Amorfos Console - Consola de jocuri programabila

Introducere

Proiectul consta intr-o consola de jocuri, un framework pentru facut jocuri pe ea si un exemplu de joc.

Partea de hardware consta in consola in sine. Pentru input consola are un joystick(care este si buton) si 4 butoane. Jocul este afisat pe un LCD, iar sunetul este redat de un buzzer. Exista 2 LEDuri ce pot fi folosite de asemenea.

Consola ar fi trebuit sa incarce jocuri de pe un card SD pe care sa le ruleze, dar nu am reusit sa fac si asta.

Partea de software consta intr-un framework care poate fi folosit pentru a coda jocuri pentru consola. In acest fel, cine doreste sa faca un joc nu va trebui sa interactioneze cu partea de hardware, ci doar sa aiba cunostinte de programare.

Am adaugat si un joc facut in acest framework ca exemplu de folosire.

Sunt pasionat de game development si am proiecte in sfera asta, asta ca mi s-a parut natural sa fac ceva similar si pentru un proiect de hardware. Am vazut cu cateva luni in urma un videoclip scurt cu cineva care arata ca si-a facut o consola simpla si am vrut de atunci sa fac si eu asta si ce oportunitate mai buna decat proiectul de PM.

Descriere generală

Schema Bloc

Descriere

Pe cardul SD se uploadeaza hex file-ul rezultat din compilarea codului jocului si codului frameworkului. Codul de pe cardul SD va fi incarcat de un bootloader la pornirea consolei.

Displayul va fi tratat ca o matrice. Fiecare entitate are o matrice de pixeli care reprezinta grafica pentru aceasta.

Buzzerul va fi folosit pentru redarea de sunet. Acesta va primi comenzii sub forma de secvente de frecventa si durata.

Inputul va fi citit prin intreruperi pentru d-pad si printr-o citire continua de pe pini pentru joystick. Joystickul functioneaza pe baza unui potentiometru. Joystickul poate fi apasat la randul lui si actioneaza ca un buton.

Exista si 2 LEDuri ce pot fi aprinse si stinse.

Pentru alimentare, se vor folosi 4 baterii. Consola va fi pornita prin intermediul unui switch.

Hardware Design

Schema Hardware

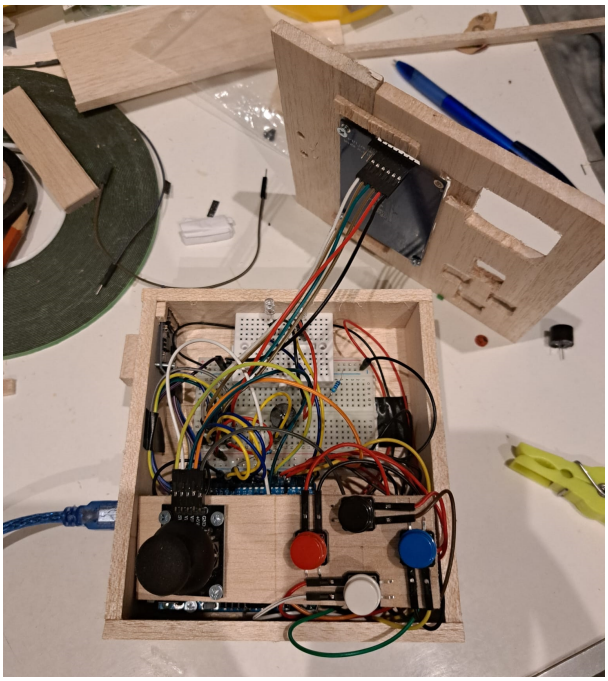
Mai sunt niste conexiuni in plus, dar dispar dintr-un motiv cand incarc poza.

