Національний технічний університет України
“Київський політехнічний інститут ім. Ігоря Сікорського”

Факультет прикладної математики
Кафедра системного програмування і спеціалізованих комп’ютерних систем

РОЗРАХУНКОВА РОБОТА

з дисципліни

“Програмування”

ТЕМА: “ІГРОВА ПРОГРАМА”

Група: КВ-34

Виконала: Фіалковська Ольга

Варіант 21

Оцінка:

Київ – 2024

Завдання

Створити ігрову програму мовою програмування С.

Розробка і реалізація ігрових програм має вестися з врахуванням графічних та звукових можливостей, що надаються конкретним комп'ютером.

Програма мусить коректно розв'язувати поставлену задачу. Логічно відокремлені частини алгоритма реалізувати за допомогою окремих функцій.

Також потрібно передбачити та забезпечити виконання всіх можливих розгалужень алгоритма, тобто програма повинна коректно реагувати на будь-які можливі ситуації (наприклад, виникнення помилкових ситуацій, перевірка файлів на порожність, правильність введених з клавіатури значень і т. д.). Передбачити взаємодію з користувачем (наприклад, можливість виводу правил гри, допомоги), таймер, лічильник числа ходів відповідно до поставленої в конкретному варіанті задачі.

Структура звіту для РГР

1. Титульний аркуш.
2. Технічне завдання: що потрібно зробити, конкретний варіант.
3. Описання структури програми, тобто ілюстрація взаємозв'язків підпрограм в програмі. Описати, що виконується в кожному блоці.
4. Текст програми з коментарями.
5. Роздруковані графічні результати.
6. Всі файли проекта надіслати на easilly7@gmail.com до 30/05/2024 р.

Варіант 21

«Полювання на лисиць». На полі випадковим чином розставляються вісім «лисиць», деякі з них можуть знаходитись в одній клітинці. Гравець вводить свої координати. У відповідь він отримує кількість «лисиць», які пеленгуються з його місця (кількість «лисиць», що розташовані з гравцем в одній вертикалі, горизонталі і діагоналі). Якщо розташування гравця співпало з «лисицею», то вона вважається знайденою. Гра ведеться, поки не будуть знайдені всі лисиці.

