Dancing Stickman

Introducere

Proiectul presupune crearea pe un ecran LCD a unui stickman care sa imite mișcările unei persoane aflate in fata unui XBOX 360 Kinect.

Consider că proiectul meu reprezintă un POC pentru o idee care ar putea fi aplicată pe semafoarele din zonele turistice urmând:

- să încurajeze respectarea semnelor de circulație, fapt care este adesea o problemă in astfel de zone;
- să anime un semafor de altfel anost.

Au fost deja implementate astfel de proiecte având efectele precizate anterior. Aveți un exemplu aici.

Am ales acest proiect din următoarele motive:

- Este mai solicitant din punct de vedere software;
- Este o ocazie bună de a înțelege cum funcționează un Kinect și API-urile puse la dispoziție de acesta;
- Mi-ar plăcea să văd ceva similar în orașele din România.

În Viena foarte multe semafoare au figurine unice în centrul orașului. La Muzeul de Tehnologie din Viena există un proiect similar cu al meu, care a fost și sursa mea de inspirație.

Descriere generală

Pași de funcționare a proiectului:

1. Utilizatorul stă în fața Kinect-ului;
2. Kinect-ul va recunoaște corpul utilizatorului și va comunica cu laptop-ul;
3. Laptop-ul va trimite plăcuței Arduino datele obținute de Kinect;
4. Datele sunt folosite ca input pentru o funcție care obține o matrice booleană;
5. Matricea se trimite display-ului pe care va apărea un stickman care imită utilizatorul.

Hardware Design

Proiectul este alcătuit din următoarele componente:

- Xbox 360 Kinect;
- Power adapter pentru Xbox 360 Kinect;
- Laptop (pentru conectarea la Kinect);
- ATmega328p;
- 3.5″ Display LCD Shield cu rezoluție 320×480.

Proiectul meu este unul software-heavy, iar din punct de vedere al asamblării este trivial si durează sub un minut.


Descriere a asamblării:

- Pe plăcuța Arduino se pune display-ul care este un shield, deci acesta ocupă majoritatea pinilor, iar pe restul îi blochează, nemaiputând fi altceva conectat.
- Plăcuța Arduino se conectează la laptop.
- Adaptorul pentru Kinect are 3 ieșiri, acesta conectându-se la:
  - Kinect;
  - Laptop;
  - Sursă de curent.

De menționat:

- Kinect-ul nu poate fi conectat direct la laptop, deoarece acesta nu poate furniza suficientă putere

prin USB, deci un power adapter este necesar.
- Modelele mai noi de Kinect pot fi conectate direct la laptop, dar modelul folosit de mine este cel lansat în 2010.
- Display-ul LCD ocupă majoritatea pin-urilor de pe plăcuța Arduino, iar pe restul le face inaccesibile.

Software Design

Pentru implementarea funcționalității am utilizat următoarele tehnologii (medii de dezvoltate, limbaje de programare și librării):

- Arduino IDE
  - C++
  - Pentru programarea LCD Display Shield-ului
    - LCDWIKI_GUI
    - LCDWIKI_KBV
- Processing IDE - versiunea 4.2
  - Processing
  - Pentru programarea Kinect-ului
    - SimpleOpenNI - versiunea pentru Processing 3.5.3 (Funcționează și pentru 4.2)
  - Pentru comunicarea serială cu plăcuța Arduino:
    - Serial

Deoarece Kinect-ul si plăcuța Arduino sunt conectate prin intermediul laptop-ului, vom avea pe lângă `dancing_stickman.ino`, pentru Arduino, și `dancing_stickman.pde`, un program scris în Processing, prin intermediul căruia Kinect-ul poate comunica cu plăcuța folosind USART.

Pentru Arduino nu trebuie să facem nimic diferit, în Processing însă va trebui să accesăm portul serial astfel:

```
import processing.serial.*;
Serial port;

void setup() {
  String portName = Serial.list()[SERIAL_PORT_INDEX];
  port = new Serial(this, portName, BAUD_RATE);
}
```

Pentru `SERIAL_PORT_INDEX` trebuie încercate mai multe valori prin trial and error pentru a descoperi valoarea care corespunde port-ului serial. Majoritatea tutorialelor de pe internet folosesc 0, dar în cazul meu am folosit 2. Dacă setup-ul este corect, apelul funcției `port.write()` din Processing ar trebui să trimită octeți către Arduino pe care să îi recepționeze cu funcția `Serial.read()`.

Dorim ca Kinect-ul să detecteze scheletul user-ului și gesturile acestuia. Pentru a utiliza aceste funcționalități ale Kinect-ului, vom avea nevoie și de matricea de adâncime (depth map), care calculează distanța de la Kinect la tot ceea ce vede utilizând două camere (RGB și IR) și un proiector de raze infraroșii. Aveți aici o explicație cu reprezentare vizuală.

