```
//Denzel P, Dariusz D
//gooseEscapeGamePlay.cpp

#include <iostream>
#include <cmath>
using namespace std;
#include <BearLibTerminal.h>
#include "gooseEscapeUtil.hpp"
#include "gooseEscapeActor.hpp"
#include "gooseEscapeConsole.hpp"
#include "gooseEscapeGamePlay.hpp"

/*
    This file is all about the game world.  You will modify this to add
    functionality to your game, first to get it working, and later to make
    it fun.

    If you add or modify functions, be sure to update the prototypes in the
    gooseEscapeGamePlay.hpp file.
*/

extern Console out;

/*
With graphics, screens are given an x,y coordinate system with the origin
in the upper left corner.  So it means the coordinate axes are:
---------------->   x direction
|
|
|
|
|
V

y direction
*/

/*
        Print the game world

        This function should draw characters to represent features of the game
        board, e.g. win location, obstacles, power ups, etc.
*/
void randomSetup(int gameBoard[NUM_SCREEN_Y][NUM_SCREEN_X], int & xSafe, int & ySafe)
{
        //Randomize Wall initial location
        int xLocation = rand() % 58 + 11;
        int yLocation = rand() % 13 + 1;

        //What If the wall/safe spot same as goose/player
```

```
        //Randmoize Safe Location
        xSafe = rand()% 58 + 11;
        ySafe = rand()% 8 + 11;

        //If location of wall is same as safe location move safe location
        if (xLocation == xSafe)
        {
                xSafe += 1;
        }
        else if (yLocation == ySafe)
        {
                ySafe += 1;
        }

        //Wall Length 10
        for(int i = 0; i<10; i++)
        {
                gameBoard[yLocation+i][xLocation] = SHALL_NOT_PASS;
        }

        //Landmine Number: 20
        for(int i = 0; i<20; i++)
        {
                //landmine
                gameBoard[rand()% 15 + 5][rand()% 58 + 11] = LANDMINE;
        }

        //PowerUp tile randomized
        gameBoard[rand()% 15 + 5][rand()% 58 + 11] = POWER;
        //Safe tile randomzied
        gameBoard[ySafe][xSafe] = WINNER;
}

void printWorld(int gameBoard[NUM_SCREEN_Y][NUM_SCREEN_X])
{
        //Print board
    for (int row = 0; row < NUM_SCREEN_Y; row++)
    {
        for (int col = 0; col < NUM_SCREEN_X; col++)
        {
                if (gameBoard[row][col] == 0)
                        terminal_put(col, row, int(' '));

                else if (gameBoard[row][col] == 1)
                        terminal_put(col, row, WALL_CHAR);

                else if (gameBoard[row][col] == 2)
                        terminal_put(col, row, WIN_CHAR);

                else if (gameBoard[row][col] == 3)
```

```
                        terminal_put(col, row, LANDMINE_CHAR);

                        else if (gameBoard[row][col] == 4)
                        terminal_put(col, row, POWER_CHAR);

                }
        }
        terminal_refresh();
}

/*
    Do something when the goose captures the player

    At the moment the function just checks to see if the player and the goose
    are in the same location.  If you want to attack or do something else, this
    function would need to change.  For example, maybe the two touch each other
    and then fight.  You could use the health that is given in the Actor class,
    and update it.  Fight, run, use weapons, it is up to you!
*/

bool captured(Actor const & player, Actor const & monster)
{
        //check if player is captured
    return (player.get_x() == monster.get_x()
         && player.get_y() == monster.get_y());
}

//check if player has won
bool safe(Actor const & player, int ySafe, int xSafe)
{
        return player.get_x() == xSafe
         && player.get_y() == ySafe;
}

//check if player is dead
bool isDead(Actor & player)
{
        return player.get_health_int() == 0;
}

/*
    Move the player to a new location based on the user input

    All key presses start with "TK_" then the character.  So "TK_A" is the A
    key that was pressed.  At the moment, only the arrow keys are used,
    but feel free to add more keys that allow the player to do something else
    like pick up a power up.

    A look-up table might be useful.
