**Game Design Document**

**Project Name:** Treasure Cave

**Team Name:** GSP 115

**Team Members:**

Joshua Kidder – Student / Programmer

**Game Design Document Version Control**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Change Description** |
| 1.1 | Jul/09 | Joshua Kidder | Add variables |
| 1.2 | Jul/16 | Joshua Kidder | Add two dimensional array, place treasure and monster in cave |
| 1.3 | Jul/23 | Joshua Kidder | Add a second two dimensional array, add weapon, if found will kill monster, modify monster to move and guard treasure |
| 1.4 | Jul/30 | Joshua Kidder | Add blank space for each location, introduce a dark cave, add torch, add a cave exit, add noise maker to stop monster, add else clause (player moves back two spaces without weapon) |
| 1.5 | Aug/06 | Joshua Kidder | Clean up code using functions |
| 1.6 | Aug/13 | Joshua Kidder | Change combat results, change the win condition, add color text, |
| 1.7 | Aug/20 | Joshua Kidder | Add new threat thief and object eye, add ability for thief to interact with player and monster, add ability for thief to find treasure |
| 1.8 | Aug/22 | Joshua Kidder | Bug fixes, Document updates |

**TABLE OF CONTENTS**

**GAME DESCRIPTION ............................................................................................................................... 4**

Design Goals............................................................................................................................................................... 4

Influences & Sources.............................................................................................................................................. 4

Target Market............................................................................................................................................................ 4

User Interface............................................................................................................................................................ 4

**GAME MECHANICS .................................................................................................................................. 5**

Core Game Play..........................................................................................................................................................5

Game Flow...................................................................................................................................................................5

Game Objects..............................................................................................................................................................5

Characters....................................................................................................................................................................6

Enemies.........................................................................................................................................................................6

Game Play………………………………………………………………………………………………………………………7

Initial Interaction……….…………………………………………………………………………………………………..7

Additional Interaction…………………………………………………………………………………………………...11

**TECHNICAL SPECIFICATIONS ............................................................................................................. 15**

Data Flow..................................................................................................................................................................15

courseProject\_Final.cpp………………………………………………………………………………………………..15

courseProject\_Final.h……………………………………………………………………………………………………34

**Game Description**

Treasure Cave is a text-based maze style adventure game, played out on a 2D model board (or virtual map), that takes place in a dark mysterious cave of an unknown location. Rumors spread of a treasure beyond your wildest dream guarded by a gruesome monster, while seekers go searching and never return. The cave is rumored to hold weapons of other adventures who never returned; a sword, a torch, a noise maker and a magical eye; as well as a monster, a treasure and an elusive enemy.

***Design Goals***

The game aims to achieve the following goals:

1. Establish the Treasure Cave and familiarize players with the game play and events

of Treasure Cave.

2. Serve as a prototype and first level of an ongoing serialized 2D adventure style game.

3. Develop a fun gameplay experience, based on:

a. Discovery – player discovers the cave, obstacles, and treasure that awaits.

b. Power – the game is about finding the treasure, fighting your enemies and

emerging victorious through several battles.

***Target Market***

1. Professor Genevieve and the GSP115 Class

The ideal user has the following characteristics:

1. Male or Female
2. 8+ years old
3. Likes games
4. Plays mobile games

***User Interface***

This game uses a text interface. Player enters commands followed by the enter key.

**Game Mechanics**

Core Game Play

The player moves in a 2D (two dimensional) plane, exploring a dark unlit cave that extends from the left to the right and top to the bottom of the screen. Enemies appear along the way as the player tries to find the treasure, and the player must defeat them if encountered.

The player may also find objects within the cave that help the player defeat the enemies. As the player advances through the cave, specific objects will trigger abilities that rewarded the player with advancement. The player will be alerted when a special object is found.

Game Flow

Actions that the player can perform are:

1. Move west (a) and east (d).
2. Move north (w) and south (s).
3. Make an irritating noise (n).
4. Attack once.
5. Quit game (x).

Game Objects

A list of things that can be found in the cave:

1. Weapon
2. Torch
3. Noisemaker
4. Eye
5. Thief
6. Monster
7. Treasure
8. Cave Exit

Character

1. Adventurer: Sword – wielding warrior that feels no pain, is fearless in the face of the enemy and can withstand tremendous punishment.

Enemies

Enemies appear on screen from either the top, bottom, right or left side.

1. Thief:
2. Enemy will start in a specific position within the cave.
3. Enemy will be invisible to the player at the start of game.
4. Enemy will be invisible to player if Torch is found.
5. Enemy is only visible to the player if the Eye is found.
6. With each move of the player, thief will make a random move.
7. Enemy will attack player as soon as player is in range.
8. If player dies when attacked game is over.
9. Enemy can attack monster if encountered.
10. If thief or monster dies, game continues.
11. Enemy can find treasure.
12. If found game is over.
13. Monster:
14. Enemy will be positioned within the cave along with the treasure.
15. Enemy will be invisible to the player at the start of game.
16. Enemy will be visible to player if Torch is found.
17. Enemy will be visible to the player if the Eye is found before Torch.
18. Enemy will stop moving if noise maker is found and used.
19. The enemy will stop for at least one turn,

then has 1 out of 3 chances to move again.

1. With each move of the player, enemy will make a move around the treasure.
2. Enemy will attack player as soon as player is in range.
3. If player finds monster and has no weapon,

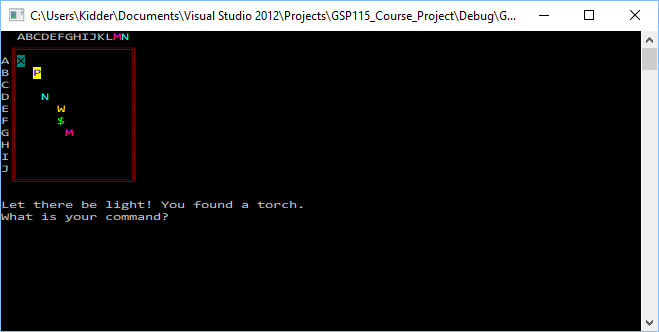
player will move back two spaces.

1. If monster finds player and player has no weapon, player dies, game over.
2. If player has weapon and dies when attacked game is over.
3. Thief can attack monster if encountered.
4. If thief or monster dies, game continues.
5. Monster cannot interact with treasure.

