Fakultet inženjerskih nauka,

Univerzitet u Kragujevcu

Računarska grafika

Projektni zadatak: Spy_Hunter

Student: Anica Arsić 633/2020

Predmetni nastavnici:

Nenad Filipovic

Tijana Šušterić

.

SADRŽAJ

1.UVOD

U ovom seminarskom radu predstavljam projekt razvoja igre „Spy Hunter“ korišćenjem OpenGL-a. Cilj igre je da igrač upravlja vozilom koje se kreće po putu i štiti civilno vozilo od neprijateljskih vozila koja se kreću prema njemu. Kroz igru, igrač može skupljati poene, gubiti živote i napredovati kroz nivoe, dok se suočava sa različitim izazovima.

Opis igre

Glavni elementi igre uključuju:

1. **Vozilo Igrača**: Ovo je glavni entitet koji igrač kontroliše. Prikazan je crvenom bojom i ima dimenzije 60x100 piksela. Igrač može pomerati ovo vozilo levo i desno duž ekrana, kao i kretati se gore i dole. Uloga vozila igrača je da štiti civilno vozilo od neprijateljskih napada i da aktivno puca na neprijateljske mete.
2. **Neprijateljska Vozila**: Ova vozila su obojena zelenom bojom i predstavljaju prepreke koje se kreću prema civilnom vozilu. Kada neprijateljsko vozilo dođe u kontakt sa civilnim vozilom ili prošao pored njega bez da je pogođeno, igrač gubi poene i živote.
3. **Civilno Vozilo**: Ova vozila su plave boje i predstavljaju ciljeve koje igrač treba da zaštiti. Ako civilno vozilo bude pogođeno ili ne zaštiti od neprijateljskih vozila, igrač gubi poene.
4. **Metak**: Metak je predstavljen žutom bojom i koristi se za napad na neprijateljska vozila. Metak se pomera prema gore od pozicije vozila igrača, a ako pogodi neprijateljsko vozilo, igrač dobija poene, dok se neprijateljsko vozilo ponovo postavlja na vrh ekrana.
5. **Poeni i Životi**: Igrač skuplja poene za svaki pogođeni cilj i za svaki nivo koji pređe. Gubi poene i živote ako neprijateljsko vozilo prođe pored civilnog vozila ili ako se udari u njega. Igra se završava kada igrač izgubi sve živote.
6. **Životi**: Igrač takodje moze da dobije bonus živote.

Postavka Zadataka

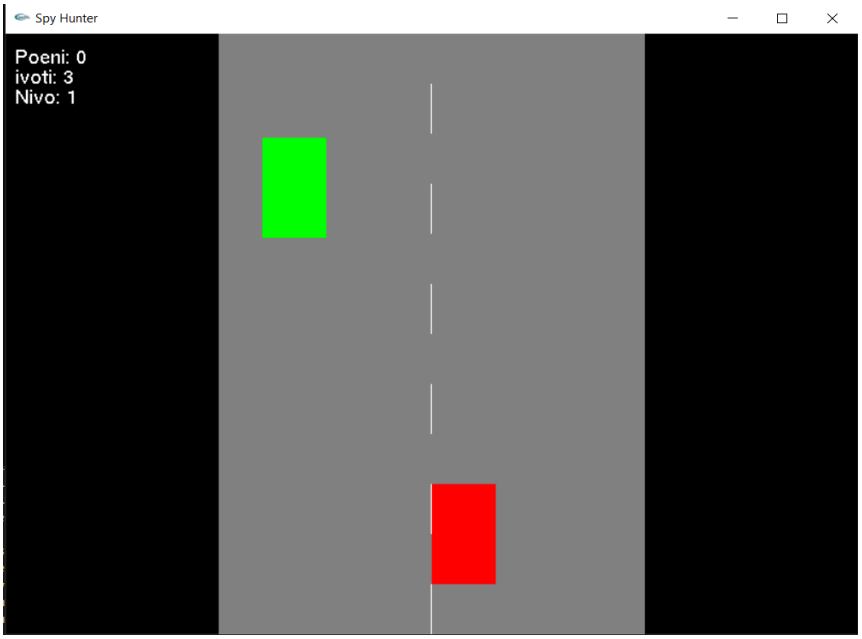
U projektu su postavljeni sledeći zadaci:

- Razviti osnovnu igru u kojoj igrač kontroliše vozilo.
- Implementirati kretanje vozila, metaka i neprijateljskih vozila.
- Dodati sistem poena, nivoa i života.
- Uključiti mehanizme za gubitak poena i života prilikom promašaja ili kontakta sa neprijateljskim vozilima i civilnim vozilom.