Exemplu de harta de adâncime. Observați faptul ca brațul stâng apare negru, deși este foarte aproape de cameră, deci ne-am aștepta să apară alb. Acest fenomen este cauzat de felul în care Kinect-ul se folosește de razele infraroșii pentru a calcula distanța până la obiecte. Distanța la care Kinect-ul este cel mai eficient este de obicei între 1.5 și 4 metri.

Kinect-ul este inițializat astfel:

```
import SimpleOpenNI.*;
SimpleOpenNI kinect;

void setup() {
  kinect = new SimpleOpenNI(this);
  kinect.enableDepth(); // activează matricea de adâncime
  kinect.enableUser(); // activează recunoașterea utilizatorilor
(recunoașterea corpului)
  kinect.enableHand(); // activează recunoașterea gesturilor
  kinect.startGesture(SimpleOpenNI.GESTURE_WAVE); // activează recunoașterea
gestului WAVE

  /* se pregătește fereasta în care vom afișa matricea de adâncime */
  size(KINECT_WIDTH, KINECT_HEIGHT); // KINECT_WIDTH = 640, KINECT_HEIGHT =
480
  fill(255, 0, 0);
}

void draw() {
  /* actualizează fereastra cu noua matrice de adâncime */
  kinect.update();
  image(kinect.depthImage(), 0, 0);
}
```

Activarea acestor funcționalități vor declanșa si apelul funcțiilor de callback `onNewUser`, `onLostUser`, `onCompletedGesture`, care vor fi folosite pentru etapa de calibrare și lansare a recunoașterii corpului.

Pentru detectarea corpului utilizatorului, Kinect-ul recunoaște 15 puncte cu care formeaza scheletul: cap, gât, doi umeri, două coate, două mâini, trunchi, două șolduri, doi genunchi, două tălpi.

Exemplu de recunoaștere a scheletului peste matricea de adâncime.

Obținerea unor coordonate 2D la aceste puncte este trivială. Folosind biblioteca `SimpleOpenNI` se utilizează funcția `getJointPositionSkeleton` pentru fiecare punct pentru a obtine poziția lui în spațiul 3D, apoi se proiectează la 2D folosind `convertRealWorldToProjective`.

```
PVector getProjectiveJoint(int userID, int jointID) {
  /* Obține coordonatele 3D a punctului de pe schelet */
  PVector joint = new PVector();
  float confidence = kinect.getJointPositionSkeleton(userID, jointID, joint);
// confidence poate fi folosit pentru verificarea preciziei rezultatelor

  /* Convertește la spațiul 2D calculând proiecția */
  PVector projectiveJoint = new PVector();
  kinect.convertRealWorldToProjective(joint, projectiveJoint);

  return projectiveJoint;
}
```

Partea mai complicată la acest proiect este transmiterea celor 15 puncte de la Kinect la Arduino, prin intermediul USART, din următoarele motive:

- Rezoluția ecranului meu LCD este de 320×480;
- Rezoluția camerei Kinect-ului este de 640×480;
- USART trimite informația serial, octet cu octet, deci putem doar sa trimitem pe rând valori cuprinse între 0 și 255.

Acest fapt înseamnă că este nevoie să fie schițat un protocol de comunicație minimal care să fie folosit de cele două dispozitive. Partea bună este totuși că această comunicație se poate produce doar într-un singur sens, de la Kinect la Arduino.

Soluția la care am ajuns a fost utilizarea funcției `map(value, fromLow, fromHigh, toLow, toHigh)`, existentă în ambele limbaje de programare folosite, pentru a codifica coordonatele (x, y)

ale punctelor astfel:

```
// Processing - Trimitere de la Kinect
void sendCoordinateAxis(float value, int lowerBound, int upperBound) {
  int intVal = floor(between(value, lowerBound, upperBound));
  int asByte = (int)map(intVal, lowerBound, upperBound, 0, 255);
  port.write(asByte);
}
```

Pentru recepție de pe Arduino:

```
// Arduino - Recepție pe plăcuță
int readCoordinateX() {
  byte xAsByte = Serial.read();
  int x = (int)map(xAsByte, 0, 255, 0, width);
  Serial.println(x);
  return x;
}

int readCoordinateY() {
  byte yAsByte = Serial.read();
  int y = (int)map(yAsByte, 0, 255, 0, height);
  Serial.println(y);
  return y;
}
```

Această tehnică va atrage mici pierderi de informație la maparea pe LCD, însă acesta sunt rareori observabile. Avantajul masiv pe care îl oferă este însă acela că minimizează numărul de octeți trimiși într-un mesaj(31):

- 15 octeți pentru coordonatele X;
- 15 octeți pentru coordonatele Y;
- un octet pentru octet-ul de trigger.