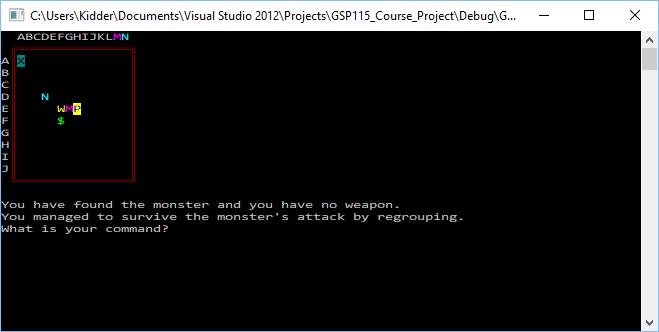
Game Play

***Initial Interaction***

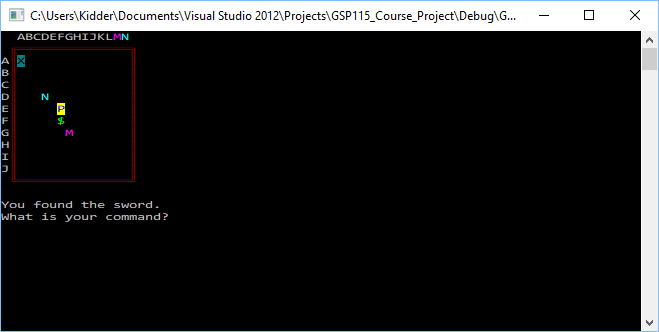
Player finds torch, reveals all objects except thief and eye



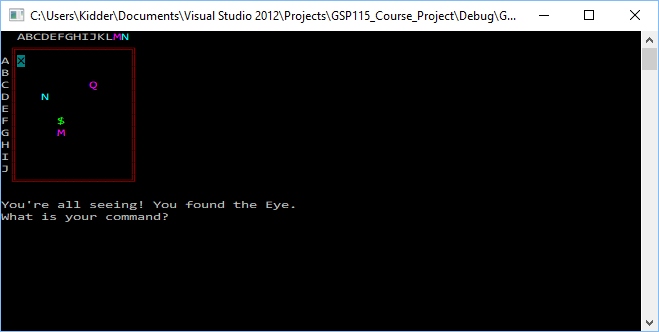
Player has no weapon, finds monster, moves back two spaces



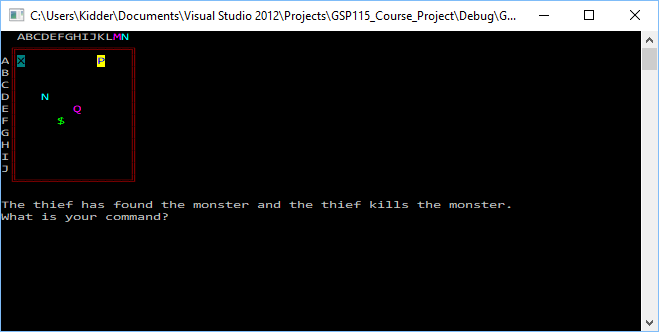
Player finds weapon



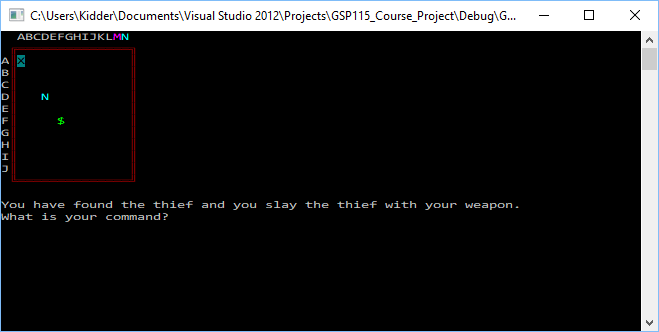
Player finds eye, reveals location of thief



