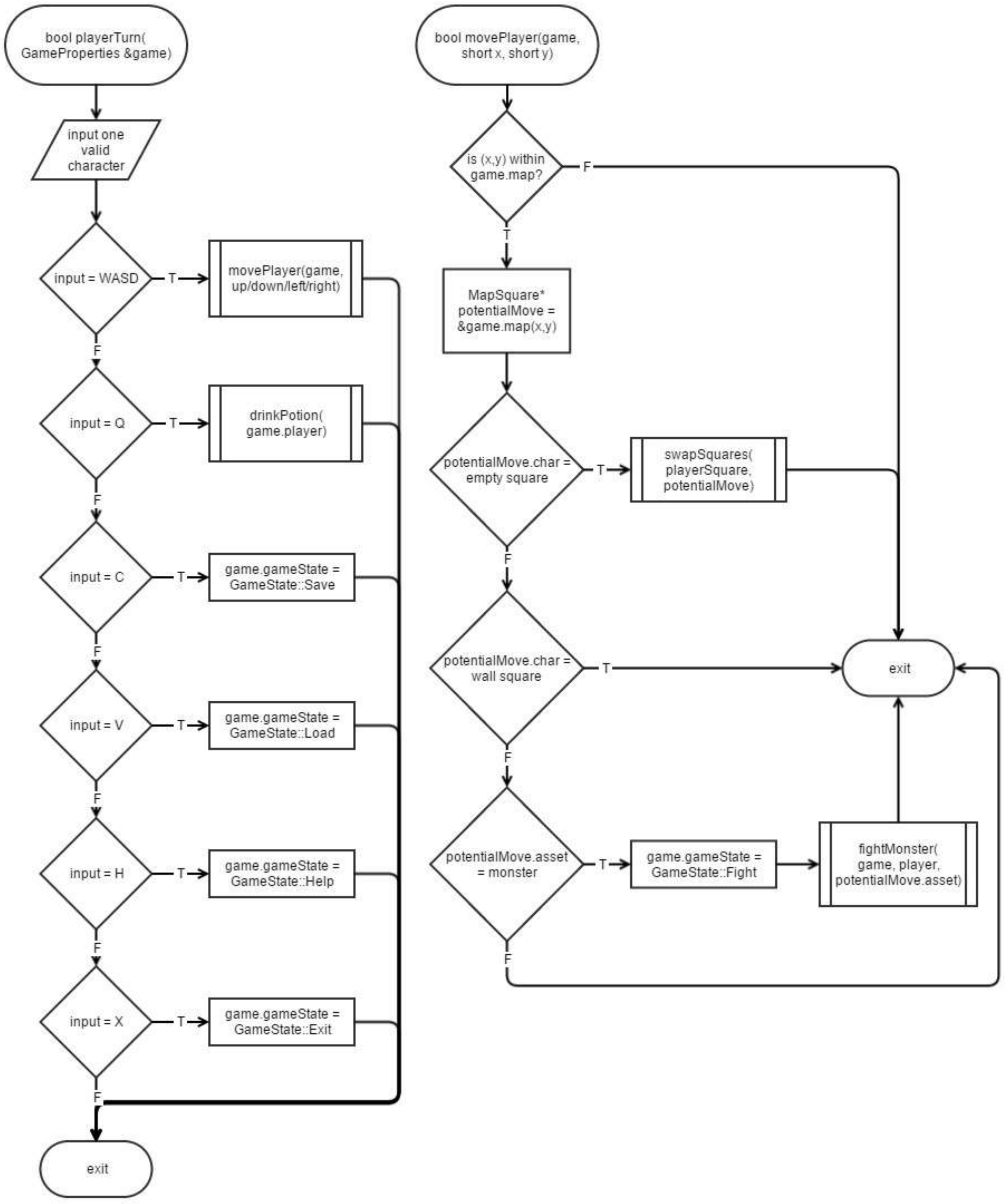
Major Methods

- `void printMap(MapSquare *map, short sizeX, short sizeY)`
    - Displays the grid of the game map from the data structure
- `void printFight(GameProperties &game, Asset* monster, Asset* player)`
    - Displays the vital stats of the combatants
- `void printControlScheme(GameState currState)`
    - Shows the controls for quick reference while playing
- `bool cls(bool WIN32_MODE = false)`
    - Clears the screen after each frame
- `bool playerTurn(GameProperties &game, bool WIN32_MODE = false)`
    - Inputs the player’s action, and calls the appropriate subroutine
- `bool movePlayer(GameProperties &game, short x, short y, bool WIN32_MODE)`
    - Moves the player’s sprite on the map
- `bool fightMonster(GameProperties &game, Asset* monster, Asset* player)`
    - Executes a complete fight against specified monster
- `short rollDie(short d)`
    - Random number generator
- `bool drinkPotion(Asset *player)`
    - Increases hit points, decreases potion count
- `short findAssetIndex(vector<Asset *> &assets, short assetID)`
    - Searches data store of game assets by numeric ID
- `bool getAKey(char& input, bool WIN32_MODE = false)`
    - Gets one single alphabet key of input using appropriate console input method specified by WIN32_MODE
- `bool saveToFile(GameProperties &game)`
    - Writes to disk
- `bool loadFromFile(GameProperties &game)`
    - Reads from disk (also loads new game at startup)
- `void sortAssetsByIndex(vector<Asset *> &assets)`
    - Sorts the data store
- `bool isRunningInAWin32Console();`
    - Tests if Windows-specific features may be used

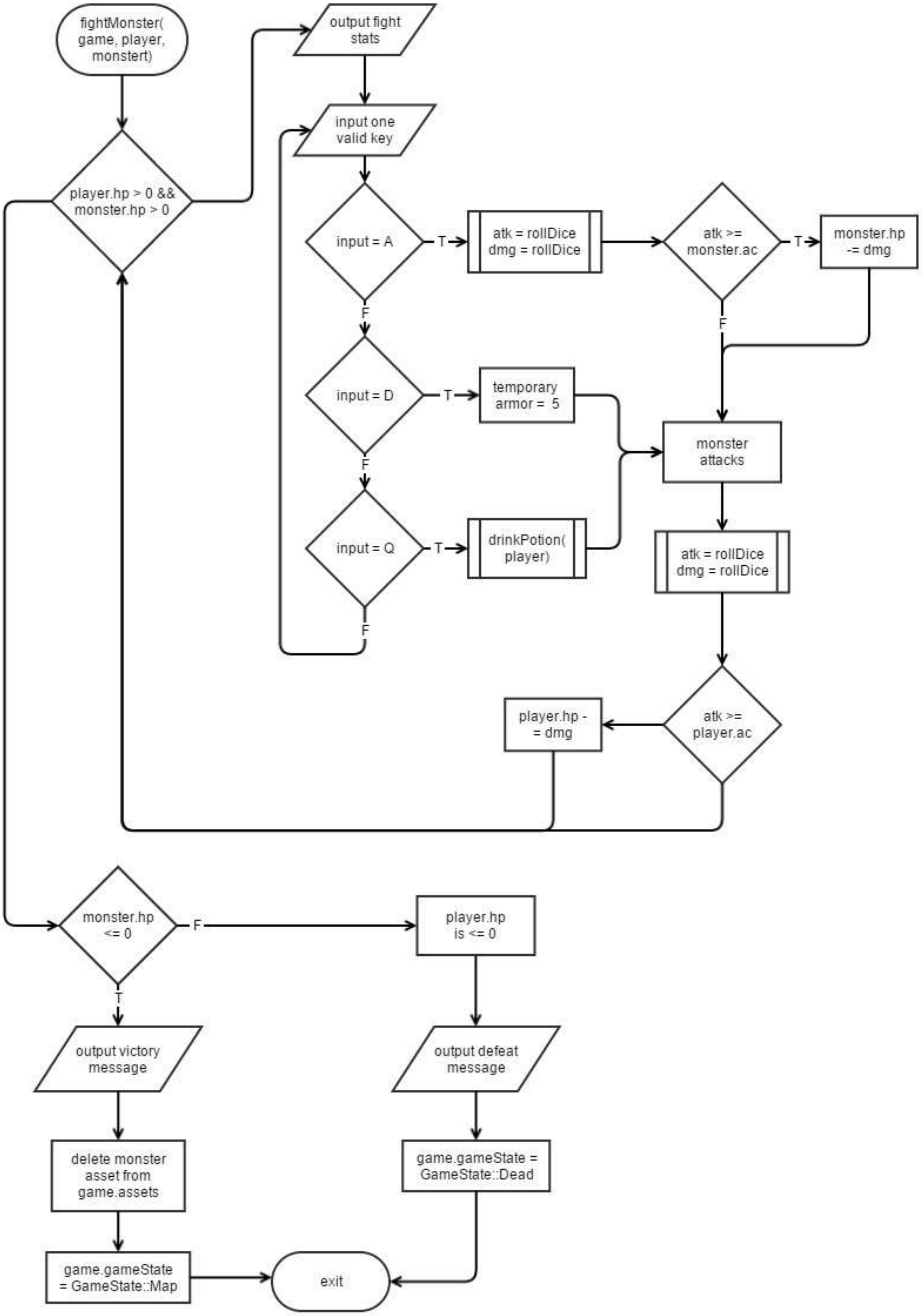
Flowcharts (Summary of Program Flow and Logic—Not Exhaustive Code)