Код:

```
// Включение заголовочных файлов
#include <SFML/Graphics.h>
#include <SFML/Audio.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

// Определение констант
#define ELEMENT_SIZE 80
#define ELEMENTS_PER_ROW 10
#define ELEMENTS_PER_COLUMN 5
#define highfield ELEMENTS_PER_COLUMN
#define widthfield ELEMENTS_PER_ROW

// Меню игры
const char* menuItems[] = {"Start Game", "Rules", "Exit"};
const int menuItemCount = 3;
int selectedItem = 0;
```

```
// Счетчики и массивы для игрового поля
int counter_1 = 0, counter_2 = 0, counter_3 = 0;
int field[highfield][widthfield];
int selected_picture_index = -1;
int numfoxes = 0, findfoxes = 8;
int showPopup = 0;
int allFoxesFound = 0;

// Структура мини-изображения
typedef struct
{
    sfVector2f position;
    sfTexture* texture;
    sfSprite* sprite;
    int row;
    int col;
    int clicked;
} MiniPicture;

// Функция для отрисовки меню
void drawMenu(sfRenderWindow* window, sfFont* font)
{
    sfText* text = sfText_create();
    sfText_setFont(text, font);
    sfText_setCharacterSize(text, 50);
    sfVector2f position = {100, 100};

    for (int i = 0; i < menuItemCount; ++i)
    {
```

```
        sfText_setString(text, menuItems[i]);
        sfText_setPosition(text, position);

        if (i == selectedItem)
        {
            sfText_setColor(text, sfColor_fromRGB(0, 225, 0));
        }
        else
        {
            sfText_setColor(text, sfColor_fromRGB(255, 255, 255));
        }

        sfRenderWindow_drawText(window, text, NULL);
        position.y += 60;
    }

    sfText_destroy(text);
}

// Функция для отрисовки правил игры
void drawRules(sfRenderWindow* window, sfFont* font)
{
    sfText* text = sfText_create();
    sfText_setFont(text, font);
    sfText_setCharacterSize(text, 30);
    sfText_setColor(text, sfColor_fromRGB(255, 255, 255));

    const char* rules =
        "Rules:\n"
```

```
        "1. Click on the grass to find foxes.\n"
        "2. The number of foxes in the row, column, and diagonals\n"
        "   will be displayed when you find a fox.\n"
        "3. Find all 8 foxes to win the game.\n\n"
        "Press any key to return to the menu.";

    sfText_setString(text, rules);
    sfVector2f position = {50, 50};
    sfText_setPosition(text, position);
    sfRenderWindow_drawText(window, text, NULL);

    sfText_destroy(text);
}

// Функция для обработки координат и подсчета лис
void input_coordinates(int h, int w, sfText* text, sfFont* font)
{
    numfoxes = field[h][w];
    selected_picture_index = numfoxes;
    switch (field[h][w])
    {
    case 0:
        printf("fox not found\n");
        count_foxes(h, w, text, font);
        break;
    case 1:
        printf("fox found\n");
        count_foxes(h, w, text, font);
        break;
```

```
    default:
        printf("%d foxes found\n", field[h][w]);
        count_foxes(h, w, text, font);
        break;
    }
}

// Функция для подсчета лис
void count_foxes(int h, int w, sfText* text, sfFont* font)
{
    counter_1 = 0;
    counter_2 = 0;
    counter_3 = 0;

    for (int i = 0; i < highfield; i++)
    {
        if (field[i][w] != 0 && i != h)
            counter_1 += field[i][w];
    }
    for (int j = 0; j < widthfield; j++)
    {
        if (field[h][j] != 0 && j != w)
            counter_2 += field[h][j];
    }
    for (int i = h - 1, j = w - 1; i >= 0 && j >= 0; i--, j--)
        if (field[i][j] != 0)
            counter_3 += field[i][j];

    for (int i = h + 1, j = w + 1; i < highfield && j < widthfield; i++, j++)
```

```
        if (field[i][j] != 0)
            counter_3 += field[i][j];

    for (int i = h - 1, j = w + 1; i >= 0 && j < widthfield; i--, j++)
        if (field[i][j] != 0)
            counter_3 += field[i][j];

    for (int i = h + 1, j = w - 1; i < highfield && j >= 0; i++, j--)
        if (field[i][j] != 0)
            counter_3 += field[i][j];

    pop_up_window_text(text, font);
}

// Функция для создания мини-изображений
void minipictures(sfRenderTexture* texture, MiniPicture pictures[])
{
    for (int i = 0; i < ELEMENTS_PER_ROW * ELEMENTS_PER_COLUMN;
i++)
    {
                                                    pictures[i].texture         =
sfTexture_createFromFile("png-pikselnaya-trava60x60.png", NULL);
        if (!pictures[i].texture)
        {
            printf("Error loading picture texture\n");
            exit(1);
        }
        pictures[i].sprite = sfSprite_create();
        sfSprite_setTexture(pictures[i].sprite, pictures[i].texture, sfTrue);
```

```
        pictures[i].clicked = 0;
    }

    int index = 0;
    for (int row = 0; row < ELEMENTS_PER_COLUMN; row++)
    {
        for (int col = 0; col < ELEMENTS_PER_ROW; col++)
        {
            pictures[index].position.x = col * ELEMENT_SIZE + 12;
            pictures[index].position.y = row * ELEMENT_SIZE + 12;
            pictures[index].row = row;
            pictures[index].col = col;
            index++;
        }
    }

    for (int i = 0; i < ELEMENTS_PER_ROW * ELEMENTS_PER_COLUMN;
i++)
    {
        sfSprite_setPosition(pictures[i].sprite, pictures[i].position);
        sfRenderTexture_drawSprite(texture, pictures[i].sprite, NULL);
    }
}

// Функция для отображения окна с информацией о лисах
void pop_up_window_text(sfText* text, sfFont* font)
{
    findfoxes -= numfoxes;