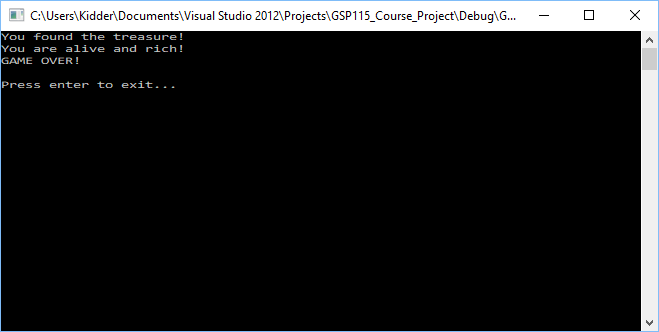
Thief kills monster



Player kills thief

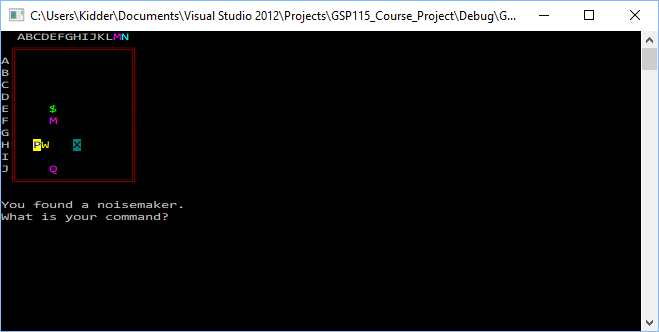


Player finds treasure, game over

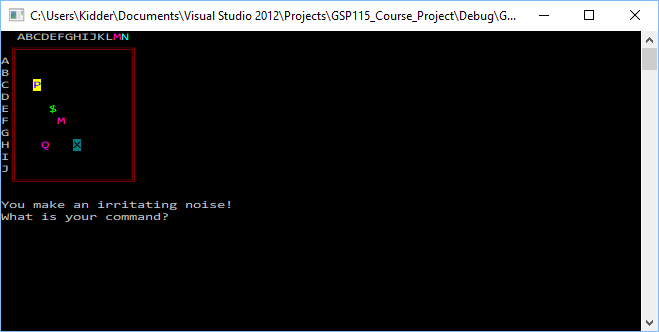


***Additional Interaction***

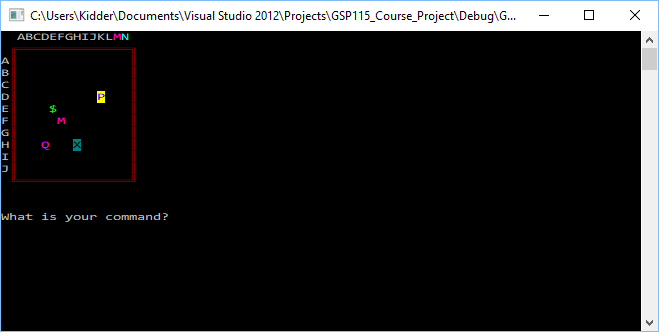
Player finds noisemaker



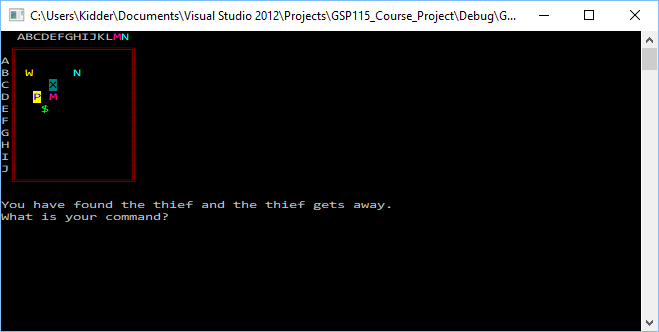
Player makes an irritating noise



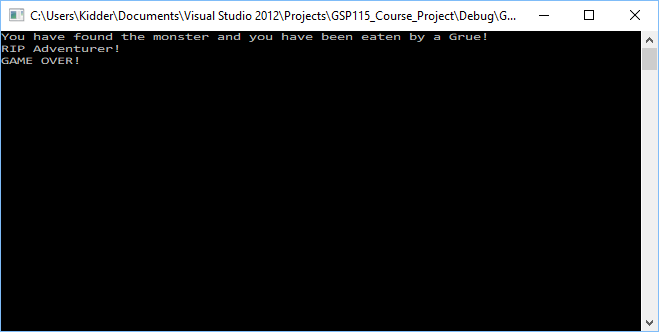
Thief and monster paused



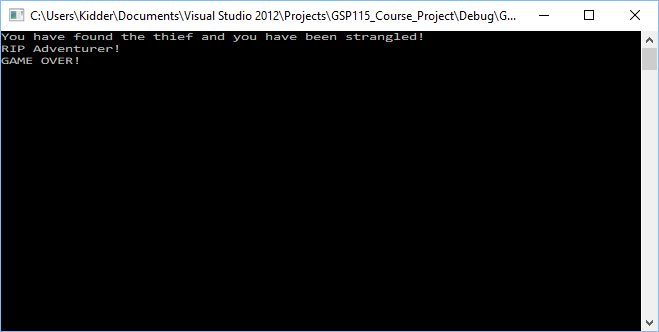
Player finds thief, thief escapes



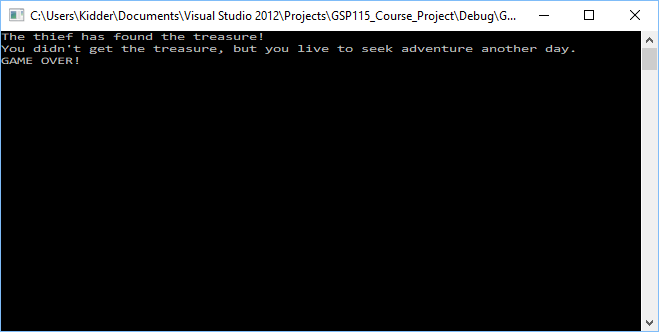
Player finds monster, monster eats player



Player finds thief, thief strangles player



Thief finds treasure



**Technical Explanation**

Data Flow

***courseProject\_Final.cpp***

// GSP115\_Course\_Project.cpp : Defines the entry point for the console application.

//

// Joshua Kidder

// GSP115 TREASURE CAVE

// Week 8

#pragma once

#include "stdafx.h"

#include "String.h"

#include <iostream>

#include <array>

#include <string>

#include <time.h>

//#include "GSP115\_Course\_Project.h"

// Global Constants

const int MAX\_ROWS = 10;

const int MIN\_ROWS = 0;

const int MAX\_COLS = 14;

const int MIN\_COLS = 0;

const int TOTAL\_ROWS = MAX\_ROWS + 1;

const int TOTAL\_COLS = MAX\_COLS + 1;

// <WK6 status=permanent>

// Combat Result Constants

const int PLAYER\_DIES\_WITH\_WEAPON = 80;

const int MONSTER\_DIES\_WITH\_WEAPON = 30;

const int PLAYER\_DIES\_NO\_WEAPON = 20;

const int MONSTER\_DIES\_NO\_WEAPON = 18;

// <WK6>

// <WK7 status=permanent>

const int PLAYER\_DIES\_TO\_THIEF = 20;

const int THIEF\_DIES\_TO\_PLAYER = 80;

// <WK7>

// <WK3 status=permanent>

//Display Constants

const char ULC = 201; //Upper left corner

const char HB = 205; //Horizontal border

const char URC = 187; //Upper right corner

const char VB = 186; //Vertical border

const char LRC = 188; //Lower right corner

const char LLC = 200; //Lower left corner

const char MT = ' '; //Empty location

const char PSymbol = 'P'; //Player symbol

// <WK7 status=permanent>

const char QSymbol = 'Q'; //Thief symbol

// </WK7>

const char TOSymbol = 'T'; //Torch symbol

const char WSymbol = 'W'; //Weapon symbol

const char TRSymbol = '$'; //Treasure symbol

const char MSymbol = 'M'; //Monster symbol

const char NSymbol = 'N'; //Noisemaker symbol

// <WK7 status=permanent>

const char YSymbol = 'Y'; //Eye symbol

// </WK7>

const char XSymbol = 'X'; //Cave exit symbol

// </WK3>

using namespace std;

#include "GSP115\_Course\_Project.h"

#include <string.h>

// <WK3 status=permanent>

// function prototypes

// <WK8 status=permanent>

void printCave(gameObjectType cave[TOTAL\_ROWS][TOTAL\_COLS]);

// </WK8>

gameObject placeInCave(gameObjectType array[TOTAL\_ROWS][TOTAL\_COLS]);

bool showOnBoard(gameObject x);

bool checkVisible(gameObject x, gameObject y, int dist);