Kroz ovaj rad, fokusirali smo se na dizajn i implementaciju igre koristeći OpenGL biblioteku, sa ciljem stvaranja interaktivnog i uzbudljivog iskustva za igrače.

2.0pis izrade i izgled aplikacije

Prvi ekran koji se prikazuje pri pokretanju igrice prikazan je na slici 1.



Slika1 Pocetni Ekran

Na početku su definisane bibioteke korišćene ya izradu ovog projekta, a zatim su definisane I sve promenljive sa njihovim početnim vrednostima.

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include "Glut.h"

// Jedinstvena deklaracija pozicije igrača
float playerX = 400.0f; // X pozicija igrača
float playerY = 50.0f;  // Početna Y pozicija igrača
float obstacleX = 300;  // Početna X pozicija prepreke
float obstacleY = 600;  // Početna Y pozicija prepreke
int score = 0;  // Početni broj poena
int lives = 3;  // Početni broj života
int level = 1;  // Početni nivo

float bonusLifeX = 0;
float bonusLifeY = 0;
bool bonusLifeActive = false;

float civilianX = 400.0f;  // Početna X pozicija civilnog vozila
float civilianY = 600.0f;  // Početna Y pozicija civilnog vozila
bool isCivilianActive = false;  // Da li je civilno vozilo aktivno


// Pozicija i status metka
float bulletX = 0;
float bulletY = 0;
bool bulletFired = false;  // Da li je metak ispaljen?
```

Funkcija init()

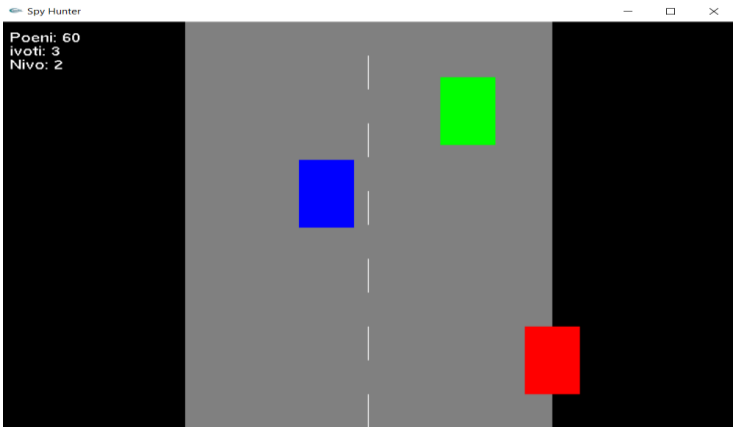
Ova funkcija postavlja osnovnu boju pozadine i dimenzije prikaza ekrana:

```
void init() {
    glClearColor(0.0, 0.0, 0.0, 1.0);  // Crna boja pozadine
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0, 800, 0, 600);  // Dvodimenzionalne koordinate ekrana
}
```

Slika 3

Funkcija display()

Ova funkcija se koristi za crtanje svih objekata u igri kao što su put, vozilo igrača, neprijateljsko vozilo I civilno voyilo:



```
void display() {
    glClear(GL_COLOR_BUFFER_BIT);

    // Crtanje puta
    glColor3f(0.5, 0.5, 0.5);  // Siva boja puta
    glBegin(GL_POLYGON);
    glVertex2f(200, 0);
    glVertex2f(600, 0);
    glVertex2f(600, 600);
    glVertex2f(200, 600);
    glEnd();

    // Crtanje belih linija (sredina puta)
    glColor3f(1.0, 1.0, 1.0);
    glBegin(GL_LINES);
    for (int i = 0; i < 600; i += 100) {
        glVertex2f(400, i);
        glVertex2f(400, i + 50);
    }
    glEnd();