    char buffer[200];
```

```
            sprintf(buffer, "Fox hunting\n\n%d foxes were not found\nvertical:\t%d\nhorizontal:\t%d\ndiagonal:\t%d\n", findfoxes, counter_1, counter_2, counter_3);
    sfText_setString(text, buffer);
    sfText_setFont(text, font);
    sfText_setCharacterSize(text, 18);
    sfText_setColor(text, sfColor_fromRGB(255, 255, 255));
    sfVector2f textPosition = {365, 100};
    sfText_setPosition(text, textPosition);
}

// Функция для создания всплывающего окна
void pop_up_window(sfRectangleShape* popup, sfRectangleShape* closeButton, sfText* text, sfFont* font)
{
    sfVector2f popupSize = {500, 300};
    sfVector2f popupPosition = {200, 50};
    sfRectangleShape_setSize(popup, popupSize);
    sfRectangleShape_setPosition(popup, popupPosition);
    sfTexture* popupTexture = sfTexture_createFromFile("back.png", NULL);
    if (!popupTexture)
    {
        printf("Error loading popup texture\n");
        return;
    }
    sfRectangleShape_setTexture(popup, popupTexture, sfTrue);

    pop_up_window_text(text, font);
```

```
    sfVector2f buttonSize = {30, 30};
    sfVector2f buttonPosition = {640, 75};
    sfRectangleShape_setSize(closeButton, buttonSize);
    sfRectangleShape_setPosition(closeButton, buttonPosition);
        sfRectangleShape_setFillColor(closeButton,  sfColor_fromRGB(128,  128,
128));
}

// Функция для генерации лис на поле
void generation_fox()
{
    srand(time(NULL));
    int eight = 0;
    for (int i = 0; i < highfield; i++)
        for (int j = 0; j < widthfield; j++)
            field[i][j] = 0;
    while (eight < 8)
    {
        field[rand() % highfield][rand() % widthfield] += 1;
        eight++;
    }
    /*for (int i = 0; i < highfield; i++)
    {
        for (int j = 0; j < widthfield; j++)
            printf("%d\t", field[i][j]);
        printf("\n");
    }*/
}
```

```
int main()
{
   generation_fox();
   sfVideoMode mode = {800, 400, 32};
        sfRenderWindow* window = sfRenderWindow_create(mode, "Game",
sfResize | sfClose, NULL);
   if (!window)
      return 1;

   sfEvent event;
      sfTexture* formula_texture = sfTexture_createFromFile("grass_pic.png",
NULL);
   if (!formula_texture)
   {
      printf("Error loading texture\n");
      return 1;
   }

   sfSprite* form_sp = sfSprite_create();
   sfSprite_setTexture(form_sp, formula_texture, sfTrue);

   sfRenderTexture* texture = sfRenderTexture_create(800, 400, sfFalse);
   if (!texture)
   {
      printf("Error creating render texture\n");
      return 1;
   }

   sfFont* font = sfFont_createFromFile("calibri.ttf");
```

```
    if (!font)
    {
        printf("Error loading font\n");
        return 1;
    }

    sfText* text = sfText_create();

    sfRectangleShape* popup = sfRectangleShape_create();
    sfRectangleShape* closeButton = sfRectangleShape_create();
    pop_up_window(popup, closeButton, text, font);
    sfVector2f form_pos = {0, 0};
    sfSprite_setPosition(form_sp, form_pos);
    sfRenderTexture_drawSprite(texture, form_sp, NULL);

    sfMusic* music = sfMusic_createFromFile("plya.wav");
    if (!music)
    {
        printf("Error loading music file\n");
        return 1;
    }
    sfMusic_play(music);

                        MiniPicture        pictures[ELEMENTS_PER_ROW        *
ELEMENTS_PER_COLUMN];
    minipictures(texture, pictures);

    enum { MENU, GAME, RULES, GAME_OVER } state = MENU;
```

```
    while (sfRenderWindow_isOpen(window))
    {
        while (sfRenderWindow_pollEvent(window, &event))
        {
            if (event.type == sfEvtClosed)
                sfRenderWindow_close(window);

            if (state == MENU)
            {
                if (event.type == sfEvtKeyPressed)
                {
                    if (event.key.code == sfKeyUp)
                    {
                                selectedItem = (selectedItem - 1 + menuItemCount) %
menuItemCount;
                    }
                    else if (event.key.code == sfKeyDown)
                    {
                        selectedItem = (selectedItem + 1) % menuItemCount;
                    }
                    else if (event.key.code == sfKeyEnter)
                    {
                        if (selectedItem == 0)
                        {
                            state = GAME;
                        }
                        else if (selectedItem == 1)
                        {
                            state = RULES;
```

```
                }
                else if (selectedItem == 2)
                {
                    sfRenderWindow_close(window);
                }
            }
        }
    }
    else if (state == GAME)
    {
        if (event.type == sfEvtMouseButtonPressed)
        {
            if (event.mouseButton.button == sfMouseLeft)
            {
                if (findfoxes != 0)
                {
                                                    sfVector2i mouse =
sfMouse_getPositionRenderWindow(window);
                                                sfFloatRect closeButtonBounds =
sfRectangleShape_getGlobalBounds(closeButton);
                        if ( sfFloatRect_contains(&closeButtonBounds, mouse.x,
mouse.y))
                    {
                        showPopup = 0;
                    }
                    else
                    {