Dungeon Crawl Game: Main Program and Flow Summary
2/5/2015
Andrew East
Navigate a Maze
System Libraries: iostream iomanip string sstream fstream vector cstdlib chrono
User Data Types Asset MapSquare GameProperties
Function Prototype (truncated) printMap playerTurn movePlayer fightMonster saveToFile loadFromFile
main
GameProperties game
game.dataFolder = "gameMap1"
loadFromFile(game)
game.gameState = GameState::Map
bool isGameRunning = true
exit
isGameRunning?
T
F
game.gameState = GameState::Map
printMap
playerTurn
monsterTurn
game.gameState = GameState::Help
printHelp
game.gameState = GameState::Map
game.gameState = GameState::Save
input save file name
saveToFile
game.gameState = GameState::Map
game.gameState = GameState::Load
input load file name
loadFromFile
game.gameState = GameState::Map
game.gameState = GameState::Exit
confirm exit
don't exit
game.gameState = GameState::Map
exit
isGameRunning = false
game.gameState = GameState::Dead
isGameRunning = false
output error message for invalid GameState

Dungeon Crawl Game: playerTurn


bool playerTurn(GameProperties &game)
input one valid character
input = WASD
T
movePlayer(game, up/down/left/right)
F
input = Q
T
drinkPotion(game.player)
F
input = C
T
game.gameState = GameState::Save
F
input = V
T
game.gameState = GameState::Load
F
input = H
T
game.gameState = GameState::Help
F
input = X
T
game.gameState = GameState::Exit
F
exit
bool movePlayer(game, short x, short y)
is (x,y) within game.map?
F
T
MapSquare* potentialMove = &game.map(x,y)
potentialMove.char = empty square
T
swapSquares(playerSquare, potentialMove)
F
potentialMove.char = wall square
T
exit
F
potentialMove.asset = monster
T
game.gameState = GameState::Fight
fightMonster(game, player, potentialMove.asset)
F

Dungeon Crawl Game: fightMonster



C++ Programming Concepts

Multi-Dimensional Array (Implemented as Dynamic Array)	`MapSquare *newMap = new MapSquare[mapSizeX * mapSizeY];` `newMap[y * mapSizeX + x]`
Pass Array Between Function	`void printMap(MapSquare *map, short sizeX, short sizeY)`
Pass by Reference	`void printStatus(GameProperties &game)`
Defaulted Parameters	`bool cls(bool WIN32_MODE = false);`
Returning Primitive Data Types	`short findAssetIndex(vector<Asset *> &assets, short assetID);`
Formatted Output	`cout << setw(15) << "HP" << setw(5) << player->hp << setw(15) << "HP" << setw(5) << monster->hp << endl;`
Read from Files	`ifstream assetFile;` `assetFile.open(filename);` `while (getline(assetFile, line).good())`
Write to Files	`for (short x = 0; x < game.mapSizeX; ++x)` `mapFile << game.map[y * game.mapSizeX + x].display;`
Sorting	`void sortAssetsByIndex(vector<Asset *> &assets)`
Searching	`short findAssetIndex(vector<Asset *> &assets, short assetID)`
Variables	`char choice = 0;`
Console Input and Output	`cout << "Would you like to quit the game?" << endl;` `cin >> choice;`
if	`if (choice == 'Y' || choice == 'y')` `return 0;`
if, else if, else	`if (potentialMove->display == ' ')` `{` `game.player->x = x;` `game.player->y = y;` `return true;` `}` `else if (potentialMove->display == '#')` `{` `return true;` `}` `else` `{` `if (potentialMove->linkedActor->isActor)` `{` `game.gameState = GameState::Fight;` `return fightMonster(game, potentialMove->linkedActor, game.player);` `}` `}`
switch	`switch (input)` `{` `case 'W':` `return movePlayer(game, game.player->x, game.player->y - 1, WIN32_MODE` `case 'A':`

	return movePlayer(game, game.player->x - 1,
	game.player->y, WIN32_MODE);
	  case 'S':
	    return movePlayer(game, game.player->x, game.player->y
	+ 1, WIN32_MODE);
	  case 'D':
	    return movePlayer(game, game.player->x + 1,
	game.player->y, WIN32_MODE);
	  case 'Q':
	    return drinkPotion(game.player);
	  case 'X':
	    game.gameState = GameState::Exit;
	    return false;
	  case 'H':
	    game.gameState = GameState::Help;
	    return false;
	  case 'C':
	    game.gameState = GameState::Save;
	    return false;
	  case 'V':
	    game.gameState = GameState::Load;
	    return false;
	  default:
	    return false;
	}

while	while (monster->hp > 0 && player->hp > 0)

do...while	char temp;
	do
	{
	  strm.get(temp);
	} while (temp != '\n' && temp != '\0');

for	for (y = 0; y < sizeY; ++y)
	{
	  for (x = 0; x < sizeX; ++x)
	    cout << map[y * sizeX + x].display;
	  cout << endl;
	}

Boolean logic statements	while (y < mapSizeY && getline(mapFile, line).good())
Increment/decrement	line.at(x++)
Comments	// newAsset = new Asset; // Note: The parens() are IMPORTANT! It initializes all members of the struct to default values (zero) when called as "new Asset()"!!
functions	void printMap(MapSquare *map, short sizeX, short sizeY);

