Dungeon Explorer

Introducere

Proiectul consta intr-un joc de tip dungeon crawler genre Roguelike RPG, in care jucatorul controleaza un personaj ce exploreaza un labirint, infrunta inamici si colecteaza diferite resurse sau imbunatatiri, la fiecare explorare labirintul va fi diferit camerele vor fi generate procedural pe baza unor reguli astfel fiecare eplorare va aduce o experienta noua pentru jucator . Scopul proiectului este de implementa un joc cu mai multe mecanici precum logica de miscare, gestionarea resurselor, AI basic pentru inamici, interactiunea jucator cu mediul inconjurator si toatea acestea folosind niste resurse limitate. Idea de la care am pornit a fost pasiunea mea pentru jocuri, astfel m-am gandit cum as putea face un proiect care sa fie interesant pentru mine, astfel am ajuns la aceasta idee. Un joc de tip dungeon crawler poate poate fi simplificat pentru a fi adaptat la constrangerile hardware insa raman si o groaza de mecanici pentru a face jocul interant in continuare.

Descriere generală

Sistemul este compus din următoarele componente:

- Joystick-ul este utilizat pentru controlul mișcării caracterului din joc.
- Keypad-ul este folosit pentru atacarea inamicilor și pentru interacțiunea cu diferite elemente din joc.
- Bloc de control principal: Arduino Mega se ocupă de afișarea sprite-urilor pe ecran. La controler sunt conectate joystickul, keypadul si buzzerul
- TFT Shield v2.2 este utilizat pentru conversia nivelului de tensiune de la 5V la 3.3V, deoarece pinii ecranului funcționează la 3.3V.
- Ecranul TFT LCD de 3.2 inch afișează harta, jucătorul și inamicii.

Hardware Design

Ecranul ILI9341 TFT LCD Display Module este conectat în modul 16-bit paralel la un Arduino Mega 2560. Pinii de comunicație ai ecranului funcționează la 3.3V, în timp ce plăcuța oferă semnale la 5V. Pentru a realiza conexiunea dintre ecran și microcontroller am utilizat un TFT Shield V2.2, care realizează conversia nivelurilor de tensiune.

Shield-ul se conectează la plăcuță astfel:

- Ocupă toți pinii de power;
- Utilizează pinii digitali 0–7 și 22–53.

Această mapare a pinilor nu a fost realizată manual, deoarece shield-ul permite un singur mod de conectare la Arduino Mega.

Conexiunile dintre shield și ecran sunt următoarele:

- Pinii **DB0–DB15** sunt utilizați pentru comunicația în modul 16-bit paralel;
- Pinii **CS, RD, WR** și **RS** sunt utilizați pentru controlul ecranului și pentru funcționarea bibliotecii grafice;
- Pinii **VDD** și **GND** sunt utilizați pentru alimentare;
- Pinii **CLK, MOSI, MISO** și **SDCS** sunt utilizați pentru comunicarea cu slotul de card SD, necesar salvării datelor persistente.

Pentru conectarea joystick-ului am utilizat:

- **A0** pentru VRx;
- **A1** pentru VRy.

Acești pini au fost aleși deoarece semnalele VRx și VRy sunt citite analogic.

Pentru conectarea keypad-ului am utilizat:

- Pinii **A8–A11**, câte unul pentru fiecare buton (1–4).

Acești pini au fost aleși deoarece suportă întreruperi de tip PCINT, necesare pentru detectarea apăsării butoanelor.

Buzzer-ul a fost conectat la pinul digital 13, deoarece este necesar un pin cu suport PWM pentru controlul acestuia.

Software Design

Mediu de dezvoltare:

- Neovim + PlatformIO
- C/C++

Librarii si surse utilizate:

- UTFT: biblioteca a fost folosita pentru a desena sprite-urile pe ecran. Am folosit aceasta biblioteca deoarece are suport pentru 16 bit parallel care a fost modul de conectare a ecranului cu microcontrolerul Arduino Mega.

Functionalitati din laborator:

- ADC: Joystickul are 2 pini VRx si VRy care sunt conectati la 2 pini analogi de pe controler si folosesc adc pentru a dectecta daca joystickul s-a mutat sau nu
- Timere: pentru a comuta frecventa buzzer-ului
- GPIO: Pentru a controla componenetele prin registri
- Intreruperi: pentru a determina daca un buton de pe keyboard sau pentru a citi valorile de pe joystick sunt am folosit intreruperi pentru ca nu am vrut sa am operatii blocante in ciclul principal al jocului astfel jocul ar fi avut micro lag-uri in timpul jocului

Flow-ul jocului:

La pornirea jocului se aude un sunet de la buzzer care semnalizeaza inceperea partidei. Apoi se incarca primul nivel, iar jucatorul intra efectiv in joc care poate dura atata timp cat doreste. La final daca hp-ul jucatorului a ajuns la 0 atunci apare un ecran de Game Over care arata cate camere a parcurs si cati inamici a invins. Aceste statistici sunt afisate pentru a permite inregistrarea recordurilor astfel ca player-ul sa aiba motivatia sa joace din nou.