                                    for (int i = 0; i < ELEMENTS_PER_ROW *
ELEMENTS_PER_COLUMN; i++)
```

```
                    {
                        sfVector2f picture_pos = pictures[i].position;
                                if (mouse.x >= picture_pos.x && mouse.x <=
picture_pos.x + ELEMENT_SIZE && mouse.y >= picture_pos.y && mouse.y
<= picture_pos.y + ELEMENT_SIZE)
                        {
                            if (pictures[i].clicked == 0)
                            {
                                pictures[i].clicked = 1;
                                if (field[pictures[i].row][pictures[i].col] != 0)
                                {
                                    sfTexture_destroy(pictures[i].texture);
                                                        pictures[i].texture =
sfTexture_createFromFile("fox60x60.png", NULL);
                                                    sfSprite_setTexture(pictures[i].sprite,
pictures[i].texture, sfTrue);

                                    if (field[pictures[i].row][pictures[i].col] > 1)
                                    {
                                                        for (int k = 1; k <
field[pictures[i].row][pictures[i].col]; k++)
                                        {
                                            sfSprite* extraFoxSprite = sfSprite_create();
                                                sfSprite_setTexture(extraFoxSprite,
pictures[i].texture, sfTrue);
                                            sfVector2f extraPosition = {picture_pos.x +
(k+15)*10, picture_pos.y + (k+15)*10};
                                                    sfSprite_setPosition(extraFoxSprite,
extraPosition);
```

```
                                        sfRenderTexture_drawSprite(texture,
extraFoxSprite, NULL);
                                            sfSprite_destroy(extraFoxSprite);
                                        }
                                    }
                                     input_coordinates(pictures[i].row, pictures[i].col,
text, font);
                                    showPopup = 1;
                                    break;
                                }
                                else
                                {
                                    sfTexture_destroy(pictures[i].texture);
                                                        pictures[i].texture =
sfTexture_createFromFile("mown_grass60x60.png", NULL);
                                                sfSprite_setTexture(pictures[i].sprite,
pictures[i].texture, sfTrue);
                                     input_coordinates(pictures[i].row, pictures[i].col,
text, font);
                                    showPopup = 1;
                                    break;
                                }
                            }
                        }
                    }
                }
                else if (findfoxes == 0)
                {
```

```
                allFoxesFound = 1;
                showPopup = 0;
                state = GAME_OVER;
            }
        }
    }
}
else if (state == RULES)
{
    if (event.type == sfEvtKeyPressed)
    {
        state = MENU;
    }
}
else if (state == GAME_OVER)
{
    if (event.type == sfEvtKeyPressed)
    {
        state = MENU;
    }
}
}

sfRenderWindow_clear(window, sfBlack);

if (state == MENU)
{
    drawMenu(window, font);
}
```

```
    else if (state == GAME)
    {
        sfRenderTexture_clear(texture, sfBlack);

        if (allFoxesFound)
        {
                                                sfTexture*  endScreenTexture  =
sfTexture_createFromFile("game_over_winner.jpg", NULL);
            if (!endScreenTexture)
            {
                printf("Error loading end screen texture\n");
                return 1;
            }
            sfSprite* endScreenSprite = sfSprite_create();
            sfSprite_setTexture(endScreenSprite, endScreenTexture, sfTrue);
            sfVector2f endScreenPosition = {0, 0};
            sfSprite_setPosition(endScreenSprite, endScreenPosition);
            sfRenderTexture_drawSprite(texture, endScreenSprite, NULL);
            sfSprite_destroy(endScreenSprite);
            sfTexture_destroy(endScreenTexture);
        }
        else
        {
            sfSprite_setPosition(form_sp, form_pos);
            sfRenderTexture_drawSprite(texture, form_sp, NULL);
                            for (int i = 0; i < ELEMENTS_PER_ROW *
ELEMENTS_PER_COLUMN; i++)
            {
                sfRenderTexture_drawSprite(texture, pictures[i].sprite, NULL);
```

```
            }
        }

        sfRenderTexture_display(texture);

        sfSprite* sp = sfSprite_create();
        sfSprite_setTexture(sp, sfRenderTexture_getTexture(texture), sfTrue);
        sfRenderWindow_drawSprite(window, sp, NULL);
        sfSprite_destroy(sp);

        if (showPopup)
        {
            sfRenderWindow_drawRectangleShape(window, popup, NULL);
            sfRenderWindow_drawText(window, text, NULL);
                    sfRenderWindow_drawRectangleShape(window,  closeButton,
NULL);
        }
    }
    else if (state == RULES)
    {
        drawRules(window, font);
    }

    sfRenderWindow_display(window);
  }

  for (int i = 0; i < ELEMENTS_PER_ROW * ELEMENTS_PER_COLUMN;
i++)
  {
```