Sample Inputs and Output

```
................................................................................
................................................................................
.........................................#############..........................
....######################################           #..........................
....#                                      S      Q  #..........................
....#                          ###########           #..........................
....#                          #.........#############..........................
....#                          #................................................
....#     I                    #..................###################...........
....#                          #..................#                 #...........
....#                 I        #..................#                 #...........
....#                          ####################                 #...........
....#                                                       O       #...........
....#                          ####################                 #...........
....#                  I       #..................#                 #...........
....#                          #..................#                 #...........
....############################..................########   ########...........
.........................................................#   #..................
.........................................................#   #..................
.......################################..................#   #..................
.......#                              #..................#   #..................
.......#                              #..................#   #..................
.......#                              #..................#   #..................
.......#                              #..................#   #..................
.......#     D                        ####################   #..................
.......#                                                     #..................
.......#                              ########################..................
.......#                              #.........................................
.......################################.........................................
................................................................................

You wake up in a dank room. You hear water slowly dripping,
and you feel slimy mold on the floor.

 [ HP: 100 ]   [ Potions: 4 ]   [ EXP: 0 ]

Move:     (W)      (Q) Quaff Potion           (C) Save (H) Help
       (A)(S)(D)                              (V) Load (X) Exit
> W

................................................................................
................................................................................
.........................................#############..........................
....######################################        Q  #..........................
....#                                      S         #..........................
....#                          ###########           #..........................
....#                          #.........#############..........................
....#                          #................................................
....#     I                    #..................###################...........
....#                          #..................#                 #...........
....#                 I        #..................#                 #...........
....#                          ####################                 #...........
....#                                                       O       #...........
....#                          ####################                 #...........
....#                  I       #..................#                 #...........
....#                          #..................#                 #...........
....############################..................########   ########...........
.........................................................#   #..................
.........................................................#   #..................
.......################################..................#   #..................
.......#                              #..................#   #..................
.......#                              #..................#   #..................
.......#                              #..................#   #..................
.......#                              #..................#   #..................
.......#     D                        ####################   #..................
.......#                                                     #..................
.......#                              ########################..................
.......#                              #.........................................
.......################################.........................................
................................................................................

You wake up in a dank room. You hear water slowly dripping,
and you feel slimy mold on the floor.

 [ HP: 100 ]   [ Potions: 4 ]   [ EXP: 0 ]

Move:     (W)      (Q) Quaff Potion           (C) Save (H) Help
       (A)(S)(D)                              (V) Load (X) Exit
> A

................................................................................
................................................................................
.........................................#############..........................
....######################################       Q   #..........................
```

```
....#                                      S         #..........................
....#                          ###########           #..........................
....#                          #.........#############..........................
....#                          #................................................
....#     I                    #..................###################...........
....#                          #..................#                 #...........
....#                 I        #..................#                 #...........
....#                          ####################                 #...........
....#                                                       O       #...........
....#                          ####################                 #...........
....#                  I       #..................#                 #...........
....#                          #..................#                 #...........
....############################..................########   ########...........
.........................................................#   #..................
.........................................................#   #..................
.......################################..................#   #..................
.......#                              #..................#   #..................
.......#                              #..................#   #..................
.......#                              #..................#   #..................
.......#                              #..................#   #..................
.......#     D                        ####################   #..................
.......#                                                     #..................
.......#                              ########################..................
.......#                              #.........................................
.......################################.........................................
................................................................................

You wake up in a dank room. You hear water slowly dripping,
and you feel slimy mold on the floor.

 [ HP: 100 ]   [ Potions: 4 ]   [ EXP: 0 ]

Move:     (W)      (Q) Quaff Potion           (C) Save (H) Help
       (A)(S)(D)                              (V) Load (X) Exit
>

Output Truncated

................................................................................
................................................................................
.........................................#############..........................
....######################################           #..........................
....#                                      SQ        #..........................
....#                          ###########           #..........................
....#                          #.........#############..........................
....#                          #................................................
....#     I                    #..................###################...........
....#                          #..................#                 #...........
....#                 I        #..................#                 #...........
....#                          ####################                 #...........
....#                                                       O       #...........
....#                          ####################                 #...........
....#                  I       #..................#                 #...........
....#                          #..................#                 #...........
....############################..................########   ########...........
.........................................................#   #..................
.........................................................#   #..................
.......################################..................#   #..................
.......#                              #..................#   #..................
.......#                              #..................#   #..................
.......#                              #..................#   #..................
.......#                              #..................#   #..................
.......#     D                        ####################   #..................
.......#                                                     #..................
.......#                              ########################..................
.......#                              #.........................................
.......################################.........................................
................................................................................

You wake up in a dank room. You hear water slowly dripping,
and you feel slimy mold on the floor.

 [ HP: 100 ]   [ Potions: 4 ]   [ EXP: 0 ]

Move:     (W)      (Q) Quaff Potion           (C) Save (H) Help
       (A)(S)(D)                              (V) Load (X) Exit
> A

You are attacking a Slime!


              Player               Slime
      --------------      --------------
             HP  100             HP   10
         Attack   10         Attack    2
      Damage 1d8 + 2      Damage 1d6 + 2
             AC   15             AC   12
        Potions    4
```

```
     (A) Attack   (D) Defend   (Q) Quaff Potion

> A

You attack the Slime! You roll a 8 on a 20-sided die.
You hit the Slime with a total attack of 18!
You deal 6 damage!

The Slime attacks you! It rolls a 18 on a 20-sided die.
The monster hits with a total attack of 20.
It does 7 damage to you!


              Player               Slime
      --------------      --------------
             HP   93             HP    4
         Attack   10         Attack    2
      Damage 1d8 + 2      Damage 1d6 + 2
             AC   15             AC   12
        Potions    4

     (A) Attack   (D) Defend   (Q) Quaff Potion

> A

You attack the Slime! You roll a 1 on a 20-sided die.
You miss the Slime with your attack total of 11.


The Slime attacks you! It rolls a 19 on a 20-sided die.
The monster hits with a total attack of 21.
It does 5 damage to you!


              Player               Slime
      --------------      --------------
             HP   88             HP    4
         Attack   10         Attack    2
      Damage 1d8 + 2      Damage 1d6 + 2
             AC   15             AC   12
        Potions    4

     (A) Attack   (D) Defend   (Q) Quaff Potion

> A

You attack the Slime! You roll a 11 on a 20-sided die.
You hit the Slime with a total attack of 21!
You deal 8 damage!





              Player               Slime
      --------------      --------------
             HP   88             HP   -4
         Attack   10         Attack    2
      Damage 1d8 + 2      Damage 1d6 + 2
             AC   15             AC   12
        Potions    4

You are victorious! You have killed the Slime! You gain 20 experience points.

 (R) Return to previous screen

> R
................................................................................
................................................................................
.........................................#############..........................
....######################################           #..........................
....#                                       Q        #..........................
....#                         ###########           #..........................
....#                         #.........#############..........................
....#                         #................................................
....#     I                    #..................###################...........
....#                 I        #..................#                 #...........
....#                         ####################                 #...........
....#                                                       O       #...........
....#                         ####################                 #...........
....#                  I       #..................#                 #...........
....#                         #..................#                 #...........
....############################..................########   ########...........
.........................................................#   #..................
.........................................................#   #..................
.......################################..................#   #..................
.......#                              #..................#   #..................
```

```
.......#                              #..................#   #..................
.......#                              #..................#   #..................
.......#                              #..................#   #..................
.......#     D                        ####################   #..................
.......#                                                     #..................
.......#                              ########################..................
.......#                              #.........................................
.......################################.........................................
................................................................................

You wake up in a dank room. You hear water slowly dripping,
and you feel slimy mold on the floor.

 [ HP: 88 ]   [ Potions: 4 ]   [ EXP: 20 ]

Move:     (W)      (Q) Quaff Potion           (C) Save (H) Help
       (A)(S)(D)                              (V) Load (X) Exit
> C
What is the folder name of your save game?
(Due to limits of the program, you must create the folder BEFORE saving.)
> save1
Saved to save1.

 (R) Return to previous screen

> R
................................................................................
................................................................................
.........................................#############..........................
....######################################           #..........................
....#                                       Q        #..........................
....#                          ###########           #..........................
....#                          #.........#############..........................
....#                          #................................................
....#     I                    #..................###################...........
....#                          #..................#                 #...........
....#                 I        #..................#                 #...........
....#                          ####################                 #...........
....#                                                       O       #...........
....#                          ####################                 #...........
....#                  I       #..................#                 #...........
....#                          #..................#                 #...........
....############################..................########   ########...........
.........................................................#   #..................
.........................................................#   #..................
.......################################..................#   #..................
.......#                              #..................#   #..................
.......#                              #..................#   #..................
.......#                              #..................#   #..................
.......#                              #..................#   #..................
.......#     D                        ####################   #..................
.......#                                                     #..................
.......#                              ########################..................
.......#                              #.........................................
.......################################.........................................
................................................................................

You wake up in a dank room. You hear water slowly dripping,
and you feel slimy mold on the floor.

 [ HP: 88 ]   [ Potions: 4 ]   [ EXP: 20 ]

Move:     (W)      (Q) Quaff Potion           (C) Save (H) Help
       (A)(S)(D)                              (V) Load (X) Exit
> X

Would you like to quit the game?
  (C) Save and quit
  (Y) Yes, quit now
  (N) Cancel
> C
Saving game now.
Save to save1 successful.
Thank you for playing!
```