// <WK6 status=permanent>

//this function is used to resolve the combat between the player and monster

bool resolveCombat(playerObject &p, gameObject &m,int youDie, int monsterDie, string &str, bool &moveP, bool &mPause, gameObjectType cave[TOTAL\_ROWS][TOTAL\_COLS], int& row, int& column, char board[TOTAL\_ROWS + 3][TOTAL\_COLS + 3]);

// </WK6>

// <WK7 status=permanent>

//this function is used to resolve the combat between the player and thief

bool resolvethiefCombat(playerObject &p, gameObject &q, int youDie, int thiefDie, string &str, bool &moveP, bool &qPause, gameObjectType cave[TOTAL\_ROWS][TOTAL\_COLS], int& row, int& column, char board[TOTAL\_ROWS + 3][TOTAL\_COLS + 3]);

//this function is used to resolve the combat between the thief and monster

bool resolvethiefMonterCombat(gameObject &monster, gameObject &q, int youDie, int thiefDie, string &str, bool &moveP, bool &qPause, gameObjectType cave[TOTAL\_ROWS][TOTAL\_COLS], int& row, int& column, char board[TOTAL\_ROWS + 3][TOTAL\_COLS + 3]);

// </WK7>

// <WK5 status=permanent>

// function prototypes

// this function is used to process user input commands

void processPlayerCommand(char command, int& row, int& column, playerObject& player, bool& movePlayer, string& msg, bool& monsterPause, bool& thiefPause, bool& gameOver);

// this function is used to check how the user interacts with objects

void checkForEvents(gameObjectType cave[TOTAL\_ROWS][TOTAL\_COLS], int& row, int& column, playerObject& player, string& msg, gameObject& thief, gameObject& treasure, gameObject& weapon, gameObject& monster, gameObject& noisemaker, gameObject& torch, gameObject& eye, gameObject& caveExit, bool& gameOver, bool& movePlayer, bool& monsterPause, bool& thiefPause, int& killedValue, int& killMonsterValue, int& killedThiefValue, int& killThiefValue,

char board[TOTAL\_ROWS + 3][TOTAL\_COLS + 3]);

// this function is used to process how the monster interacts with objects

void processMonster(bool monsterPause, gameObjectType cave[TOTAL\_ROWS][TOTAL\_COLS], gameObject& monster, int& MonsterMoveCounter, playerObject& player, string& msg, bool& gameOver, int& killedValue, int& killMonsterValue, bool& movePlayer, gameObjectType& hold,

char board[TOTAL\_ROWS + 3][TOTAL\_COLS + 3]);

// </WK5>

// <WK7 status=permanent>

// process thief

void processThief(bool thiefPause, gameObjectType cave[TOTAL\_ROWS][TOTAL\_COLS], gameObject& thief, int& thiefMoveCounter, playerObject& player, string& msg, bool& gameOver, int& killedThiefValue, int& killThiefValue, bool& movePlayer, gameObjectType& hold, char board[TOTAL\_ROWS + 3][TOTAL\_COLS + 3], gameObject& monster);

// </WK7>

// <WK6 status=permanent>

void setcolor(int c);

void gamecolor(unsigned char type);

// </WK6>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

MAIN() FUNCTION

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int \_tmain(int argc, \_TCHAR\* argv[])

{

//\*\*Initialize Variables\*\*

srand((unsigned int)time(NULL)); // Seed the random number function

// <WK7 status=permanent>

gameObjectType cave[TOTAL\_ROWS][TOTAL\_COLS]; // the cave--a two dimensional array

char board[TOTAL\_ROWS + 3][TOTAL\_COLS + 3] = // the game board--a two dimensional array

{

{MT,MT,'A','B','C','D','E','F','G','H','I','J','K','L','M','N', MT},

{MT,ULC,HB, HB, HB, HB, HB, HB, HB, HB, HB, HB, HB, HB, HB, HB, URC},

{'A',VB,MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, VB},

{'B',VB,MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, VB},

{'C',VB,MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, VB},

{'D',VB,MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, VB},

{'E',VB,MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, VB},

{'F',VB,MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, VB},

{'G',VB,MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, VB},

{'H',VB,MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, VB},

{'I',VB,MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, VB},

{'J',VB,MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, MT, VB},

{MT,LLC,HB, HB, HB, HB, HB, HB, HB, HB, HB, HB, HB, HB, HB, HB, LRC}

};

// </WK7>

// <WK6 status=permanent>

gameObjectType hold = EMPTY; // Holds objects under the monster<bug fix>

// </WK6>

// <WK8 status=permanent>

gameObjectType monsterHold = EMPTY; // Holds objects under the monster<bug fix>

gameObjectType thiefHold = EMPTY; // Holds objects under the thief<bug fix>

// </WK8>

playerObject player = { true,false,false,false,false,false, { -1, -1, false, true } }; // the player

// <WK7 status=permanent>

gameObject thief = { -1, -1, false, true,true }; // the thief

// </WK7>

gameObject treasure = { -1, -1, false, true,true }; // the treasure

gameObject monster = { -1, -1, false, true,true }; // the monster

gameObject weapon = { -1, -1, false, true,true }; // the weapon

// <WK4 status=permanent>

gameObject torch = { -1, -1, false, true,true }; // the torch

gameObject noisemaker = { -1, -1, false, true,true }; // the noisemaker

// </WK4>

// <WK7 status=permanent>

gameObject eye = { -1, -1, false, true }; // the eye

// </WK7>

// <WK6 status=permanent>

gameObject caveExit = {-1, -1, false, true,true}; // the cave exit

// </WK6>

int row, column; // temporarily hold the new player position

int MonsterMoveCounter = 6; // track and control monster movement around treasure

// <WK7 status=permanent>

int thiefMoveCounter = 10; // track and control thief movement around board

// </WK7>

// <WK6 status=permanent>

// a random value above this level means you are killed by the monster

int killedValue = PLAYER\_DIES\_NO\_WEAPON;

// a random value above this level and below killedValue means you kill the monster

int killMonsterValue = MONSTER\_DIES\_NO\_WEAPON;

// </WK6>

// <WK7 status=permanent>

// a random value above this level means you are killed by the thief

int killedThiefValue = PLAYER\_DIES\_TO\_THIEF;

// a random value above this level and below killedValue means you kill the thief

int killThiefValue = THIEF\_DIES\_TO\_PLAYER;

