Flappy Bird

Introducere

Acest proiect intenționează să implementeze jocul Flappy Bird, în care o jucătorul controlează o pasăre care trebuie să zboare printr-o serie de țevi. Fiecare țeavă are o gaură prin care pasărea poate trece. Pasărea poate zbura în sus doar prin input-ul jucătorului, altfel cade constant.

Scopul proiectului este de a încerca să implementez un joc popular de telefon pe un Arduino unde apar limitări de memorie / putere de procesare. Apăsarea repetată de butoane la laboratoarele de PM mi-a adus aminte de zilele în care jucam Flappy Bird pe telefon, lucru care m-a făcut să aleg acest proiect. Proiectul este util deoarece arată o platformă diferită pentru jocuri, și, după finalizare, oferă posibilitatea de a te juca Flappy Bird.

Descriere generală

Pentru a interacționa cu jocul, jucătorul trebuie să apese butonul, lucru ce cauzează pasărea să zboare. Poziția păsării și a țevilor este afișată constant pe ecranul LCD. Difuzorul este folosit pentru a semnala diferite evenimente (ex. Pasărea zboară). Dacă pasărea intră în contact cu o țeavă, jocul se termină.

Hardware Design

Piese:

- Placă Arduino Nano
- breadboard
- buton
- ecran LCD 128×128 1.44'
- buzzer
- fire de legătură

Montaj

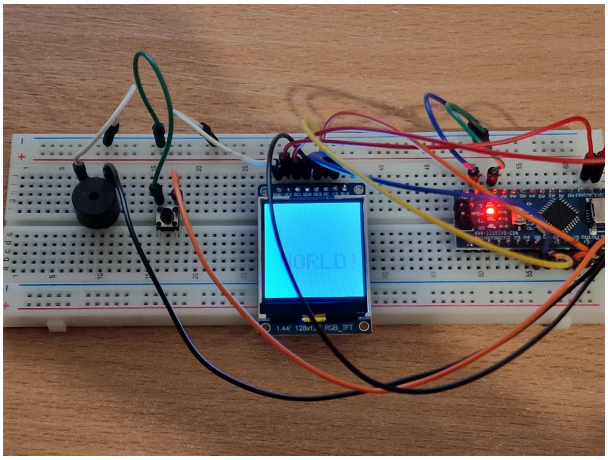

Diagrama electrica

Software Design

Biblioteci folosite:

“Adafruit_GFX.h” - biblioteca core pentru grafica

“Adafruit_ST7735.h” - biblioteca pentru ecranul specific ales

Am ales aceste biblioteci deoarece faciliteaza scrierea de cod pentru ecranul LCD ales.

Mai intai am conectat butonul la arduino, folosind rezistenta interna de pull up. Am realizat aceasta conexiune prin registrii, dupa cum am invatat la laborator. Pentru a se inregistra apasarea butonului doar 1 data a trebuit sa calculez un delay intre ultima citire si sa iau in considerare apasarea doar daca elapsed_time-ul este mai mare decat delay-ul.

Urmatoarea componenta a fost buzzer-ul, pe care l-am implementat tot prin registrii. Deoarece acesta scotea sunete cand era plugged, a trebuit sa il opresc dupa fiecare utilizare. Emiterea de sunet este legata de apasarea butonului.

In continuare a urmat ecranul. Initial am incercat ca la inceputul fiecarei rulari de functie “loop” sa dau fill la ecran cu culoarea neagra (culoarea de fill), procedeu utilzat si in jocuri, ca dupa sa pot desena elementele de joc la noile pozitii. Din pacate ecranul nu este destul de performant cat sa suporte atat de multe schimbari in fiecare frame asa ca a trebuie sa gasesc o modalitate prin care sa updatez doar elementele care isi schimba pozitia.

Din acest motiv am incercat sa limitez numarul de elemente care se schimba. Am inversat jocul Flappy Bird, acum tevile stau pe loc iar pasarea se misca. Pentru a muta pasarea trebuie sa dau fill cu negru la pozitia veche a pasarii si dupa sa o desenez in noua pozitie.

Din cauza dimensiunii limitate a ecranului (128×128, 1.44') am ales sa am 3 tevi pe ecran la un moment dat. Cand pasarea trece de ultima teava, este transportata in stanga ecranului si apar 3 tevi noi (randomizate).

Pentru o teava ii sunt tinute minte pozitia pe axa Ox si pozitia gap-ului (pe Oy). Un gap are mereu acelasi size. Cand apare o teava lucrul care este randomizat este pozitia gap-ului. Pentru a testa daca pasarea s-a lovit de o teava, verific daca este in intervalul tevii si NU se afla in gap. Pentru asta m-am folosit de ideea de Axis Aligned Bounding Boxes (AABB).

In coltul din stanga sus se tine minte scorul jucatorului (pentru fiecare teava de care a trecut, primeste 1 punct), iar cand acesta pierde este afisat pe ecran “Game Over” si scorul curent. Pentru a reincepe jocul, trebuie apasat butonul.

Deoarece chiar si cu viteza de un pixel per frame, pasarea se misca prea repede, am decis sa limitez numarul de frame-uri care se pot rula pe secunda (pentru a evita calculele cu float-uri). Astfel, functia game se ruleaza de un numar limitat de ori pe secunda.

Am incercat sa limitez distanta dintre doua gap-uri consecutive pe verticala, pentru a nu da de situatii imposibile. Acest lucru a functionat dar nu neaparat in totalite. Din cauza randomness-ului nu pot testa toate posibilitatile.

Jucatorul castiga daca ajunge la 100 de puncte.

Rezultate Obţinute

Am o implementare (redusa din prisma complexitatii) a jocului Flappy Bird pe un Arduino. Joculetul este putin frustrant (din cauza randomness-ului), dar acest lucru este in spiritul Flappy Bird.

Jocul functioneaza, buzzer-ul emite sunet cand este apasat buton-ul. Jocul se termina cand jucatorul castiga sau pasarea se loveste de o teava. In alte cuvinte am obtinut rezultate bune.

Concluzii

M-am distrat realizand aceasta implementare. Viteza redusa a ecranului m-a cauzat sa imi schimb modul in care doream sa implementez jocul, dar am gasit o alternativa. Nu m-am lovit de limite de procesare / stocare, deci experienta cu arduino-ul a fost una buna.

Daca o sa mai comand in viitor piese, o sa am mai mare grija sa cumpar piese bine documentate, si care sa vina cu pinii deja lipiti. Am pierdut foarte mult timp cautand datasheet-uri care nu corespund pe internet pentru a afla cum se leaga ecranul, intrucat pe site-ul de unde l-am cumparat nu erau oferite informatii.

Download

flappybirdarduino.zip

Jurnal

13.05.2024 - Sosire Piese

16.05.2024 - Montare Piese

18.05.2024 - Implementare Cod

22.05.2024 - Finalizare Documentatie

Bibliografie/Resurse

https://developer.mozilla.org/en-US/docs/Games/Techniques/2D_collision_detection

https://github.com/adafruit/Adafruit-ST7735-Library

Export to PDF