Appendix: Code Listing

```
/*
    File:   main.cpp
    Author: Andrew Reid East
    Class: CSC-5 40718
    Created on February 5, 2015, 5:54 PM
    Purpose: Play an ASCII-text based RPG/Adventure/Maze game inspired by Rogue
 */

//System Libraries
#include <iostream>
#include <iomanip>
#include <string>
#include <sstream>
#include <fstream>
#include <vector>
#include <cstdlib> //rand
#include <chrono> //seed

// MSDN Method to Read Unbuffered Input from Keyboard
//  reference:
//    reading input buffer events
//      https://msdn.microsoft.com/en-
us/library/windows/desktop/ms685035(v=vs.85).aspx
//    cls using WinAPI
//      https://msdn.microsoft.com/en-
us/library/windows/desktop/ms682022(v=vs.85).aspx
#include <windows.h> //for Windows-only implementation of keyboard input and CLS
// #include <stdio.h> //for printf and stderr only, i think. remove those, then
disable.

using namespace std;

//Global Constants

//struct Prototypes
struct Asset
{
  short assetID;
  char display;
  short x;
  short y;

  string name;
  bool isActor;
  short hp;
  short ac;
  short hitBonus;
  short damage;
  short damageBonus;
  short exp;

  bool isPlayer;
  // short mp;
  short qtyPotion;
  short potionHeals;
  short expTotal;
  // short equippedWeaponID;
  // vector<Asset> inventory;
  // bool isItem;
  // char *art; //string (including \n) of ASCII art of item
};

struct MapSquare
{
  char display;
  Asset* linkedActor;
};

enum class GameState {Map, Fight, Dead, Save, Load, Help, Dialog, Exit};

struct GameProperties
{
  GameState gameState;

  MapSquare *map;

  vector<Asset *> gameAssets;
  Asset *player;

  short currStatus;
  vector<string> statusDictionary;

  string dataFolder; //default to "gameMap1"
  bool userRenamedSave;

  short mapSizeX;
  short mapSizeY;
  short screenSizeX;
  short screenSizeY;
};

//Function Prototypes

//screen display:
void printStatus(GameProperties &game);
void printHelp();
void printMap(MapSquare *map, short sizeX, short sizeY);
void printFight(GameProperties &game, Asset* monster, Asset* player);
void printControlScheme(GameState currState);
bool cls(bool WIN32_MODE = false);
bool cls_win32();
void clearStreamNewlines(istream &strm);

//execute actions
bool playerTurn(GameProperties &game, bool WIN32_MODE = false);
bool monstersTurn(GameProperties &game);
bool movePlayer(GameProperties &game, short x, short y, bool WIN32_MODE = false);
bool overwriteSquare(MapSquare *from, MapSquare *to);
bool fightMonster(GameProperties &game, Asset* monster, Asset* player, bool WIN32_MODE
= false);
short rollDie(short d);
bool drinkPotion(Asset *player); //returns false if there wasn't a potion to drink
short findAssetIndex(vector<Asset *> &assets, short assetID);

//user input:
bool getAKey(char& input, bool WIN32_MODE = false);

//file I/O:
bool saveToFile(GameProperties &game);
bool saveAssetFile(GameProperties &game, Asset &assetToSave);
bool loadFromFile(GameProperties &game);
bool loadAssetFile(GameProperties &game, Asset &assetToLoad);
void sortAssetsByIndex(vector<Asset *> &assets);
void swapAssetPointers(vector<Asset *> &assets, short a, short b);
void testPrintAssets(vector<Asset *> &assets);

//system checks:
bool isRunningInAWin32Console();
bool resizeConsole_win32(short cols, short rows);




//Execution Begins Here
int main(int argc, char** argv)
{
  srand(chrono::system_clock::now().time_since_epoch().count());

  GameProperties game;
  game.dataFolder = "gameMap1"; //default game folder
  game.userRenamedSave = false; //user has not used SaveAs to rename the load/save dir

  //get properties and assets from save files
  if (!loadFromFile(game))
  {
    cout << "Reading game from disk has failed." << endl;
    delete [] game.map;
    return 1;
  }


  //note: move this into the struct Game
  short countStatus = 0;
  game.statusDictionary.push_back("You wake up in a dank room. You hear water slowly
dripping,\nand you feel slimy mold on the floor.");
  short STATUS_INIT = countStatus++;
  game.statusDictionary.push_back("You have died. Hopefully the next adventurer will
have more luck.");
  short STATUS_DEAD = countStatus++;
  game.statusDictionary.push_back("You have been victorious in battle!\nYou press on
through the dungeon in high spirits.");
  short STATUS_VICTORY = countStatus++;

  game.currStatus = STATUS_INIT;

  //test for running in the proper console, and give a chance to quit if user wants
  bool WIN32_MODE = isRunningInAWin32Console();
  if (!WIN32_MODE)
  {
    cout << "You are not running this program in a Windows Command Prompt console.
Input and\nanimation will be more primitive. It is recommended you quit and run the
.exe\nfrom outside any IDE's.\nDo you want to continue anyway? (y/n) ";
    char choice = 0;
    cin >> choice;
    if (choice == 'n' || choice == 'N')
      return 1; //exits with Run Failed
  }
  else //user is running a win32 console; resize it to size requested by gameMap.txt
  {
    if (!resizeConsole_win32(game.screenSizeX, game.screenSizeY)) //{cols, rows}
      cout << "Resizing the console window failed. This program will not work right
with a console buffer smaller than " << game.screenSizeX << " characters wide by " <<
```

```
game.screenSizeY << " tall. Please use the settings to resize this console window
before continuing." << endl;
  }

  //game running loop
  //each iteration represents one "turn"
  char input = 0;
  bool isGameRunning = true;
  while (isGameRunning)
  {
    cls(WIN32_MODE);

    if (game.gameState == GameState::Map)
    {
      printMap(game.map, game.mapSizeX, game.mapSizeY);
      printStatus(game);
      printControlScheme(game.gameState);

      if (playerTurn(game, WIN32_MODE)) //true if player made an actual move, so give
the monsters a turn
        monstersTurn(game); //this doesn't do anything yet
    }
    else if (game.gameState == GameState::Help)
    {
      printHelp();
      printControlScheme(game.gameState);
      getAKey(input, WIN32_MODE); //get a key and trash it
      game.gameState = GameState::Map;
    }
    else if (game.gameState == GameState::Exit)
    {
      cout << endl << "Would you like to quit the game?" << endl;
      cout << "   (C) Save and quit" << endl;
      cout << "   (Y) Yes, quit now" << endl;
      cout << "   (N) Cancel" << endl;
      cout << "> ";
      if (getAKey(input, WIN32_MODE))
      {
        switch (input)
        {
          case 'C': case 'c':
            cout << "Saving game now." << endl;
            if (!saveToFile(game))
            {
              cout << "The game save failed. Are you sure you want to exit without
saving? (y/n) ";
              if (getAKey(input, WIN32_MODE))
              {
                if (input != 'Y' && input != 'y')
                {
                  game.gameState = GameState::Map;
                  break;
                }
                else
                  cout << "The game was not saved.";
              }
              else
                cout << "Key input failed." << endl;
            }
            else
            {
              cout << "Save to " << game.dataFolder << " successful." << endl;
            }
          case 'Y': case 'y':
            cout << "Thank you for playing!" << endl;
            isGameRunning = false;
            break;
          case 'N': case 'n':
          default:
            game.gameState = GameState::Map;
        }
      }
      else
      {
        cout << "Key input failed." << endl;
      }
    }
    else if (game.gameState == GameState::Dead)
    {
      game.currStatus = STATUS_DEAD;
      printMap(game.map, game.mapSizeX, game.mapSizeY);
      printStatus(game);

      printControlScheme(GameState::Dialog);
      getAKey(input, WIN32_MODE); //get a key and trash it

      cout << "Please try again!" << endl;
      isGameRunning = false;
    }
    else if (game.gameState == GameState::Load)
    {
      cout << "Are you sure you want to load a game?" << endl;
      cout << "You will loose any unsaved progress! (y/n) ";
      if (getAKey(input, WIN32_MODE))
      {
        if (input == 'Y' || input == 'y')
        {
          if (!game.userRenamedSave) //user has not provided a folder name yet
          {
            cout << "What is the folder name of your existing save game?" << endl;
            cout << "If it doesn't exist, the load will fail." << endl;
            cout << "> ";
            cin >> game.dataFolder;
            if (game.dataFolder.size() == 0)
            {
              cout << "The save folder cannot be blank." << endl;
              printControlScheme(GameState::Dialog);
              getAKey(input, WIN32_MODE); //get a key and trash it
              game.gameState = GameState::Map;
            }
            else if (game.dataFolder == "gameMap1")
            {
              cout << "That is the default load directory." << endl;
              printControlScheme(GameState::Dialog);
              getAKey(input, WIN32_MODE); //get a key and trash it