// </WK7>

string msg; // status message variable

char command; // player input

// <WK4 status=permanent>

bool monsterPause = false; // flag to indicate the monster has stopped moving

// </WK4>

// <WK7 status=permanent>

bool thiefPause = false; // flag to indicate the thief has stopped moving

// </WK7>

bool movePlayer = true; // flag to indicate the player position can be updated

bool gameOver = false; // status flag

//\*\*Prepare Cave\*\*\*\*\*\*\*\*\*\*\*

//..Initialize an empty cave

for (gameObjectType (&R)[TOTAL\_COLS] : cave)

{

for (auto &C : R) C = EMPTY;

}

//...Add player in rows 0-2, columns 0-2

player.position.row = rand() %3;

player.position.column = rand() %3;

// <WK5 status=permanent>

// set back positions

player.oneBackPosition = player.position;

player.twoBackPosition = player.position;

// </WK5>

cave[player.position.row][player.position.column] = PLAYER;

// <WK7 status=permanent>

//...Add Thief in row 8, column 0

thief.row = 8;

thief.column = 0;

cave[thief.row][thief.column] = THIEF;

// </WK7>

//...Add Treasure in rows 4-6, column 1-6

treasure.row = rand() %3 + 4;

treasure.column = rand() %8 + 2;

cave[treasure.row][treasure.column] = TREASURE;

//...Add Monster at treasure row +1, column -1

monster.row = treasure.row + 1;

monster.column = treasure.column - 1;

cave[monster.row][monster.column] = MONSTER;

//...Add Weapon in any empty location

weapon = placeInCave(cave);

cave[weapon.row][weapon.column] = WEAPON;

// <WK4 status=permanent>

//...Add Noisemaker in any empty location

noisemaker = placeInCave(cave);

cave[noisemaker.row][noisemaker.column] = NOISEMAKER;

//...Add Torch in any empty location

torch = placeInCave(cave);

cave[torch.row][torch.column] = TORCH;

// </WK4>

// <WK7 status=permanent>

//...Add Eye in any empty location

eye = placeInCave(cave);

cave[eye.row][eye.column] = EYE;

// </WK7>

// <WK6 status=permanent>

//...Add Cave Exit in any empty location.

caveExit = placeInCave(cave);

cave[caveExit.row][caveExit.column] = CAVE\_EXIT;

// </WK6>

//\*\*Play Game\*\*\*\*\*\*\*\*\*\*\*\*\*

//...Begin Game Loop

while (!gameOver)

{

// <WK7 status=permanent>

//....Display Board

//.....Check visibility

if (!(player.hasTorch || player.hasEye))

{

torch.isVisible = checkVisible(player.position, torch, 2);

weapon.isVisible = checkVisible(player.position, weapon, 2);

treasure.isVisible = checkVisible(player.position, treasure, 2);

monster.isVisible = checkVisible(player.position, monster, 2);

noisemaker.isVisible = checkVisible(player.position, noisemaker, 2);

// <WK7 status=permanent>

thief.isVisible = checkVisible(player.position, thief, 2);

eye.isVisible = checkVisible(player.position, eye, 1);

// </WK7>

// <WK6 status=permanent>

caveExit.isVisible = checkVisible(player.position, caveExit, 2);

// </WK7>

}

// </WK7>

//.....Place visible objects on board: changing order will create a visual bug.

board[weapon.row + 2][weapon.column + 2] = showOnBoard(weapon) ? WSymbol : MT;

// <WK4 status=permanent>

board[torch.row + 2][torch.column + 2] = showOnBoard(torch) ? TOSymbol : MT;

// </WK4>

board[treasure.row + 2][treasure.column + 2] = showOnBoard(treasure) ? TRSymbol: MT;

// <WK4 status=permanent>

board[noisemaker.row + 2][noisemaker.column + 2] = showOnBoard(noisemaker) ?

NSymbol : MT;

// </WK4>

// <WK6 status=permanent>

board[caveExit.row + 2][caveExit.column + 2] = showOnBoard(caveExit) ? XSymbol : MT;

// </WK6>

board[monster.row + 2][monster.column + 2] = showOnBoard(monster) ? MSymbol : MT;

board[player.position.row + 2][player.position.column + 2] = player.alive ? PSymbol : MT;

// <WK7 status=permanent>

board[thief.row + 2][thief.column + 2] = showOnBoard(thief) ? QSymbol : MT;

board[eye.row + 2][eye.column + 2] = showOnBoard(eye) ? YSymbol : MT;

// </WK7>

// Put the board on the screen

for (char(&R)[TOTAL\_COLS + 3] : board)

{

for (char &C : R)

{

// <WK6 status=permanent>

gamecolor(C);

// </WK6>

cout << C;

}

cout << endl;

}

// <WK6 status=permanent>

gamecolor('Z');

// </WK6>

cout << msg.c\_str() << endl;

//printCave(cave);

//....Get command

cout << "What is your command? ";

cin >> command;

//....Clear display and message

msg.clear();

system("cls");

// <WK7 status=permanent>

// process player command

processPlayerCommand(command, row, column, player, movePlayer, msg,

monsterPause, thiefPause, gameOver);

// </WK7>

//....Check if the game is over

if (!gameOver)

{

// <WK7 status=permanent>

checkForEvents(cave, row, column, player, msg, thief, treasure, weapon,

monster, noisemaker, torch, eye, caveExit, gameOver, movePlayer, monsterPause, thiefPause, killedValue, killMonsterValue, killedThiefValue, killThiefValue, board);

// </WK7>

//.....Move Player

if (movePlayer)

{

//updates position information

cave[player.position.row][player.position.column] = EMPTY;

//clear the screen where player was

cave[row][column] = PLAYER;

board[player.position.row + 2][player.position.column + 2] = MT;

player.position.row = row;

player.position.column = column;

}

movePlayer = true;

//.....Process Monster

if (!monster.isFound)

{

// <WK8 status=permanent>

processMonster(monsterPause, cave, monster, MonsterMoveCounter,

player, msg, gameOver, killedValue, killMonsterValue, movePlayer, monsterHold, board);

// </WK8>

}

movePlayer = true;

// <WK7 status=permanent>

//.....Process Thief

if (!thief.isFound)

{

// <WK8 status=permanent>

processThief(thiefPause, cave, thief, thiefMoveCounter, player,

msg, gameOver, killedThiefValue, killThiefValue, movePlayer,

thiefHold, board, monster);

// </WK8>

}

movePlayer = true;