```

```
    Going further: You could decide to learn about switch statements
*/

//check move based
int whichMove(int key, Actor & player, int gameBoard[NUM_SCREEN_Y][NUM_SCREEN_X], int
powerCount,
        int addBackMines[NUM_SCREEN_Y][NUM_SCREEN_X])
{
        if (checkPower(player,gameBoard) == true || powerCount > 1)
        {
                if (checkPower(player,gameBoard) == true)
                {
                        gameBoard[player.get_y()][player.get_x()] = 0;
                        powerCount = 10;//Starts counter based on power up
                        powerUp(powerCount, gameBoard, addBackMines);
                        }
                movePlayer(key,player,gameBoard);
                powerCount--;
                }
                else if (powerCount == 1)//call power up to reprint world
                {
                        powerCount--;
                        movePlayer(key,player,gameBoard);
                        powerUp(powerCount, gameBoard, addBackMines);
                }
                else//normal move
                {
                        movePlayer(key,player,gameBoard);
                }
                return powerCount;
}

//remove health if player hits a landmine
void injure(Actor & player, int gameBoard[NUM_SCREEN_Y][NUM_SCREEN_X])
{
        if(gameBoard[player.get_y()][player.get_x()] == LANDMINE)
        {
                player.update_health(-25);
                out.writeLine(player.get_health_message());

                gameBoard[player.get_y()][player.get_x()] = 0;
        }
}

//check if player is on powerup
bool checkPower(Actor & player, int gameBoard[NUM_SCREEN_Y][NUM_SCREEN_X])
{
        return gameBoard[player.get_y()][player.get_x()] == POWER;
}
```

```
//move player generally
void movePlayer(int key, Actor & player, int gameBoard[NUM_SCREEN_Y][NUM_SCREEN_X])
{
    int yMove = 0, xMove = 0;
    if (key == TK_UP)
        yMove = -1;
    else if (key == TK_DOWN)
        yMove = 1;
    else if (key == TK_LEFT)
        xMove = -1;
    else if (key == TK_RIGHT)
        xMove = 1;

    if (player.can_move(xMove, yMove)
      && gameBoard[player.get_y() + yMove][player.get_x() + xMove]!= SHALL_NOT_PASS)
         {
               player.update_location(xMove, yMove);
         }
}

//powerup to print or remove landmines based on counter
void powerUp(int powerCount, int gameBoard[NUM_SCREEN_Y][NUM_SCREEN_X],
        int addBackMines[NUM_SCREEN_Y][NUM_SCREEN_X])
{
        if (powerCount == 10)
        {
                for (int row = 0; row < NUM_SCREEN_Y; row++)
            {
                for (int col = 0; col < NUM_SCREEN_X; col++)
                {
                        if (gameBoard[row][col] == LANDMINE)
                        {
                                gameBoard[row][col] = 0;
                                        addBackMines[row][col] = LANDMINE;
                                }
                }
                }
        }
        else if (powerCount == 0)
        {
                for (int row = 0; row < NUM_SCREEN_Y; row++)
            {
                for (int col = 0; col < NUM_SCREEN_X; col++)
                {
                        if (addBackMines[row][col] == LANDMINE)
                        {
                                gameBoard[row][col] = LANDMINE;
                                }
                }
                }
```

```
        }
        printWorld(gameBoard);
        out.writeLine("Escape the Goose! o:Wall,#:Landmine,F:Power-up for 10 moves");
        out.writeLine("Use the arrow keys to move");
        out.writeLine("If the goose catches you, you lose! Don't hit the landmines!");
        out.writeLine("Remember! Both you and the goose can't pass through walls!");
}

//Goose moves 1 space based on shortest distance to player
void moveGoose(Actor & player, Actor & monster, int gameBoard[NUM_SCREEN_Y][NUM_SCREEN_X])
{
        int yMove = 0, xMove = 0;

        int left = monster.get_x()-1;
        int right = monster.get_x()+1;
        int up = monster.get_y()-1;
        int down = monster.get_y()+1;

        int playerX = player.get_x();
        int playerY = player.get_y();


        double leftDistance = sqrt(pow((left-playerX),2)+pow((monster.get_y()-playerY),2));
        double rightDistance = sqrt(pow((right-playerX),2)+pow((monster.get_y()-playerY),2));
        double upDistance = sqrt(pow((monster.get_x()-playerX),2)+pow((up-playerY),2));
        double downDistance = sqrt(pow((monster.get_x()-playerX),2)+pow((down-playerY),2));

        double distances[4] = {leftDistance, rightDistance, upDistance, downDistance};

        double shortestDistance = 10000000;

        for(int i = 0; i<4; i++)
        {
                if(distances[i]<shortestDistance)
                {
                        shortestDistance = distances[i];
                }
        }

        if(leftDistance == shortestDistance && rightDistance == shortestDistance
        && upDistance == shortestDistance && downDistance == shortestDistance)
        {
                xMove = 0;
                yMove = 0;
        }

        else if(leftDistance == shortestDistance)
                xMove=-1;
        else if(rightDistance == shortestDistance)
                xMove=1;
```

```
        else if(upDistance == shortestDistance)
                yMove=-1;
        else if(downDistance == shortestDistance)
                yMove=1;

        if (monster.can_move(xMove, yMove)
                && gameBoard[monster.get_y() + yMove][monster.get_x() + xMove]!= SHALL_NOT_PASS
                && gameBoard[monster.get_y() + yMove][monster.get_x() + xMove]!= WINNER)
                {
                        gameBoard[monster.get_y() + yMove][monster.get_x() + xMove]=0;
                        monster.update_location(xMove, yMove);
                }

}

/*
    What other functions do you need to make the game work?  What can you add
    to the basic functionality to make it more fun to play?
*/
```