              game.gameState = GameState::Map;
            }
            else
            {
              game.userRenamedSave = true;
            }
          }

          if (game.userRenamedSave) //there is now a proper load name
          {
            if (!loadFromFile(game))
```

```
             {
               cout << "Load from folder named " << game.dataFolder << " failed!" <<
endl;
               game.userRenamedSave = false;
             }
             else
               cout << "Loaded game from " << game.dataFolder << "." << endl;
             printControlScheme(GameState::Dialog);
             getAKey(input, WIN32_MODE); //get a key and trash it
             game.gameState = GameState::Map;
           }
         }
         else // 'N' no, do not save
         {
           game.gameState = GameState::Map;
         }
       }
       else
       {
         cout << "Key input failed." << endl;
       }
     }
     else if (game.gameState == GameState::Save)
     {
       if (!game.userRenamedSave) //user has not provided a folder name yet
       {
         cout << "What is the folder name of your save game?" << endl;
         cout << "(Due to limits of the program, you must create the folder BEFORE
saving.)" << endl;
         cout << "> ";
         cin >> game.dataFolder;
         if (game.dataFolder.size() == 0)
         {
           cout << "The save folder cannot be blank." << endl;
           printControlScheme(GameState::Dialog);
           getAKey(input, WIN32_MODE); //get a key and trash it
           game.gameState = GameState::Map;
         }
         else if (game.dataFolder == "gameMap1")
         {
           cout << "You cannot save to the default load directory." << endl;
           printControlScheme(GameState::Dialog);
           getAKey(input, WIN32_MODE); //get a key and trash it
           game.gameState = GameState::Map;
         }
         else
         {
           game.userRenamedSave = true;
         }
       }

       if (game.userRenamedSave) //there is now a proper save name
       {
         if (!saveToFile(game))
         {
           cout << "Save to folder named " << game.dataFolder << " failed!" << endl;
           game.userRenamedSave = false;
         }
         else
           cout << "Saved to " << game.dataFolder << "." << endl;
         printControlScheme(GameState::Dialog);
         getAKey(input, WIN32_MODE); //get a key and trash it
         game.gameState = GameState::Map;
```

```
      }
    }
    else
    {
      cout << "Error: The game is in an invalid state and cannot continue." << endl;
      isGameRunning = false;
      printControlScheme(GameState::Dialog);
      getAKey(input, WIN32_MODE); //get a key and trash it
    }
  }

  delete [] game.map;
  return 0;
}




//*******************************************************************************
//********************************   Do Game    ********************************
//*******************************************************************************

bool playerTurn(GameProperties &game, bool WIN32_MODE)
{
  char input = 0;
  if (getAKey(input, WIN32_MODE))
  {
    if (WIN32_MODE)
      cout << endl;
    switch ((input >= 'a' && input <= 'z') ? (input - 'a' + 'A') : input) //to upper
    {
      //movement
      case 'W':
        return movePlayer(game, game.player->x, game.player->y - 1, WIN32_MODE);
//returns false if move was invalid (like trying to go out of bounds) so this turn
will not count
      case 'A':
        return movePlayer(game, game.player->x - 1, game.player->y, WIN32_MODE);
      case 'S':
        return movePlayer(game, game.player->x, game.player->y + 1, WIN32_MODE);
      case 'D':
        return movePlayer(game, game.player->x + 1, game.player->y, WIN32_MODE);

      //other game controls (quaff potion, inventory, etc)
      case 'Q':
        return drinkPotion(game.player); //true if potion was drunk and counts as a
move, false if no potion

      //meta-game control (exit, help, save)
      case 'X':
        game.gameState = GameState::Exit;
        return false;
      case 'H':
        game.gameState = GameState::Help;
        return false;
      case 'C':
        game.gameState = GameState::Save;
        return false;
      case 'V':
        game.gameState = GameState::Load;
        return false;

      //any other valid keyboard key was returned
      default:
        // cout << "Keyboard input doesn't do anything" << endl;
        return false; //incorrect key, so don't give the monsters a turn
    }
  }
  else
  {
    cout << "Key input failed." << endl;
    return false;
  }
}

//returns false if there wasn't a potion to drink
bool drinkPotion(Asset *player)
{
  // cout << "DEBUG: heals:"<<player->potionHeals<<endl;
  if (player->isPlayer)
    if (player->qtyPotion > 0)
    {
      player->hp += player->potionHeals;
      --player->qtyPotion;
      return true;
    }

  return false;
}

bool movePlayer(GameProperties &game, short x, short y, bool WIN32_MODE)
{
  // cout << "DEBUG: Moving player to (" << x << "," << y << ")" << endl;
  // cout << "DEBUG: mapSize: (" << game.mapSizeX << "," << game.mapSizeY << ")" <<
endl;
  if (x >= 0 && x < game.mapSizeX && y >= 0 && y < game.mapSizeY)
  {
    MapSquare *potentialMove = (game.map + y * game.mapSizeX + x);
    if (potentialMove->display == ' ')
    {
      overwriteSquare((game.map + game.player->y * game.mapSizeX + game.player->x),
potentialMove);
      game.player->x = x;
      game.player->y = y;
      return true;
    }
    else if (potentialMove->display == '#')
    {
      //no move, but still return true so this counts as a move
      return true;
    }
    else
    {
      //not a wall and not an empty square
      //what is it??
      if (potentialMove->linkedActor->isActor)
      {
        game.gameState = GameState::Fight;
        return fightMonster(game, potentialMove->linkedActor, game.player,
WIN32_MODE);
      }
    }
  }
  else
  {
    // cout << "DEBUG: Tried to move player out-of-bounds." << endl;
    return false;
  }
}

//This is a dumb function. It does not test for existence first, and it simply
overwrites what is in "to" and simply leaves "from" to set to ' ';
bool overwriteSquare(MapSquare *from, MapSquare *to)
{
  to->display = from->display;
  from->display = ' ';
  to->linkedActor = from->linkedActor;
  from->linkedActor = nullptr; //nothing pointer
}

bool fightMonster(GameProperties &game, Asset* monster, Asset* player, bool
WIN32_MODE)
{
  cls(WIN32_MODE);
  cout << "You are attacking a " << monster->name << "!" << endl;

  bool playerActionValid = false;
  short playerAtkRoll = 0, playerDmgRoll = 0, monsterAtkRoll = 0, monsterDmgRoll = 0,
playerDefenseBonus = 0, monsterDefenseBonus = 0;
  char input = 0;
  while (monster->hp > 0 && player->hp > 0)
  {
    playerActionValid = true;
    playerAtkRoll = 0, playerDmgRoll = 0, monsterAtkRoll = 0, monsterDmgRoll = 0,
playerDefenseBonus = 0;

    printFight(game, monster, player);
    printControlScheme(game.gameState);
    if (getAKey(input, WIN32_MODE))
    {
      cls(WIN32_MODE);
      switch ((input >= 'a' && input <= 'z') ? (input - 'a' + 'A') : input) //to upper
      {
        case 'A':
          playerAtkRoll = rollDie(20);
          cout << "You attack the " << monster->name << "! ";
          if (monsterDefenseBonus > 0) cout << "It is defending. ";
          cout << "You roll a " << playerAtkRoll << " on a 20-sided die." << endl;
          if ((playerAtkRoll + player->hitBonus) >= (monster->ac +
monsterDefenseBonus))
          {
            playerDmgRoll = rollDie(player->damage);
            cout << "You hit the " << monster->name << " with a total attack of " <<
(playerAtkRoll + player->hitBonus) << "!" << endl;
            cout << "You deal " << (playerDmgRoll + player->damageBonus) << " damage!"
<< endl;
            monster->hp -= (playerDmgRoll + player->damageBonus);
          }
          else
          {
            cout << "You miss the " << monster->name << " with your attack total of "
<< (playerAtkRoll + player->hitBonus) << "." << endl << endl;
          }
          break;

        case 'D':
          playerDefenseBonus = 5;
          cout << "You are defending! You get + " << playerDefenseBonus << " AC." <<
endl << endl << endl;
```

```
          break;

        case 'Q':
          if (playerActionValid = drinkPotion(player)) //true if potion was drunk and
counts as a move, false if no potion
            cout << "You drink a potion to heal " << player->potionHeals << " HP!" <<
endl << endl << endl;
          else
            cout << "You do not have a potion to drink." << endl << endl << endl;
          break;

        default:
          cout << "That key does nothing." << endl << endl << endl;
          playerActionValid = false; //incorrect key, so don't give the monsters a
turn
      }
    }
    else
    {