// </WK7>

}

}

//...End Game Loop

//\*\*End Game\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// <WK8 status=permanent>

//...Provide end win/loss message

cout << msg.c\_str() << endl;

if (player.alive)

{

if (player.hasTreasure) msg = "You are alive and rich!\nGAME OVER!\n\nPress enter to exit...";

else msg = "You didn't get the treasure, but you live to seek adventure another day.\nGAME OVER!";

}

else

{

msg = "RIP Adventurer!\nGAME OVER!";

}

cout << msg.c\_str() << endl;

// </WK8>

//...Do clean-up

//...Quit

cin.get();

cin.get();

return 0;

}

/\*/--------------------------------------------------------------------------------

END of MAIN FUNCTION

//================================================================================\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

DEFINITION of FUNCTIONS

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// <WK8 status=permanent>

// print cave

void printCave(gameObjectType cave[TOTAL\_ROWS][TOTAL\_COLS])

{

for (int r = 0; r < MAX\_ROWS; r++)

{

for (int c = 0; c < MAX\_COLS; c++)

{

switch (cave[r][c])

{

case EMPTY:

cout << '-';

break;

case PLAYER:

cout << PSymbol;

break;

case THIEF:

cout << QSymbol;

break;

case TREASURE:

cout << TRSymbol;

break;

case MONSTER:

cout << MSymbol;

break;

case WEAPON:

cout << WSymbol;

break;

case TORCH:

cout << TOSymbol;

break;

case NOISEMAKER:

cout << NSymbol;

break;

case EYE:

cout << YSymbol;

break;

case CAVE\_EXIT:

cout << XSymbol;

break;

}

}

cout << endl;

}

}

// </WK8>

gameObject placeInCave(gameObjectType array[TOTAL\_ROWS][TOTAL\_COLS])

{

int r, c;

gameObject obj;

do

{

r = rand() % 8;

c = rand() % 8;

} while (array[r][c] != EMPTY);

obj.row = r;

obj.column = c;

obj.isFound = false;

return obj;

}

// <WK4 status=permanent>

bool checkVisible(gameObject x, gameObject y, int dist)

{

if ((abs(x.row - y.row) < dist && (abs(x.column - y.column) < dist))) return true;

else return false;

}

// </WK4>

bool showOnBoard(gameObject x)

{

return ((x.isVisible) && (!x.isFound));

}

// <WK5 status=permanent>

//....Process player command

void processPlayerCommand(char command, int& row, int& column, playerObject& player, bool& movePlayer, string& msg, bool& monsterPause, bool& thiefPause, bool& gameOver)

{

row = player.position.row;

column = player.position.column;

// store last 2 player moves

player.twoBackPosition = player.oneBackPosition;

player.oneBackPosition = player.position;

switch (command)

{

// go west

case 'a':

column = player.position.column - 1;

if (column < MIN\_COLS)

{

column = player.position.column;

msg = "You can't walk through walls!";

}

break;

// go south

case 's':

row = player.position.row + 1;

if (row >= MAX\_ROWS)

{

row = player.position.row;

msg = "You can't walk through walls!";

}

break;

// go north

case 'w':

column = player.position.column;//Is this really needed?

row = player.position.row - 1;

if (row < MIN\_ROWS)

{

row = player.position.row;

msg = "You can't walk through walls!";

}

break;

// go east

case 'd':

row = player.position.row;//Is this really needed?

column = player.position.column + 1;

if (column >= MAX\_COLS)

{

column = player.position.column;

msg = "You can't walk through walls!";

}

break;

// <WK4 status=permanent>

// noise

case 'n':

if (player.hasNoisemaker)

{

msg = "You make an irritating noise!";

monsterPause = true;

// <WK7 status=permanent>

thiefPause = true;

// </WK7>

}

else

{

msg = "You make a feeble whimper.";

}

break;

// </WK4>

// quit

case 'x':

gameOver = true;

msg = "Quitting?\nPress enter to continue...";

break;

default:

movePlayer = false;

break;

}

}

// </WK5>

// <WK7 status=permanent>

// check for player events

void checkForEvents(gameObjectType cave[TOTAL\_ROWS][TOTAL\_COLS], int& row, int& column, playerObject& player, string& msg, gameObject& thief, gameObject& treasure, gameObject& weapon, gameObject& monster, gameObject& noisemaker, gameObject& torch, gameObject& eye, gameObject& caveExit, bool& gameOver, bool& movePlayer, bool& monsterPause, bool& thiefPause, int& killedValue, int& killMonsterValue, int& killedThiefValue, int& killThiefValue, char board[TOTAL\_ROWS + 3][TOTAL\_COLS + 3])

{

// Check for events

switch (cave[row][column])

{

// If exit found, set flag to exit game

// <WK6 status=permanent>

case CAVE\_EXIT:

gameOver = true;

msg = "You exit the cave.";

break;

// </WK6>

// If treasure found, set flag to show player has treasure

case TREASURE:

player.hasTreasure = true;

treasure.isFound = true;

msg = "You found the treasure!";

gameOver = true;

break;

// If weapon found, set flag to show player has weapon

case WEAPON:

player.hasWeapon = true;

weapon.isFound = true;

msg = "You found the sword.";

// <WK6 status=permanent>

killedValue = PLAYER\_DIES\_WITH\_WEAPON;

killMonsterValue = MONSTER\_DIES\_WITH\_WEAPON;

// </WK6>

// <WK7 status=permanent>

// a random value above this level means you are killed by the thief

killedThiefValue = PLAYER\_DIES\_TO\_THIEF;

// a random value above this level and below killedValue means you kill thief

killThiefValue = THIEF\_DIES\_TO\_PLAYER;

// </WK7>

break;

// <WK4 status=permanent>

// If noise-maker found, set flag to show player has noise-maker

case NOISEMAKER:

player.hasNoisemaker = true;

noisemaker.isFound = true;

msg = "You found a noisemaker.";

break;

// If torch found, set flag to show player has torch

case TORCH:

torch.isFound = true;

msg = "Let there be light! You found a torch.";

// <WK6 status=permanent>

caveExit.isVisible = true;

// </WK6>

weapon.isVisible = true;

treasure.isVisible = true;

noisemaker.isVisible = true;

monster.isVisible = true;

// <WK8 status=permanent>

// if the torch is found first, it reveals all objects except the eye and thief

if (!player.hasEye)