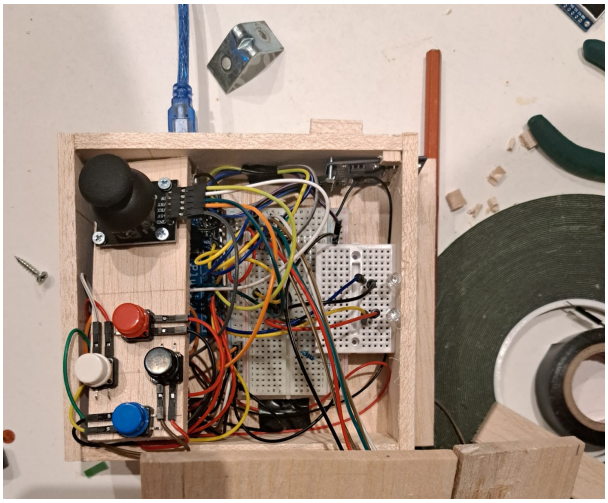
Lista de piese

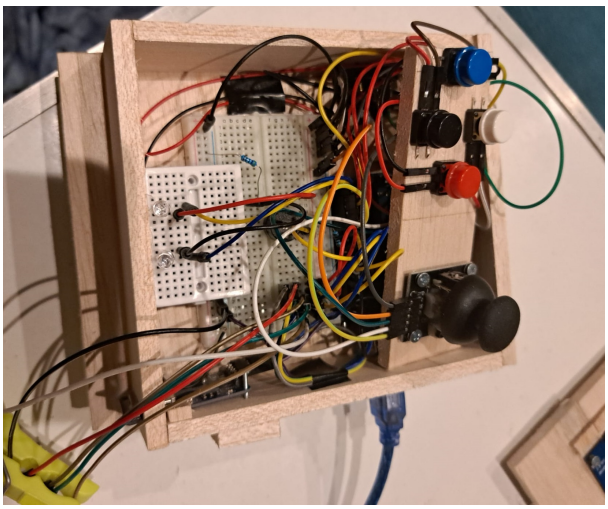
- Arduino UNO R3
- Display LCD
- Joystick
- 4 Butoane
- Buzzer Pasiv
- Cititor SD
- Comutator LED
- 2 LEDuri
- Suport Baterii
- Breadboarduri
- Rezistoare
- Fire

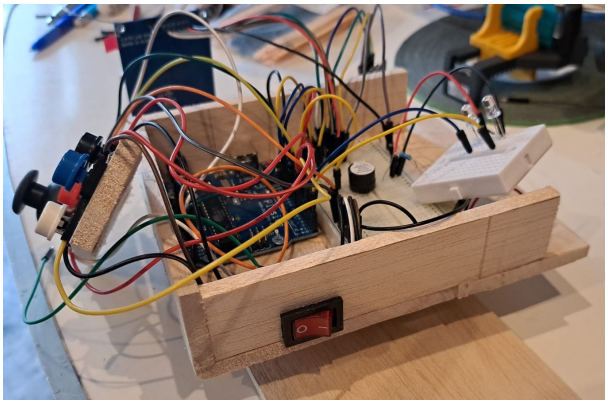
Carcasa este facuta din lemn.

Poze









Software Design

Framework Amorfos

Documentatia si codul pentru frameworkul facut de mine pentru consola se pot gasi aici: https://github.com/Zeyt8/Amorfos-Console

Exemplu joc - Space Defenders

```
#if 1 // Remove this or change it to 1 to use. Also remove the #endif at the bottom of the file.
// This is needed because there are multiple game projects in the repo and they are all implementing the functions from amorfos.h.

#include "../../framework/amorfos.h"
#include <stdint.h>

using namespace amorfos;

enum EntityType {
    PLAYER,
    ENEMY,
    BULLET,
    NOTHING
};

Entity* player;
uint8_t playerHealth = 2;
uint8_t maxBullets = 3;
uint8_t bulletCount = 0;
bool needToReload = false;
int reloadTicks = 2;
float tickTimer = 1.0f;
int ticksForEnemyVertical = 7;
int ticksForSpawnEnemy = 2;
int enemyDir = 1;

static void createPlayer() {
    player = createEntity(EntityType::PLAYER, newVector2(0, 0), newVector3(255, 0, 0), true, true, NULL);
    uint16_t graphics[3][3] = {
        { 0, 1, 0 },
        { 1, 1, 1 },
        { 0, 0, 0 }
    };
    setGraphics(player, graphics);
    setCollisionRadius(player, 1);
}

static void createEnemy(int x, int y) {
    Entity* enemy = createEntity(EntityType::ENEMY, newVector2(x, y), newVector3(0, 255, 0), true, true, NULL);
    uint16_t graphics[3][3] = {
        { 1, 0, 1 },
        { 0, 1, 0 },
        { 1, 0, 1 }
    };
```

```
    setGraphics(enemy, graphics);
    setCollisionRadius(enemy, 1);
}

static void createBullet(int x, int y) {
    Entity* bullet = createEntity(EntityType::BULLET, newVector2(x, y),
newVector3(255, 255, 0), true, true, NULL);
    uint16_t graphics[3][3] = {
        { 0, 0, 0 },
        { 0, 1, 0 },
        { 0, 1, 0 }
    };
    setGraphics(bullet, graphics);
    setCollisionRadius(bullet, 0);
}

void amorfos::start() {
    createPlayer();
    setPosition(player, LCD_WIDTH / 2, LCD_HEIGHT / 2 - 10);
    setLED(true, 0);
    setLED(true, 1);
}

static void onTick()
{
    // reload
    if (needToReload) {
        reloadTicks--;
        if (reloadTicks <= 0) {
            bulletCount = maxBullets;
            reloadTicks = 2;
            needToReload = false;
        }
    }
    // entity movement
    ticksForEnemyVertical--;
    for (int i = 0; i < entityCount; i++) {
        if (entities[i]->type == EntityType::ENEMY) {
            move(entities[i], 5 * enemyDir, 0);
            if (ticksForEnemyVertical <= 0) {
                move(entities[i], 0, -10);
            }
        }
        else if (entities[i]->type == EntityType::BULLET) {
            move(entities[i], 0, 2);
        }
        else if (entities[i]->type == EntityType::PLAYER) {
            move(entities[i], input.joystickX, input.joystickY);
        }
    }
```

```
    if (ticksForEnemyVertical <= 0) {
        ticksForEnemyVertical = 7;
        enemyDir *= -1;
    }
    // enemy ocasionally spawns
    ticksForSpawnEnemy--;
    if (ticksForSpawnEnemy <= 0) {
        ticksForSpawnEnemy = 22;
        for (int i = (LCD_WIDTH / 2) - 12; i < (LCD_WIDTH / 2) + 13; i += 5)
{
            createEnemy(i, LCD_HEIGHT - 10);
        }
    }
}

void amorfos::update(float deltaTime) {
    tickTimer -= deltaTime;
    if (tickTimer <= 0) {
        tickTimer = 1.0f;
        onTick();
    }
}

void amorfos::onButtonPress(int button) {
    if (button == BUTTON_TOP) {
        if (bulletCount > 0) {
            createBullet(player->position.x, player->position.y);
            bulletCount--;
        }
        else {
            needToReload = true;
        }
    }
}

void amorfos::onButtonRelease(int button) {
}

static void gameOver() {
    restart();
}

void amorfos::onCollision(amorfos::Entity* entity1, amorfos::Entity* entity2)
{
    if (entity1->type == EntityType::PLAYER && entity2->type == EntityType::
ENEMY) {
        playerHealth--;
        if (playerHealth == 1) {
            setLED(false, LED1);
        }
        else if (playerHealth == 0) {
```

```
            setLED(false, LED0);
            // game over
            gameOver();
        }
        destroyEntity(entity2);
    }
    else if (entity1->type == EntityType::BULLET && entity2->type ==
EntityType::ENEMY) {
        destroyEntity(entity1);
        destroyEntity(entity2);
    }
}

#endif // If you removed the #if 1 at the top of the file, remove this too.
```