      cout << "Key input failed." << endl << endl << endl;
      playerActionValid = false;
    }

    //action the player selection was a valid turn, so give the monster a turn
    if (playerActionValid)
    {
      monsterDefenseBonus = 0;
      cout << endl;
      if (monster->hp > 0)
      {
        //monster "AI" == 80% random chance to attack, 20% to defend
        if (rollDie(10) < 9) //attack on a 1-8
        {
          monsterAtkRoll = rollDie(20);
          cout << "The " << monster->name << " attacks you! It rolls a " <<
monsterAtkRoll << " on a 20-sided die." << endl;
          if ((monsterAtkRoll + monster->hitBonus) >= (player->ac +
playerDefenseBonus))
          {
            monsterDmgRoll = rollDie(monster->damage);
            cout << "The monster hits with a total attack of " << (monsterAtkRoll +
monster->hitBonus) << "." << endl;
            cout << "It does " << (monsterDmgRoll + monster->damageBonus) << " damage
to you!" << endl;
            player->hp -= (monsterDmgRoll + monster->damageBonus);
          }
          else
          {
            cout << "The monster misses you!" << endl << endl;
          }
        }
        else //defend on a 9-10
        {
          monsterDefenseBonus = 5;
          cout << "The monster defends for +" << monsterDefenseBonus << " AC." << endl
<< endl << endl;
        }
      }
    }
    else
    {
      cout << endl << endl << endl << endl;
    }
```

```
  }

  if (monster->hp <= 0)
  {
    cout << endl << endl << endl;
    printFight(game, monster, player);
    cout << endl;
    cout << "You are victorious! You have killed the " << monster->name << "! You gain
" << monster->exp << " experience points." << endl;
    player->expTotal += monster->exp;

    printControlScheme(GameState::Dialog);
    getAKey(input, WIN32_MODE); //get a key and trash it

    //remove monster from map and game asset vector
    (game.map + monster->y * game.mapSizeX + monster->x)->display = ' ';
    (game.map + monster->y * game.mapSizeX + monster->x)->linkedActor = nullptr;
    // testPrintAssets(game.gameAssets);
    short found = findAssetIndex(game.gameAssets, monster->assetID);
    // cout << "DEBUG: monster found in asset vector as pos=" << found << endl;
    if (found != -1)
      game.gameAssets.erase(game.gameAssets.begin() + found); //don't need to re-sort,
because we're popping out in-place
    else
      cout << "Error: Monster was not in the game's asset list." << endl;
    // testPrintAssets(game.gameAssets);

    game.gameState = GameState::Map;
  }
  if (player->hp <= 0)
  {
    cout << endl << endl << endl;
    printFight(game, monster, player);
    cout << "You have been killed by the " << monster->name << "." << endl;

    //show deceased player as 'x' on the map
    (game.map + player->y * game.mapSizeX + player->x)->display = 'x';

    printControlScheme(GameState::Dialog);
    getAKey(input, WIN32_MODE); //get a key and trash it

    game.gameState = GameState::Dead;
  }
}

short rollDie(short d)
{
  if (d > 0)
    return rand() % d + 1;
  else
    return -1;
}

short findAssetIndex(vector<Asset *> &assets, short assetID)
{
  //binary search (now that vector is sorted)
  short min = 0;
  short max = assets.size() - 1;
  short center = (max / 2); //size()==10: 0...9 -> 5 / ==9: 0..8 -> 4
  while (min <= max)
  {
    center = (max + min) / 2;
    if (assets[center]->assetID == assetID)
    {
      return center;
    }
    else if (assetID > assets[center]->assetID)
    {
      min = center + 1;
    }
    else // if (assetID < assets[center]->assetID)
    {
      max = center - 1;
    }
  }

  return -1;

  //linear search:
  // for (short i = 0; i < assets.size(); ++i)
    // if (assets[i]->assetID == assetID)
      // return i;
  // return -1;
}

bool monstersTurn(GameProperties &game)
{
  //implement monster AI here!

  //move towards PC when PC is in radius monster->sightRange
  //move away from PC if monster is timid (or PC is higher level?)
  //wander around the dungeon sometimes

  //group with similar monsters?
}


//****************************************************************************
//********************************   Graphics   ********************************
//****************************************************************************

void printStatus(GameProperties &game)
{
  cout << endl;
  cout << game.statusDictionary[game.currStatus] << endl;
  cout << endl;
  cout << " [ HP: " << game.player->hp << " ]    [ Potions: " << game.player->qtyPotion
<< " ]    [ EXP: " << game.player->expTotal << " ]" << endl;
}

void printMap(MapSquare *map, short sizeX, short sizeY)
{
  short x, y;
  for (y = 0; y < sizeY; ++y)
  {
    // cout << setw(2) << y << " - "; // DEBUG
    for (x = 0; x < sizeX; ++x)
      cout << map[y * sizeX + x].display;
    cout << endl;
  }
}

void printControlScheme(GameState currState)
{
  cout << endl;
  if (currState == GameState::Map)
```

```
    cout << "Move:     (W)      (Q) Quaff Potion          (C) Save (H) Help" << endl
         << "       (A)(S)(D)                              (V) Load (X) Exit" << endl;
  else if (currState == GameState::Fight)
    cout << "     (A) Attack   (D) Defend   (Q) Quaff Potion" << endl << endl;
  else if (currState == GameState::Help || currState == GameState::Dialog)
    cout << " (R) Return to previous screen" << endl << endl;
  else
    cout << endl << endl;
  cout << "> ";
}

void printHelp()
{
  cout << "Welcome to the Adventures of \"Q\"!" << endl;
  cout << endl;
  cout << "You are an eager dungeon-delver named Q, and you must navigate a" << endl;
  cout << "dungeon full of monsters to win the game." << endl;
  cout << endl;
  cout << "Use the WASD keys to navigate the dungeon map. When you run into a
monster," << endl;
  cout << "you will fight it! You can drink a potion to regain hit points with the Q
key." << endl;
  cout << endl;
  cout << "In a battle, you can attack with your sword or defend with your shield." <<
endl;
  cout << "Hit the monster to damage its hp before it depletes yours!" << endl;
  cout << endl;
  cout << "Use the C key to save your game to a folder name you type in. It will
overwrite" << endl;
  cout << "anything previously under that name. Use the V key to load back from a
file." << endl;
  cout << endl;
  cout << endl;
  cout << "Can you defeat the terrifying dragon at the end of the dungeon??" << endl;
  cout << endl;
  cout << endl;
  cout << endl;
}

void printFight(GameProperties &game, Asset* monster, Asset* player)
{
  cout << endl;
  cout << endl;
  cout << setw(20) << "Player" << setw(20) << monster->name << endl;
  cout << setw(20) << "--------------" << setw(20) << "--------------" << endl;
  cout << setw(15) << "HP" << setw(5) << player->hp << setw(15) << "HP" << setw(5) <<
monster->hp << endl;
  cout << setw(15) << "Attack" << setw(5) << player->hitBonus << setw(15) << "Attack"
<< setw(5) << monster->hitBonus << endl;
  cout << setw(15) << "Damage 1d" << setw(2) << left << player->damage << "+" << right
<< setw(2) << player->damageBonus << setw(15) << "Damage 1d" << left << setw(2) <<
monster->damage << "+" << right << setw(2) << monster->damageBonus << endl;
  cout << setw(15) << "AC" << setw(5) << player->ac << setw(15) << "AC" << setw(5) <<
monster->ac << endl;
  cout << setw(15) << "Potions" << setw(5) << player->qtyPotion << endl;
}



//*******************************************************************************
//******************************   Save Game   ***********************************
//*******************************************************************************
```

```
bool saveToFile(GameProperties &game)
{
  if (game.userRenamedSave)
  {
    ostringstream concatenator;
    concatenator << game.dataFolder << "\\" << "gameMap.txt";
    string filename = concatenator.str();

    ofstream mapFile;
    mapFile.open(filename);
    if (mapFile.fail())
    {
      cout << "Error while opening " << filename << "." << endl;
      return false;
    }
    else
    {
      mapFile << game.screenSizeX << ' ' << game.screenSizeY << "\r\n" <<
game.mapSizeX << ' ' << game.mapSizeY << "\r\n";

      short x = 0, y = 0;
      for (short y = 0; y < game.mapSizeY; ++y)
      {
        for (short x = 0; x < game.mapSizeX; ++x)
          mapFile << game.map[y * game.mapSizeX + x].display;
        mapFile << "\r\n";
      }

      // Write asset list
      for (short i = 0; i < game.gameAssets.size(); ++i)
      {
        mapFile << game.gameAssets[i]->display << ','
                << game.gameAssets[i]->x << ','
                << game.gameAssets[i]->y << ','
                << game.gameAssets[i]->assetID << "\r\n";
        if (!saveAssetFile(game, *game.gameAssets[i]))
        {
          cout << "Saving game asset (ID#" << game.gameAssets[i]->assetID << ")
failed. Cannot save to this folder." << endl;
          mapFile.close();
          return false;
        }
      }

      mapFile.close();
      return true;