{

thief.isVisible = false;

eye.isVisible = false;

}

// if the eye is found after the torch, then it reveals the thief

else

{

thief.isVisible = true;

}

player.hasTorch = true;

// </WK8>

break;

// </WK4>

//......If eye found, set flag to show player has eye

// <WK8 status=permanent>

case EYE:

eye.isFound = true;

msg = "You're all seeing! You found the Eye.";

caveExit.isVisible = true;

weapon.isVisible = true;

treasure.isVisible = true;

noisemaker.isVisible = true;

monster.isVisible = true;

eye.isVisible = true;

// if the eye is found first, it reveals all objects on board including thief

if(!player.hasEye)

thief.isVisible = true;

player.hasEye = true;

break;

// </WK8>

// If monster found

case MONSTER:

if (!monster.isFound)

{

msg = "You have found the monster";

// Resolve combat

// <WK6 status=permanent>

gameOver = resolveCombat(player, monster, killedValue, killMonsterValue,

msg, movePlayer, monsterPause, cave, row, column, board);

// </WK6>

}

break;

// <WK8 status=permanent>

// If thief found

case THIEF:

if (!thief.isFound)

{

msg = "You have found the thief";

// Resolve combat

//gameOver = false;

gameOver = resolvethiefCombat(player, thief, killedThiefValue, killThiefValue, msg, movePlayer, monsterPause, cave, row, column, board);

}

break;

// </WK8>

}// moved else clause WK4

}

// </WK7>

// <WK8 status=permanent>

// process monster

void processMonster(bool monsterPause, gameObjectType cave[TOTAL\_ROWS][TOTAL\_COLS], gameObject& monster, int& MonsterMoveCounter, playerObject& player, string& msg, bool& gameOver, int& killedValue, int& killMonsterValue, bool& movePlayer, gameObjectType& monsterHold, char board[TOTAL\_ROWS + 3][TOTAL\_COLS + 3])

{

int row = 0, column = 0;

// <WK4 status=permanent>

if (monsterPause)

{

// Monster paused. Check if Monster starts moving again

if (rand() %3 == 1)

{

monsterPause = false;

}

}

else

{

// </WK4>

// Move Monster

MonsterMoveCounter = (++MonsterMoveCounter) %8;

row = monster.row;

column = monster.column;

switch (MonsterMoveCounter)

{

case 0:

column++;

break;

case 1:

row--;

break;

case 2:

row--;

break;

case 3:

column--;

break;

case 4:

column--;

break;

case 5:

row++;

break;

case 6:

row++;

break;

case 7:

column++;

break;

default:

break;

}

//......Check for events

//.......If player found

if (cave[row][column] == PLAYER)

{

msg = "The monster has found you";

//........Resolve combat

// <WK6 status=permanent>

gameOver = resolveCombat(player, monster, killedValue-2, killMonsterValue-2, msg, movePlayer, monsterPause, cave, row, column, board);

// </WK6>

// <WK3 status=5>

if (player.hasWeapon)

{

monster.isFound = true;

msg = "The monster found you but was slain.";

}

else

{

// <WK2 status=5>

gameOver = true;

player.alive = false;

msg = "The monster found you and you have died.";

}

// </WK2>

}

else

{

//........Move Monster

// <WK8 status=permanent>

// reveal what is under the monster <bug fix in week 6>

cave[monster.row][monster.column] = monsterHold;

// </WK6>

// clear the cave location <creates bug>

//cave[monster.row][monster.column] = EMPTY;

// clear the screen behind monster

board[monster.row + 2][monster.column + 2] = MT;

// <WK6 status=permanent>

// save what the monster is about to move over <bug fix in week 6>

monsterHold = cave[row][column];

// </WK8>

// update monster's row

monster.row = row;

// update monster's column

monster.column = column;

// change monster's location in the cave

cave[row][column] = MONSTER;

}

}

}

// </WK8>

// <WK6 status=permanent>

bool resolveCombat(playerObject &p, gameObject &m, int youDie, int monsterDie, string &str, bool &moveP, bool &mPause, gameObjectType cave[TOTAL\_ROWS][TOTAL\_COLS], int& row, int& column, char board[TOTAL\_ROWS + 3][TOTAL\_COLS + 3])

{

int combatRoll = rand()%75;

// <WK8 status=permanent>

// monster kills player

if (combatRoll > youDie)

{

str += " and you have been eaten by a Grue!";

moveP = false;

p.position.isVisible = false;

p.alive = false;

return true;

}

// </WK8>

// <WK8 status=permanent>

// player kills monster

if (combatRoll > monsterDie)

{

if (p.hasWeapon) str += " and you slay the monster with your weapon.";

else str += " but--against all odds--\nyou actually killed the monster with your bare hands.";

m.isFound = true;

return false;

}

// </WK8>

// Survive Attack

else

{

str += " and you have no weapon.\nYou managed to survive the monster's attack by

regrouping.";

moveP = true;

mPause = false;

// <WK6 status=permanent>

// when player has no weapon and monster is found

// player will move back two spaces

// interacting message will display

p.position = p.oneBackPosition;

//updates position information

cave[p.position.row][p.position.column] = EMPTY;

cave[p.position.row][p.position.column] = PLAYER;

//clear the screen where player was

board[p.position.row + 2][p.position.column + 2] = MT;

// set up to move back another space

row = p.twoBackPosition.row;

column = p.twoBackPosition.column;

return false;

// </WK6>

} // updated WK6 to else clause from WK4

}

// </WK6>

// <WK8 status=permanent>

// process thief

void processThief(bool thiefPause, gameObjectType cave[TOTAL\_ROWS][TOTAL\_COLS], gameObject& thief, int& thiefMoveCounter, playerObject& player, string& msg, bool& gameOver, int& killedThiefValue, int& killThiefValue, bool& movePlayer, gameObjectType& thiefHold,

char board[TOTAL\_ROWS + 3][TOTAL\_COLS + 3], gameObject& monster )

{

int row = 0, column = 0;

if (thiefPause)

{

// thief paused. check if thief starts moving again

if (rand() %3 == 1)

{

thiefPause = false;

}

}

else

{

// move thief

thiefMoveCounter++;

row = thief.row;

column = thief.column;

// thief makes random move

switch (rand() % 4)

{

// go east

case 0:

column++;

// check for wall

if (column >= MAX\_COLS)

column-=2;

break;