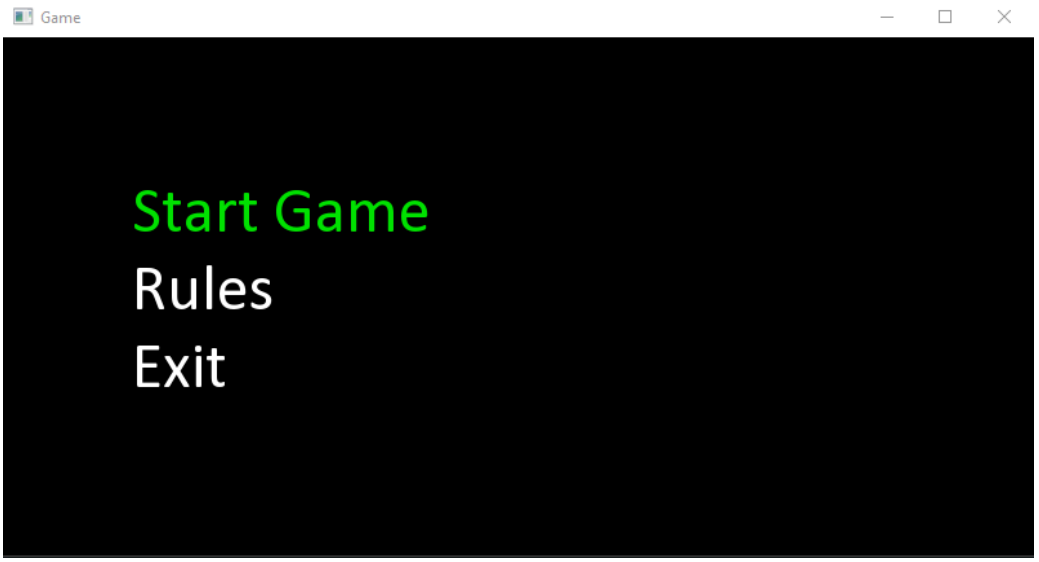
```
        sfTexture_destroy(pictures[i].texture);
        sfSprite_destroy(pictures[i].sprite);
    }

    sfText_destroy(text);
    sfFont_destroy(font);
    sfRectangleShape_destroy(closeButton);
    sfRectangleShape_destroy(popup);

    sfMusic_destroy(music);
    sfTexture_destroy(formula_texture);
    sfSprite_destroy(form_sp);
    sfRenderTexture_destroy(texture);
    sfRenderWindow_destroy(window);
    return 0;
}
```