    }
  }
  else
  {
    cout << "Save file has not been renamed. Cannot save to default load folder!" <<
endl;
    return false;
  }
}

bool saveAssetFile(GameProperties &game, Asset &assetToSave)
{
  ostringstream concatenator;
  concatenator << game.dataFolder << "\\" << assetToSave.assetID << ".txt";
  string filename = concatenator.str();
  ofstream assetFile;
  assetFile.open(filename);
  if (assetFile.fail())
  {
    cout << "Error: Asset file open fail (ID#" << assetToSave.assetID << ")" << endl;
    return false;
  }
  else
  {
      assetFile << "name" << ' ' << assetToSave.name << "\r\n";
      assetFile << "isActor" << ' ' << assetToSave.isActor << "\r\n";
      assetFile << "hp" << ' ' << assetToSave.hp << "\r\n";
      assetFile << "ac" << ' ' << assetToSave.ac << "\r\n";
      assetFile << "hitBonus" << ' ' << assetToSave.hitBonus << "\r\n";
      assetFile << "damage" << ' ' << assetToSave.damage << "\r\n";
      assetFile << "damageBonus" << ' ' << assetToSave.damageBonus << "\r\n";
      assetFile << "exp" << ' ' << assetToSave.exp << "\r\n";
      assetFile << "isPlayer" << ' ' << assetToSave.isPlayer << "\r\n";
      assetFile << "qtyPotion" << ' ' << assetToSave.qtyPotion << "\r\n";
      assetFile << "potionHeals" << ' ' << assetToSave.potionHeals << "\r\n";
      assetFile << "expTotal" << ' ' << assetToSave.expTotal << "\r\n";
  }
  assetFile.close();
  return true;
}


//*******************************************************************************
//*******************************    Load Game    **********************************
//*******************************************************************************

// bool loadFromFile(string filename, char *map, short maxX, short maxY)
bool loadFromFile(GameProperties &game)
{
  ostringstream concatenator;
  concatenator << game.dataFolder << "\\" << "gameMap.txt";
  string filename = concatenator.str();

  ifstream mapFile;
  mapFile.open(filename);
  if (mapFile.fail())
  {
    cout << "Error while opening " << filename << "." << endl;
    return false;
  }
  else
  {
    short screenSizeX, screenSizeY, mapSizeX, mapSizeY;
    mapFile >> screenSizeX >> screenSizeY >> mapSizeX >> mapSizeY;

    //consume line break left in from >> operator:
    clearStreamNewlines(mapFile);

    MapSquare *newMap = new MapSquare[mapSizeX * mapSizeY]; //malloc??

    string line;
    short x = 0, y = 0;
    while (y < mapSizeY && getline(mapFile, line).good())
    {
      x = 0;
      //parse line:
      while (x < line.length() && line.at(x) != '\n' && line.at(x) != '\r')
      {
        newMap[y * mapSizeX + x].display = line.at(x++);
      }
      if (x != mapSizeX)
      {
        cout << "Improperly formatted Map file (" << filename << "). Line num " << (y
+ 1) << " is " << x << " wide, not the proper length of " << mapSizeX << "." << endl;
        mapFile.close();
        return false;
      }
      ++y;
    }
    if (y != mapSizeY && x != mapSizeX)
    {
      cout << "Map file (" << filename << ") was not " << mapSizeX << " wide by " <<
mapSizeY << " tall. It was " << x << " by " << y << "." << endl;
      mapFile.close();
      return false;
    }

    // Read asset list
    short linePos = 0;
    string currItem = "";
    Asset *playerPtr;
    vector<Asset *> newAssets;
    Asset *newAsset;
    bool foundPlayer = false;
    while (getline(mapFile, line).good())
    {
      //line is not full enough to be a full asset line. ignore it.
      if (line.length() < 1)
        continue;

      //parse line:
      linePos = 0;

      // newAsset = new Asset; // Note: The parens() are IMPORTANT! It initializes all
members of the struct to default values (zero) when called as "new Asset()"!!
      newAsset = new Asset();
      newAssets.push_back(newAsset);

      //read char display
      newAsset->display = line.at(linePos++);
      ++linePos; //skip comma

      // read x coordinate
      while (linePos < line.length() && line.at(linePos) != '\n' && line.at(linePos)
!= '\r' && line.at(linePos) != ',')
        currItem += line.at(linePos++);
      newAsset->x = atoi(currItem.c_str());
      currItem = "";
      ++linePos; //skip comma

      // read y coordinate
      while (linePos < line.length() && line.at(linePos) != '\n' && line.at(linePos)
!= '\r' && line.at(linePos) != ',')
        currItem += line.at(linePos++);
      newAsset->y = atoi(currItem.c_str());
      currItem = "";
      ++linePos; //skip comma

      // read assetID
      while (linePos < line.length() && line.at(linePos) != '\n' && line.at(linePos)
!= '\r')
```

```
         currItem += line.at(linePos++);
       newAsset->assetID = atoi(currItem.c_str());
       currItem = "";
       //P.S. I just gave in and used <sstream> in the function I wrote second,
loadAssetFile(). I only left this mess here because I worked hard on it, and I'm
therefore attached to it.

       (newMap + newAsset->y * mapSizeX + newAsset->x)->linkedActor = newAsset;
       (newMap + newAsset->y * mapSizeX + newAsset->x)->display = newAsset->display;

       if (!loadAssetFile(game, *newAsset))
       {
         cout << "Loading game asset (ID#" << newAsset->assetID << ") failed. Cannot
load this save folder." << endl;
         mapFile.close();
         return false;
       }

       if (newAsset->isPlayer)
       {
         if (foundPlayer) //already found
           cout << "More than one player asset was found. This is surely an error. Only
the first one will be on your side..." << endl;
         else
           playerPtr = newAsset;
         foundPlayer = true;
       }
     }
     if (!foundPlayer)
     {
       cout << "A player asset was not loaded from this save file. This game file
cannot be loaded." << endl;
       mapFile.close();
       return false;
     }

     //Future idea: Move x/y and display from mapFile to assetFiles. They really belong
there...

     //Future idea: add capability to put Items at the end of the list of Actors

     mapFile.close();

     //user has asked to load/save already, therefore this is NOT an initial load on
game start, so delete previous GameProperties
     if (game.userRenamedSave)
     {
       delete [] game.map;
       game.gameAssets.clear();
     }

     //set up game as it should start
     game.screenSizeX = screenSizeX;
     game.screenSizeY = screenSizeY;
     game.mapSizeX = mapSizeX;
     game.mapSizeY = mapSizeY;
     game.map = newMap;
     game.player = playerPtr;
     for (int i = 0; i < newAssets.size(); ++i)
       game.gameAssets.push_back(newAssets[i]);

     // testPrintAssets(game.gameAssets);
     sortAssetsByIndex(game.gameAssets);
```

```
    // testPrintAssets(game.gameAssets);

    //start all loaded games from the main map screen
    game.gameState = GameState::Map;

    return true;
  }
}


bool loadAssetFile(GameProperties &game, Asset &assetToLoad)
{
  ostringstream concatenator;
  concatenator << game.dataFolder << "\\" << assetToLoad.assetID << ".txt";
  string filename = concatenator.str();
  ifstream assetFile;
  assetFile.open(filename);
  if (assetFile.fail())
  {
    cout << "Error: Asset file open fail (ID#" << assetToLoad.assetID << ")" << endl;
    return false;
  }
  else
  {
    //didn't really want to use a string stream, but the alternative is either
to_string() (which my complier isn't finding in <string> even with my complier set to
C++11 mode...) or the big mess of atoi(valueString.c_str()). ug.
    stringstream reader;
    string line, attribute;
    while (getline(assetFile, line).good())
    {
      reader << line;
      reader >> attribute;
      if (attribute == "name")
        reader >> assetToLoad.name;
      else if (attribute == "isActor")
        reader >> assetToLoad.isActor;
      else if (attribute == "hp")
        reader >> assetToLoad.hp;
      else if (attribute == "ac")
        reader >> assetToLoad.ac;
      else if (attribute == "hitBonus")
        reader >> assetToLoad.hitBonus;
      else if (attribute == "damage")
        reader >> assetToLoad.damage;
      else if (attribute == "damageBonus")
        reader >> assetToLoad.damageBonus;
      else if (attribute == "exp")
        reader >> assetToLoad.exp;
      else if (attribute == "isPlayer")
        reader >> assetToLoad.isPlayer;
      else if (attribute == "qtyPotion")
        reader >> assetToLoad.qtyPotion;
      else if (attribute == "potionHeals")
        reader >> assetToLoad.potionHeals;
      else if (attribute == "expTotal")
        reader >> assetToLoad.expTotal;
    }
    //DEBUG:
    // cout << "name: " << assetToLoad.name << endl;
    // cout << "display: " << assetToLoad.display << endl;
    // cout << "x: " << assetToLoad.x << endl;
    // cout << "y: " << assetToLoad.y << endl;
    // cout << "assetID: " << assetToLoad.assetID << endl;
    // cout << "isPlayer: " << assetToLoad.isPlayer << endl;
    // cout << "isActor: " << assetToLoad.isActor << endl;