Implementarea jocului:

Pentru a implementa jocul am definit mai multe clase pentru a modulariza codul:

- **Keypad**: aceasta clasa se ocupa de functionalitatea keypad-ului. In cadrul proiectului am folosit un keypad 1×4. Pentru inceput clasa initializeaza registrii corespunzatori pinilor la care este conecata keypad-ul la microcontroler. Apoi prin functia **getKey** se determina ce buton a fost apasat. Verificarea daca un buton a fost apasat se procude in interiorul unei intreruperi.
- **Joystick**: aceasta clasa se ocupa de functionalitatea joystick-ului. Clasa initializeaza registrii corespunzatori pinilor de pe placuta. Apoi prin intermediul functiilor **getX** si **getY** determina pozitia joystickului. Citirea pentru valorile din joystick se fac intr-o intrerupere ADC care se intampla de fiecare data cand conversia ADC sa terminat. Astfel initial citest X apoi fac switch pe canalul pentru Y si citesc Y si fac switch pe canalul lui X si tot asa. Am folosit aceasta intrerupere deoarece nu am vrut ca citirea joystick-ului sa fie o operatie blocanta ci sa fie facut de hardware.
- **DrawUtils**: aceasta clasa se ocupa de desenarea imaginilor pe ecran. Este un mini engine pentru a desena. Pentru inceput clasa intializeaza ecranul. Apoi in clasa principala **Game** utilizez functiile din aceasta clasa pentru de desena blocuri, inamici pe ecran in pozitii predefinite.
- **Game**: aceasta este clasa care se ocupa de logica jocului.

Implementarea clasei Game:

In aceasta sectiune voi descrie functiile principale din cadrul clasei Game

- initGame: initializeaza toate resursele cum ar fi joystick-ul, keypad-ul, ecranul si se deseneaza primul nivel al jocului care este un fel de tutorial unde jucatorul se poate obisnui cu controalele din joc.
- loop: game loop-ul principal aici se face efectiv update la toti inamicii, player, detecteaza coliziuni. Citeste input-urile de la player pentru a le executa in joc
- resetGame: daca jucatorul a pierdut are optiunea de a da restart la joc pentru a reconecta placuta la sursa de curent sau pentru a nu da upload din nou la cod. Aici hp-ul, attack power-ul jucatorului sunt resetate si totul incepe de la nivelul initial.
- updatePlayerPos, updateEnemies, updateProjectiles: aceste functii fac update la pozitia jucatorului, inamicilor, personajului principal verificand pe parcurs coliziunea cu blocurile, alti inamici.
- drawLevel: fiecare nivel este reprezentat in forma ascii intr-o matrice 14×10 unde # reprezinta bloc obisnuit, P pozitia la jucator, E inamici, C chest-uri cu diferite upgrade-uri. aceasta functie parcurge toata matricea si deseneaza nivelul
- interactChest: in joc exista 2 stats-uri hp-ul jucatorului si puterea lui de atac adica cat damage da cand un proiectil a avut o coliziune cu un inamic. Cand jucatorul interactioneaza cu un chest atunci in mod random se va decide ce fel de resursa va primi jucatorul
- pickNextLevel: aceasta functie se asigura ca urmatorul nivel care va fi in joc poate fi conectat cu precedentul. Astfel daca jucatorul a intrat pe usa de sus atunci trebuie sa asigur ca urmatorul nivel tot are usa din partea de sus a nivelului. Plus aceasta functie asigura faptul ca nivelurile cu chest-uri si fara chest-uri vor aparea balansat adica nu va fi situatia in care jucatorul da de 5 camere la rand cu chest-uri.

Jocul functioneaza asa cum mi-am propus deoarece am facut mai multe expeditii in care am explorat mai multe camere, inclusiv interactiunea cu inamicii, chest-uri si proiectile.

Optimizari facute:

Desi Arduino Mega are 8 KB de SRAM pentru jocul meu aceasta dimensiune este prea mica. Astfel m-am folosit de memoria flash a microcontrelui. Toate sprite-urile folosite in jocul meu au fost incarcate in memoria flash folosind in cod atributul PROGMEM. Fiecare sprite are in medie dimensiunea 24×24 pixeli. Nivelele tot sunt salvate in memoria flash. Am in total 30 de nivele, initial jucatorul se afla la nivelul 0 care este practic gol, in acest timp urmatorul nivel este deja decis, care va fi nivelul urmator este ales la random.

Miscarea entatilor pe ecran este la fel optimizata. Astfel este redesenat doar dreptunghiul din ecran care se modifica si este marginit de pozitia veche si pozitia noua a entitatii. Apoi sprite-urile sunt reprezentate ca niste vectori de 32×32, 16×16, 24×24 elemente dar au si portiuni care sunt transparente astfel am ales o culoare care va reprezenta ca acel pixel este transparent si astfel el nu va fi desenat. Astfel am incercat sa compensez faptul ca placuta are o frecventa a procesorului mai mica ca un procesor normal si sa nu am ca personajul apare si dispare de pe ecran, dar se modifica doar acei pixeli care trebuie redesenati.

Concluzii, Rezultate

Acest proiect m-a ajutat foarte mult sa inteleg cum sa folosesc mai bine microcontrolerele cum ar fi

Arduino si cum sa lucrez cu un ecran care are un refresh rate mai mic Ca rezultate proiectul este finisat si jocul este interesant de jucat in timpul liber.

Link demo: demo

Download

dungeonexplorer.zip

https://github.com/SanduCondorache/DungeonExplorer

Bibliografie/Resurse

Arduino Mega datasheet 1

Arduino Mega datasheet 2

TFT Shield v2.2