Librarii si surse 3rd-party

Pentru LCD folosesc aceasta librarie: https://github.com/olikraus/ucglib

Pentru a incarca jocurile de pe cardul SD folosesc acest bootloader: https://github.com/zevero/avr_boot

Rezultate Obţinute

Am o consola functionala si un framework usor de folosit pentru a face jocuri. Jocul facut pentru consola merge ok, chiar daca are tot overheadul unui framework in spate si pare ca poate sa duca si mai mult.

Displayul, inputul, audioul functioneaza. LCDul are refresh rate cam mic pentru gameplay smooth, dar este acceptabil.

Din pacate am aflat prea tarziu ca am nevoie de un ISP ca sa pun un alt bootloader pe arduino si desi facusem totul pentru a putea incarca jocuri de pe cardul SD, in lipsa ISP nu am ce sa fac.

Concluzii

Am invatat cum sa ma descurc sa citesc singur data sheeturi si sa experimentez cu componente electronice. De asemenea am aflat si cat de usor este sa le strici.

Debuggingul cu hardware este mult mai greu pentru ca niciodata nu esti sigur daca piesa este buna sau e codul tau gresit.

Sa iti dai seama cum se conecteaza si cum functioneaza piese cu documentatie sumara este foarte

foarte greu.

Shieldurile de arduino mai mult de incurca. Mai bine folosesti componente separate.

Sa faci un framework de facut jocuri nu este asa de greu pe cat credeam.

Download

Hex fileul de pus pe cardul SD pentru Space Defenders:

Bibliografie/Resurse

Laburile de PM:

- https://ocw.cs.pub.ro/courses/pm/lab/lab0-2022
- https://ocw.cs.pub.ro/courses/pm/lab/lab2-2022
- https://ocw.cs.pub.ro/courses/pm/lab/lab5-2022

Singurele surse care mai sunt relevante pentru componentele curente sunt paginile de wiki ale bibliotecilor mentionate mai sus:

- https://github.com/olikraus/ucglib/wiki
- https://github.com/zevero/avr_boot/tree/gh-pages

Surse consultate pentru piese pe care nu le mai folosesc, dar pe care le-am incercat in trecut:

- https://cb-electronics.com/products/funduino-joystick-shield-v1-a-ky-023-shield/
- https://forum.arduino.cc/t/ili9341-tft-display-without-exposed-cs-pin/566641/1
- https://github.com/sumotoy/TFT_ILI9163C