    // cout << "hp: " << assetToLoad.hp << endl;
    // cout << "ac: " << assetToLoad.ac << endl;
    // cout << "hitBonus: " << assetToLoad.hitBonus << endl;
    // cout << "damage: " << assetToLoad.damage << endl;
    // cout << "damageBonus: " << assetToLoad.damageBonus << endl;
    // cout << "qtyPotion: " << assetToLoad.qtyPotion << endl;
    // cout << "potionHeals: " << assetToLoad.potionHeals << endl;
    // cout << "expTotal: " << assetToLoad.expTotal << endl;
    // cout << "exp: " << assetToLoad.exp << endl;
  }
  assetFile.close();
  return true;
}

void testPrintAssets(vector<Asset *> &assets)
{
  for (short i = 0; i < assets.size(); ++i)
    cout << "assets[" << i << "]=" << assets[i]->assetID << endl;
}

void sortAssetsByIndex(vector<Asset *> &assets)
{
  //implement a selection sort
  short smallest = 0;
  for (short i = 0; i < (assets.size() - 1); ++i)
  {
    smallest = i;
    for (short j = i; j < assets.size(); ++j)
    {
      if (assets[j]->assetID < assets[smallest]->assetID)
        smallest = j;
    }
    if (smallest != i) //actually found one that needs swapping
      swapAssetPointers(assets, smallest, i);
  }
}

void swapAssetPointers(vector<Asset *> &assets, short a, short b)
{
  Asset *temp = assets[a];
  assets[a] = assets[b];
  assets[b] = temp;
}

//tests if there is a win32 console running this program right now
bool isRunningInAWin32Console()
{
  // uses a function (from the Input Buffer Read method), then ignores the result
  DWORD fdwSaveOldMode;
  if (GetConsoleMode(GetStdHandle(STD_INPUT_HANDLE), &fdwSaveOldMode))
    return true; //console was found
  else
    return false; //console not found
}


//MSDN Method to resize the console window, increase buffer size
//https://msdn.microsoft.com/en-us/library/windows/desktop/ms686044(v=vs.85).aspx
bool resizeConsole_win32(short cols, short rows)
{
  HANDLE hStdout = GetStdHandle(STD_OUTPUT_HANDLE);
  COORD consoleSize = {cols, rows}; //{cols, rows}
  SMALL_RECT windowSize = {0, 0, static_cast<short>(cols - 1), static_cast<short>(rows
- 1)};
  if(!SetConsoleScreenBufferSize(hStdout, consoleSize))
  {
    cout << "Setting the console size failed." << endl;
    return false;
  }
  if(!SetConsoleWindowInfo(hStdout, true, &windowSize ))
  {
    cout << "Setting the window size failed." << endl;
    return false;
  }
  return true;
}


//****************************************************************************
//*********************************   Input   ***********************************
//****************************************************************************

//MSDN Method to read unbuffered console input
//https://msdn.microsoft.com/en-us/library/windows/desktop/ms685035(v=vs.85).aspx
bool getAKey(char& input, bool WIN32_MODE)
{
  if(!WIN32_MODE)
  {
    cin >> input;
    return true;
  }
  else
  {
    cout.flush(); //necessary to display anything in cout before ReadConsoleInput
starts polling?

    HANDLE hStdin;
    DWORD fdwSaveOldMode;
    DWORD cNumRead, fdwMode;
    const short BUFF_SIZE = 8; // original buffer size: const int BUFF_SIZE = 128;
    INPUT_RECORD irInBuf[BUFF_SIZE];

    // Get the standard input handle.
    hStdin = GetStdHandle(STD_INPUT_HANDLE);
    if (hStdin == INVALID_HANDLE_VALUE)
    {
      cout << "Error: GetStdHandle" << endl;
      // SetConsoleMode(hStdin, fdwSaveOldMode); // Restore input mode on exit.
      return false; //ExitProcess(0);
    }

    // Save the current input mode, to be restored on exit.
    if (!GetConsoleMode(hStdin, &fdwSaveOldMode))
    {
      cout << "Error: Getting the Console State Failed: No Console Found\nPlease run
this program in a Windows Command Prompt console if you would like to enable input w/o
pressing the enter key." << endl;
      // SetConsoleMode(hStdin, fdwSaveOldMode); // Restore input mode on exit.
      return false; //ExitProcess(0);
    }

    // change console mode to non-buffered input, allowing the Windows console to send
input to C++ w/o waiting for the {Enter} key
```

```
    fdwMode = ENABLE_WINDOW_INPUT;
    if (!SetConsoleMode(hStdin, fdwMode))
    {
      cout << "Error: SetConsoleMode" << endl;
      SetConsoleMode(hStdin, fdwSaveOldMode); // Restore input mode on exit.
      return false; //ExitProcess(0);
    }

    // Loop to read and handle inputs until a valid character is read
    KEY_EVENT_RECORD keyEventRecord;
    int i = 0;
    bool isInputDone = false;
    while (!isInputDone)
    {
      // Wait for the events.
      if (!ReadConsoleInput(hStdin, irInBuf, BUFF_SIZE, &cNumRead))
      {
        cout << "Error: While waiting for ReadConsoleInput" << endl;
        SetConsoleMode(hStdin, fdwSaveOldMode); // Restore input mode on exit.
        return false; //ExitProcess(0);
      }

      // if (cNumRead > 1)
        // cout << "Warning: For this cycle, more than one (" << cNumRead << ") input
event was read." << endl;

      // Check all events in the buffer (should be only one)
      for (i = 0; i < cNumRead; i++)
      {
        switch(irInBuf[i].EventType) //filter out all events that are not KEY_EVENT
        {
          case KEY_EVENT: // keyboard input
            keyEventRecord = irInBuf[i].Event.KeyEvent;
            //note: the hexadecimal "Virtual Key Codes" are from MSDN:
https://msdn.microsoft.com/en-us/library/windows/desktop/dd375731(v=vs.85).aspx
            // ctrl+c: has to check dwControlKeyState using bit masks
            if (keyEventRecord.wVirtualKeyCode == 0x43 &&
(keyEventRecord.dwControlKeyState & 0x0008 || keyEventRecord.dwControlKeyState &
0x0004) && !keyEventRecord.bKeyDown)
            {
              cout << "Ctrl+C has been used to kill the process." << endl;
              ExitProcess(0); //from MSDN methods
            }
            if (!keyEventRecord.bKeyDown) //if the keyboard key has not been released
yet, ignore it
            {
              isInputDone = true; //assume input is good. change it to false if input
is actually bad

              //test if this is an allowed character
              if (keyEventRecord.wVirtualKeyCode >= '0' &&
keyEventRecord.wVirtualKeyCode <= '9')
                input = keyEventRecord.wVirtualKeyCode; // - 0x30 + '0'; //note: that
was useless, 0x30=='0'==48. I need to check my ASCII table better!
              else if (keyEventRecord.wVirtualKeyCode >= 'A' &&
keyEventRecord.wVirtualKeyCode <= 'Z')
                input = keyEventRecord.wVirtualKeyCode;
              else if (keyEventRecord.wVirtualKeyCode == ' ') //space bar
                input = ' ';
              else if (keyEventRecord.wVirtualKeyCode == 13) //carriage return
                input = '\n';
              else
                isInputDone = false;
              if (isInputDone)
                // printf("%c", input); //no longer needed b/c I figured out
cout.flush() before opening up the buffer to wait for input
                cout << input;
            }
          break;

          // Ignore all other potential event types
          case MOUSE_EVENT: // mouse input
          case WINDOW_BUFFER_SIZE_EVENT: // scrn buf. resizing
          case FOCUS_EVENT:  // disregard focus events
          case MENU_EVENT:   // disregard menu events
          default:
            break;
        }
      }
    }

    // Restore input mode on exit.
    SetConsoleMode(hStdin, fdwSaveOldMode);
    return true; //no errors
  }
}

//clear screen (will do it nicely using MSDN's console-handle method if enabled by
WIN32_MODE
bool cls(bool WIN32_MODE)
{
  if (WIN32_MODE)
    cls_win32();
  else //will create a lot of flicker! ew.
    for (int i = 0; i < 30; ++i)
      cout << endl;
}

//MSDN method of clearing the console by writing ' ' to every spot
//Reference: https://msdn.microsoft.com/en-
us/library/windows/desktop/ms682022(v=vs.85).aspx
bool cls_win32()
{
  HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
  COORD coordScreen = {0, 0}; // top, left corner of the console
  DWORD cCharsWritten;
  CONSOLE_SCREEN_BUFFER_INFO csbi;
  DWORD dwConSize;

  // Get the number of character cells in the current buffer.
  if(!GetConsoleScreenBufferInfo(hConsole, &csbi))
    return false;
  dwConSize = csbi.dwSize.X * csbi.dwSize.Y;

  // Fill the entire screen with blanks.
  if( !FillConsoleOutputCharacter(hConsole, (TCHAR) ' ', dwConSize, coordScreen,
&cCharsWritten))
    return false;
  // Get the current text attribute.
  if(!GetConsoleScreenBufferInfo(hConsole, &csbi))
    return false;
  // Set the buffer's attributes accordingly.
  if(!FillConsoleOutputAttribute(hConsole, csbi.wAttributes, dwConSize, coordScreen,
&cCharsWritten)) // Receive number of characters written
    return false;
  // Put the cursor at its home coordinates.
  SetConsoleCursorPosition(hConsole, coordScreen);

  return true;
}

//function from Savitch 9th ed Ch 6 pg 347
void clearStreamNewlines(istream &strm)
{
  char temp;
  do
  {
    strm.get(temp);
  } while (temp != '\n' && temp != '\0');
}
```