    // Crtanje vozila igrača
    glColor3f(1.0, 0.0, 0.0);  // Crvena boja vozila igrača
    glBegin(GL_POLYGON);
    glVertex2f(playerX, playerY);               // Donji levi ugao
    glVertex2f(playerX + 60, playerY);          // Donji desni ugao
    glVertex2f(playerX + 60, playerY + 100);    // Gornji desni ugao
    glVertex2f(playerX, playerY + 100);         // Gornji levi ugao
    glEnd();
```

```
    // Crtanje neprijateljskog vozila (prepreke)
    glColor3f(0.0, 1.0, 0.0);  // Zelena boja neprijatelja
    glBegin(GL_POLYGON);
    glVertex2f(obstacleX, obstacleY);       // Donji levi ugao
    glVertex2f(obstacleX + 60, obstacleY);  // Donji desni ugao
    glVertex2f(obstacleX + 60, obstacleY + 100); // Gornji desni ugao
    glVertex2f(obstacleX, obstacleY + 100); // Gornji levi ugao
    glEnd();

    displayScore();  // Prikaz poena
    displayLives();  // Prikaz broja života
    displayLevel();  // Prikaz trenutnog nivoa
    drawBullet(); // Crtanje metka
    drawBonusLife();  // Crtanje bonus života
    drawCivilian();  // Crtanje civilnog vozila


    glFlush();
}
```

Funkcija updateBullet()

Ova funkcija ažurira poziciju metka i proverava da li je pogodio neprijatelja:

```
// Funkcija za kretanje metka
void updateBullet(int value) {
    if (bulletFired) {
        bulletY += 10;  // Metak se kreće prema gore
        if (bulletY > 600) {  // Ako metak izađe sa ekrana
            bulletFired = false;  // Reset metka
        }

        // Proveri da li je metak pogodio neprijatelja
        if (bulletX > obstacleX && bulletX < obstacleX + 60 && bulletY > obstacleY && bulletY < obstacleY + 100) {
            score += 10;  // Dodaj poene
            printf("Pogodak! Poeni: %d\n", score);
            bulletFired = false;  // Reset metka
            obstacleY = 600;  // Resetuj neprijatelja
            obstacleX = 200 + rand() % 400;  // Nasumična X pozicija neprijatelja
        }
    }

    // Provera da li je metak pogodio civilno vozilo
    if (bulletX > civilianX && bulletX < civilianX + 60 && bulletY > civilianY && bulletY < civilianY + 100) {
        score -= 20;  // Oduzimanje poena
        printf("Pucao si na civilno vozilo! Poeni: %d\n", score);
        bulletFired = false;  // Reset metka
        isCivilianActive = false;  // Civilno vozilo nestaje nakon pogodka
        civilianY = 600;
    }

    glutPostRedisplay();
    glutTimerFunc(50, updateBullet, 0);  // Postavi tajmer za sledeće ažuriranje
}
```

Slika 4

U okviru igre, funkcija ”handleKeys” i “ handleKeyboard” je ključna za upravljanje kretanjem vozila igrača i ispaljivanje metaka.

Kontrola Kretanja Vozila:

- **Pomeranje Gore (GLUT_KEY_UP)**: Kada igrač pritisne taster za pomeranje vozila prema gore, funkcija proverava da li je vozilo još uvek unutar dozvoljenih granica ekrana. Ako nije dostignuta gornja granica (500 piksela od vrha ekrana), vozilo se pomera za 10 piksela gore.
- **Pomeranje Dole (GLUT_KEY_DOWN)**: Kada igrač pritisne taster za pomeranje vozila prema dole, funkcija proverava da li je vozilo iznad donje granice ekrana. Ako vozilo nije blizu donjeg ruba (0 piksela), pomera se za 10 piksela dole.
- **Pomeranje Levo (GLUT_KEY_LEFT)**: Funkcija omogućava pomeranje vozila levo, sve dok vozilo ne dođe do leve ivice ekrana (0 piksela). Pomeranje se vrši za 10 piksela u levo.
- **Pomeranje Desno (GLUT_KEY_RIGHT)**: Ako igrač pritisne taster za pomeranje vozila desno, funkcija proverava da li je vozilo blizu desne ivice ekrana. Ako nije dostignuta desna granica (740 piksela), vozilo se pomera za 10 piksela udesno.

Ispaljivanje Metaka:

- **Pritisak na SPACE Taster**: Kada igrač pritisne taster SPACE, funkcija proverava da li je metak već ispaljen. Ako metak nije ispaljen (bulletFired je false), funkcija postavlja varijable za metak (bulletX i bulletY) na poziciju iznad vozila igrača i menja status metka na true kako bi označila da je metak ispaljen. Metak će se kretati prema gore od trenutne pozicije vozila.

Funkcija handleKeys()