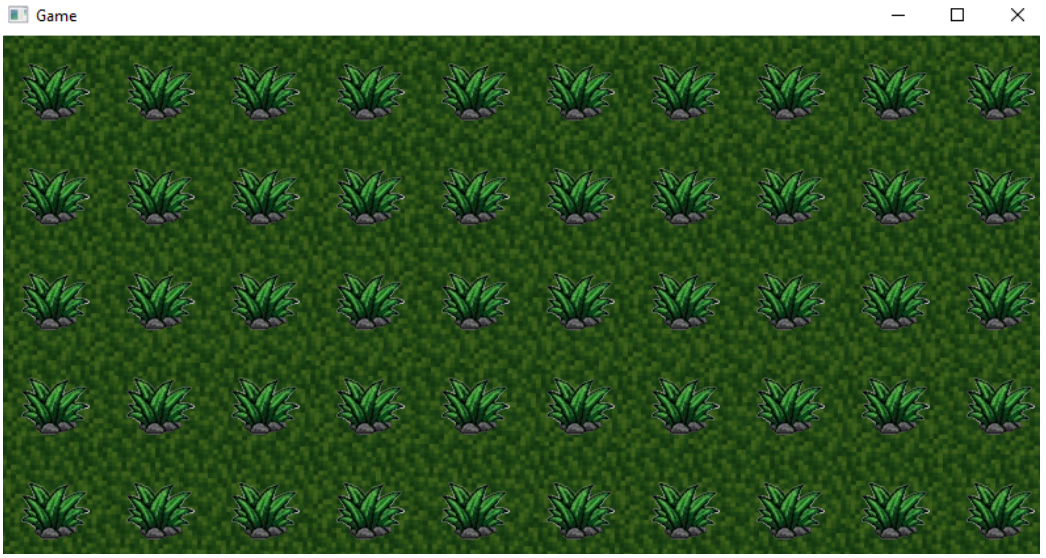
Описання блоків:

1. Початок програми та визначення констант та глобальних змінних:Включення необхідних бібліотек.Визначення розмірів поля, розмірів елементів, розмірів вікна тощо.Визначення констант, таких як рядки меню і кількість пунктів меню.Оголошення змінних для обліку кількості лисиць, вибраного індексу картинки тощо.
2. Опис структури MiniPicture та декларація функцій:Оголошення структури `MiniPicture`, яка представляє маленьке зображення.Оголошення функцій, які будуть використовуватися пізніше в коді.
3. Функції для малювання меню та правил гри:`drawMenu`: Малює меню з використанням SFML.`drawRules`: Виводить правила гри на екран.

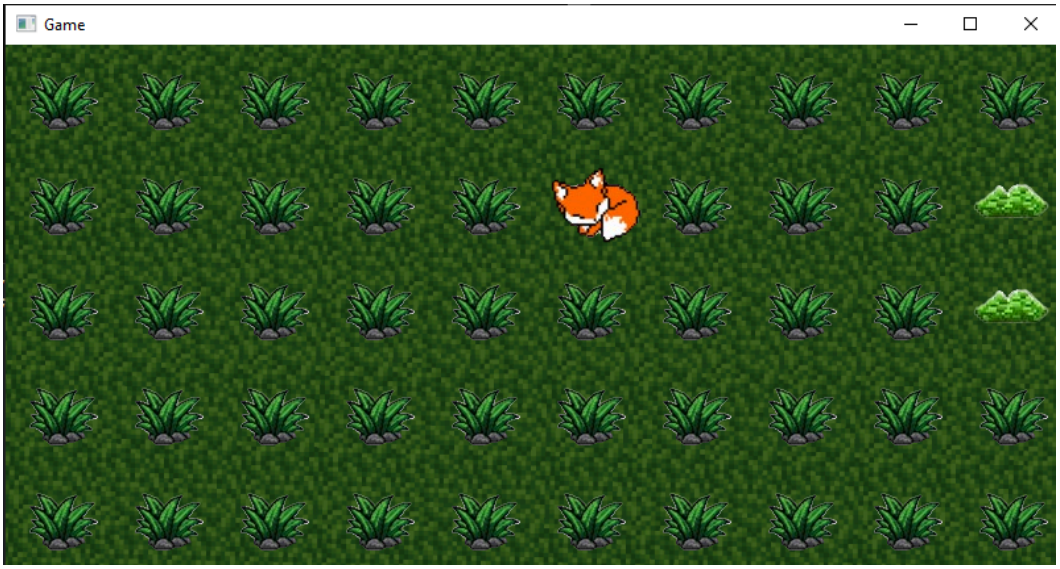
4. Функції для введення координат та обчислення кількості лисиць:`input_coordinates`: Отримує координати натискання миші та визначає кількість лисиць у вибраному місці.`count_foxes`: Обчислює кількість лисиць у вертикальних, горизонтальних та діагональних напрямках.
5. Функції для створення маленьких зображень та вспливаючого вікна:`minipictures`: Створює маленькі зображення для кожного елементу поля.`pop_up_window_text`: Генерує текст для вспливаючого вікна з інформацією про кількість лисиць.`pop_up_window`: Створює вспливаюче вікно з інформацією про кількість лисиць.
6. Функція для генерації розташування лисиць на полі:`generation_fox`: Заповнює поле лисицями за допомогою випадкового розташування.
7. Основна функція main:Ініціалізує SFML та створює вікно гри.Завантажує текстури, музику та шрифти.Здійснює обробку подій та оновлення відображення згідно зі станом гри.
8. Цикл обробки подій у головній функції main:Обробляє події користувача в залежності від поточного стану гри (головне меню, гра, правила гри, гра закінчена).
9. Очищення ресурсів та завершення програми:Звільнення пам'яті, вивід повідомлень про помилки та завершення програми зі статусом 0.