// go north

case 1:

row--;

// check for wall

if (row < 0)

row+=2;

break;

// go south

case 2:

row++;

// check for wall

if (row >= MAX\_ROWS)

row -= 2;

break;

// go west

case 3:

column--;

// check for wall

if (column < 0)

column+=2;

break;

}

// check for events

// if player found

if (cave[row][column] == PLAYER)

{

msg = "The thief found you";

//...resolve combat

gameOver = resolvethiefCombat(player, thief, killedThiefValue-2,

killThiefValue-2, msg, movePlayer, thiefPause, cave, row, column, board);

}

// thief finds monster

else if (cave[row][column] == MONSTER)

{

msg = "The thief has found the monster";

//..resolve combat

resolvethiefMonterCombat(monster, thief, killedThiefValue - 2,

killThiefValue - 2, msg, movePlayer, thiefPause, cave, row, column, board);

}

// thief finds treasure

else if (cave[row][column] == TREASURE)

{

// game over

msg = "The thief has found the treasure!";

gameOver = true;

}

else

{

// move thief

// reveal what is under the thief

cave[thief.row][thief.column] = thiefHold;

// clear the cave location <creates bug>

cave[thief.row][thief.column] = EMPTY;

// clear the screen behind thief

board[thief.row+2][thief.column+2] = MT;

// save what the thief is about to move over

thiefHold = cave[row][column];

// update thief's row

thief.row = row;

// update thief's column

thief.column = column;

// change thief's location in the cave

cave[row][column] = THIEF;

}

}

}

// </WK8>

// <WK8 status=permanent>

// resolve thief player combat

bool resolvethiefCombat(playerObject &p, gameObject &q, int playerDie, int thiefDie, string &str, bool &moveP, bool &qPause, gameObjectType cave[TOTAL\_ROWS][TOTAL\_COLS], int& row, int& column, char board[TOTAL\_ROWS + 3][TOTAL\_COLS + 3])

{

int combatRoll = rand()%50;

// player kills thief

if (p.hasWeapon)

{

p.alive = true;

q.isFound = true;

str += " and you slay the thief with your weapon.";

return false;

}

// thief kills player

else if (combatRoll > playerDie)

{

str += " and you have been strangled!";

moveP = false;

p.position.isVisible = false;

p.alive = false;

return true;

}

// player kills thief

else if (combatRoll > thiefDie)

{

if (p.hasWeapon) str += " and you slay the thief with your weapon.";

q.isFound = true;

return false;

}

else str += " and the thief gets away.";

return false;

}

// </WK8>

// <WK8 status=permanent>

// resolve thief monster combat

bool resolvethiefMonterCombat(gameObject &monster, gameObject &q, int youDie, int thiefDie, string &str, bool &moveP, bool &qPause, gameObjectType cave[TOTAL\_ROWS][TOTAL\_COLS], int& row, int& column, char board[TOTAL\_ROWS + 3][TOTAL\_COLS + 3])

{

int combatRoll = rand()% 50;

// thief kills monster

if (combatRoll > MONSTER\_DIES\_NO\_WEAPON)

{

str += " and the thief kills the monster.";

moveP = false;

monster.isVisible = false;

monster.isAlive = false;

return true;

}

// monster kills thief

else if (combatRoll > thiefDie)

{

str += " and the monster kills the thief.";

q.isVisible = false;

//q.isAlive = false;

return false;

}

else return false;

}

// </WK8>

// <WK6 status=permanent>

void gamecolor(unsigned char type)

{

switch(type)

{

case 201: case 205: case 200: case 188: case 186: case 187:

setcolor(FOREGROUND\_RED);

break;

case 'W':

setcolor(FOREGROUND\_RED|FOREGROUND\_GREEN|FOREGROUND\_INTENSITY);

break;

case '$':

setcolor(FOREGROUND\_GREEN|FOREGROUND\_INTENSITY);

break;

case 'P':

setcolor(FOREGROUND\_BLUE|FOREGROUND\_INTENSITY|BACKGROUND\_RED|BACKGROUND\_GREEN|BA

CKGROUND\_INTENSITY);

break;

case 'T':

setcolor(FOREGROUND\_RED|FOREGROUND\_INTENSITY);

break;

case 'X':

setcolor(BACKGROUND\_BLUE|BACKGROUND\_GREEN);

break;

case 'M':

setcolor(FOREGROUND\_RED|FOREGROUND\_BLUE|FOREGROUND\_INTENSITY);

break;

case 'N':

setcolor(FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_INTENSITY);

break;

// <WK7 status=permanent>

case 'Q':

setcolor(FOREGROUND\_RED|FOREGROUND\_BLUE|FOREGROUND\_INTENSITY);

break;

case 'Y':

setcolor(FOREGROUND\_RED|FOREGROUND\_GREEN|FOREGROUND\_INTENSITY);

break;

// </WK7>

default:

setcolor(FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

}

}

// </WK6>

// <WK6 status=permanent>

void setcolor(int c)

{

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), c);

}

// </WK6

***courseProject\_Final.h***

//GSP115\_Course\_Project.h

#pragma once

#include <iostream>

#include <conio.h>

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#include <windows.h>

// <WK7 status=permanent>

enum gameObjectType

{

EMPTY, PLAYER, THIEF, TREASURE, MONSTER, WEAPON, TORCH, NOISEMAKER, EYE, CAVE\_EXIT // add in week 6

};

// <WK7>

struct gameObject

{

int row; // row position of the object

int column; // column position of the object

bool isFound; // flag to indicate if the object has been found

bool isVisible; // flag to indicate if object is seen on the board

// <WK7 status=permanent>

bool isAlive; // flag to indicate if thief and monster is alive or dead

// <WK7>

};

struct playerObject

{

bool alive; // flag to indicate if the player is alive or dead

bool hasWeapon; // flag to indicate if the player has the weapon

bool hasTreasure; // flag to indicate if the player has the treasure

bool hasTorch; // flag to indicate if the player has the torch -week 4

bool hasNoisemaker; // flag to indicate if the player has the noisemaker -week 4

// <WK7 status=permanent>

bool hasEye; // flag to indicate if the player has the eye

// <WK7>

gameObject position; // variables for row, column and visibility

// <WK4 status=permanent>

gameObject oneBackPosition; // variables for row, column from the last movement

gameObject twoBackPosition; // variables for row, column from two last movements

// <WK4>

};