```
void handleKeys(int key, int x, int y) {
    switch (key) {
    case GLUT_KEY_UP:
        if (playerY + 10 < 500) { // Ograničava kretanje prema vrhu
            playerY += 10.0f;
        }
        break;
    case GLUT_KEY_DOWN:
        if (playerY - 10 > 0) {  // Ograničava kretanje prema dnu
            playerY -= 10.0f;
        }
        break;
    case GLUT_KEY_LEFT:
        if (playerX > 0) {
            playerX -= 10.0f;
        }
        break;
    case GLUT_KEY_RIGHT:
        if (playerX < 740) {  // Ograničenje za desnu stranu
            playerX += 10.0f;
        }
        break;
    }


    glutPostRedisplay();  // Ponovno iscrtavanje
}
void handleKeyboard(unsigned char key, int x, int y) {
    if (key == ' ') {  // Ako se pritisne SPACE taster
        if (!bulletFired) {  // Ako metak već nije ispaljen
            bulletFired = true;
            bulletX = playerX + 25;  // Postavi metak iznad vozila igrača
            bulletY = playerY + 100;
        }
    }
    glutPostRedisplay();  // Ponovno iscrtavanje
}
```

Slika 5

Funkcija checkCollision()

Funkcija checkCollision() služi za detekciju sudara između vozila igrača i neprijateljske prepreke. Poredi koordinate vozila i prepreke kako bi utvrdila da li su se njihovi pravougaonici preklopili. Ako se detektuje sudar, smanjuje broj života igrača za jedan, ispisuje odgovarajuću poruku u konzolu i vraća true. Ako igrač izgubi sve živote, funkcija završava igru. U suprotnom, vraća false ako nema sudara.

```
bool checkCollision() {
    if (playerX < obstacleX + 60 && playerX + 60 > obstacleX &&
        playerY < obstacleY + 100 && playerY + 100 > obstacleY) {
        lives--;  // Gubiš život
        printf("Životi preostali: %d\n", lives);
        if (lives <= 0) {
            printf("Kraj igre! Izgubio si sve živote.\n");
            exit(0);  // Izađi iz igre
        }
        return true;
    }
    return false;
}
```

Slika 6

Funkcija update()

Funkcija update() ažurira stanje igre tako što pomera neprijateljske prepreke prema dole, povećava nivo igre ako igrač prikupi dovoljno poena i proverava da li su prepreke prošle kroz ekran bez da budu pogođene. Ako je prepreka prošla kroz ekran bez da je pogođena, broj poena se smanjuje. Funkcija takođe proverava sudare između vozila i prepreka koristeći checkCollision() i resetuje poziciju prepreka ako su izašle iz okvira. Postavlja vremenski tajmer za ponovo ažuriranje stanja igre.

3.ZAKLJUCAK

Rad na ovom projektu bio je izazovan, ali i veoma poučan. Kroz razvoj igre naučila sam kako da implementiram osnovne elemente igre koristeći OpenGL, uključujući kretanje objekata, detekciju sudara i upravljanje stanjima igre. Takođe, rad sa metkom i neprijateljskim vozilima, kao i dodavanje kompleksnih funkcionalnosti poput skupljanja poena i upravljanja životima, značajno je poboljšao moje razumevanje programiranja grafike i interaktivnih aplikacija. Ovaj projekat je obogatio moje veštine u programiranju i dao mi vredno iskustvo u razvoju igara.