Результати:


Game
Start Game
Rules
Exit

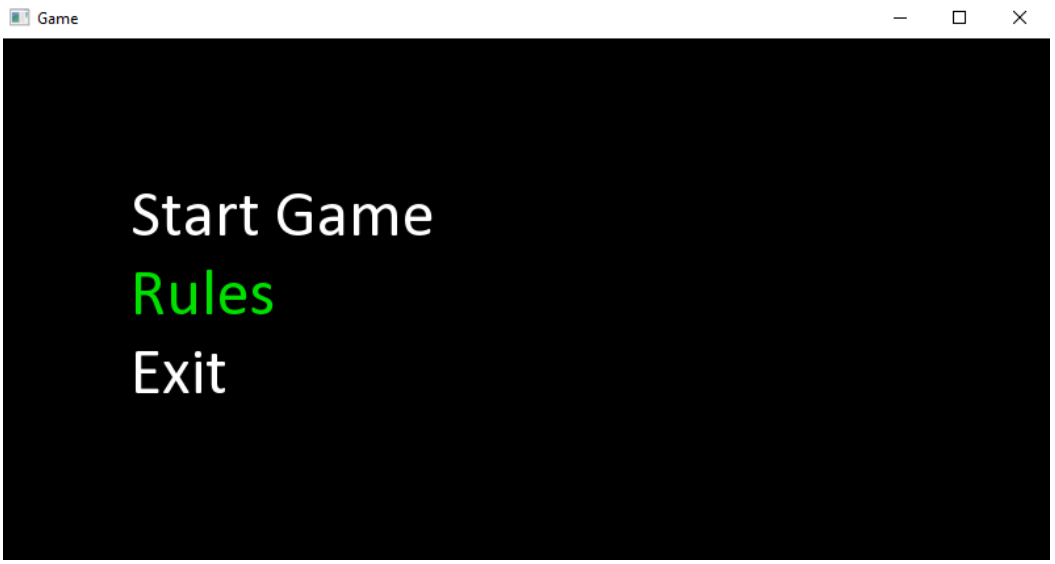

Game

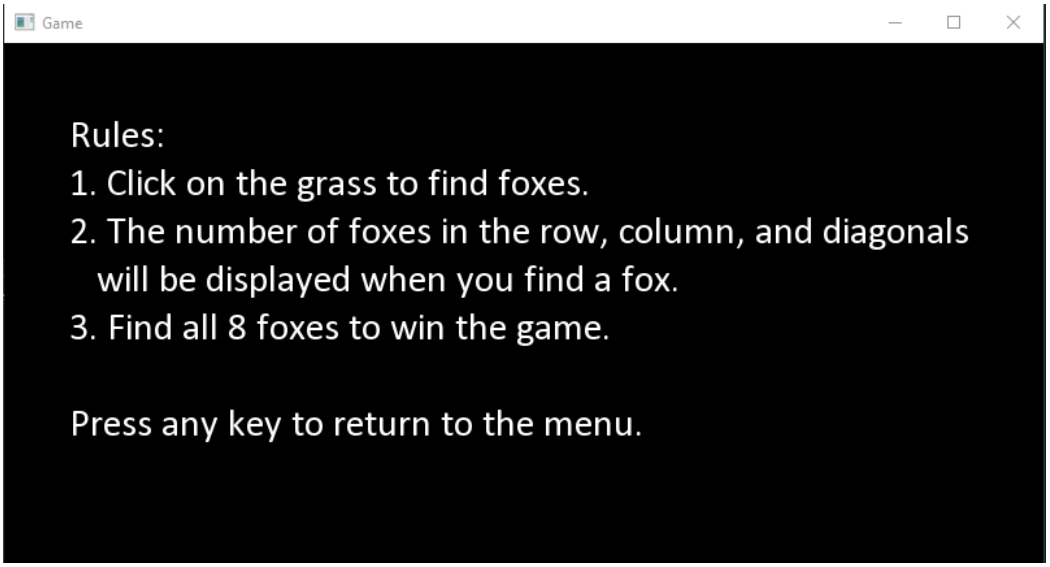

Game
Fox hunting
8 foxes were not found
vertical: 1
horizontal: 0
diagonal: 2

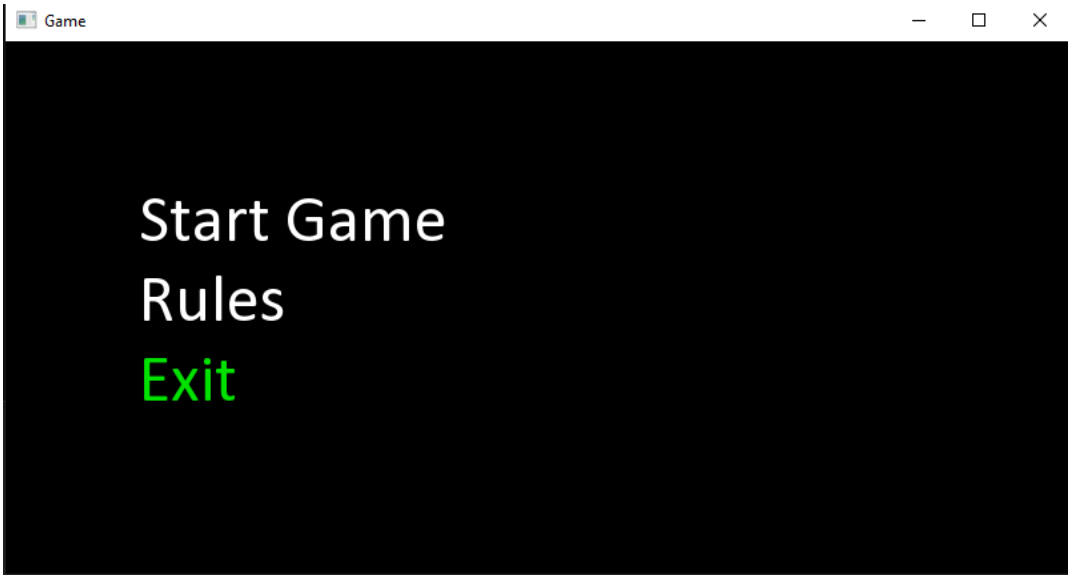

Game


Game
Fox hunting
6 foxes were not found
vertical: 0
horizontal: 0
diagonal: 1


Game
Fox hunting
0 foxes were not found
vertical: 2
horizontal: 0
diagonal: 2


Game
Start Game
Rules
Exit


Game
Rules:
1. Click on the grass to find foxes.
2. The number of foxes in the row, column, and diagonals will be displayed when you find a fox.
3. Find all 8 foxes to win the game.
Press any key to return to the menu.


Game
Start Game
Rules
Exit