Această soluție este cea mai bună, deoarece permite un refresh rate mai mare.

O altă soluție pe care am luat-o în considerare ar fi fost transmiterea a doi octeți pentru fiecare coordonată, însă aceasta ar adăuga complexitate programului și, cel mai important, ar dubla dimensiunea mesajului, îngreunând treaba plăcuței Arduino.

Mesajele plăcuței vor fi mereu de 31 de octeți dintre care primul va fi folosit pentru recunoașterea mesajului:

- BYTE_RECOGNISED - Kinect-ul a detectat jucătorul;
- BYTE_START_PLAYING - Kinect-ul a detectat gestul de wave și semnalează începerea jocului;
- BYTE_LOST - Kinect-ul nu mai detectează jucătorul;
- BYTE_START - Kinect-ul trimite coordonatele curente ale scheletului utilizatorului;
- BYTE_NOT_RECOGNISED - Valore dată coordonatelor punctelor de pe schelet care nu sunt vizibile pentru Kinect.

Codul este scris astfel încât coordonatele punctelor de pe schelet sunt considerate în aceeași ordine de ambele dispozitive, de la cap (SKEL_HEAD) la tălpi (SKEL_LEFT_FOOT, SKEL_RIGHT_FOOT);

Pentru desenarea corpului pe display-ul LCD se vor folosi din biblioteca LCDWIKI_GUI funcțiile:

- `Fill_Circle(int16_t x, int16_t y, int16_t radius);`
- `Fill_Triangle(int16_t x0, int16_t y0, int16_t x1, int16_t y1,int16_t x2, int16_t y2);`
- `Draw_Line(int16_t x1, int16_t y1, int16_t x2, int16_t y2).`

Acestea sunt utilizate în funcții wrapper pentru a desena corpul stickman-ului care va imita utilizatorul:

```
void drawHead() {
  lcd.Fill_Circle(skeletonPoints[SKEL_HEAD].x, skeletonPoints[SKEL_HEAD].y,
25);
}
```

```
void drawBodyPart(int firstIndex, int secondIndex, int thirdIndex) {
  PVector p1 = skeletonPoints[firstIndex];
  PVector p2 = skeletonPoints[secondIndex];
  PVector p3 = skeletonPoints[thirdIndex];

  lcd.Fill_Triangle(p1.x, p1.y, p2.x, p2.y, p3.x, p3.y);
}
```

```
void drawLimb(int firstIndex, int secondIndex) {
  PVector p1 = skeletonPoints[firstIndex];
  if (p1.x == BYTE_NOT_DETECTED && p1.y == BYTE_NOT_DETECTED) {
    return;
  }

  PVector p2 = skeletonPoints[secondIndex];
  if (p2.x == BYTE_NOT_DETECTED && p2.y == BYTE_NOT_DETECTED) {
    return;
  }

  lcd.Draw_Line(p1.x, p1.y, p2.x, p2.y);
  /* Draw around the point for thicker lines */
  for (int i = 1; i <= 2; i++) {
    lcd.Draw_Line(p1.x + i, p1.y, p2.x + i, p2.y);
    lcd.Draw_Line(p1.x - i, p1.y, p2.x - i, p2.y);
    lcd.Draw_Line(p1.x, p1.y + i, p2.x, p2.y + i);
    lcd.Draw_Line(p1.x, p1.y - i, p2.x, p2.y - i);
  }
}
```

Pentru a desena mai gros liniile dintre punctele de pe schelet, desenăm mai multe linii cu punctele limită adiacente cu sursa. Totuși, este important de ținut minte că această tehnică nu trebuie folosită excesiv, deoarece poate îngreuna foarte mult plăcuța, rezultând într-un aspect neplăcut și timpi de

așteptare excesivi.

```
void drawSkeleton() {
  lcd.Fill_Screen(BLACK);
  drawHead();

  /* Draw a circle for each skeleton point */
  for (int i = 1; i < NUM_SKEL_POINTS; i++) {
    PVector p = skeletonPoints[i];
    lcd.Fill_Circle(p.x, p.y, 4);
  }

  drawLimb(SKEL_HEAD, SKEL_NECK);

  drawLimb(SKEL_NECK, SKEL_LEFT_SHOULDER);
  drawLimb(SKEL_LEFT_SHOULDER, SKEL_LEFT_ELBOW);
  drawLimb(SKEL_LEFT_ELBOW, SKEL_LEFT_HAND);

  drawLimb(SKEL_NECK, SKEL_RIGHT_SHOULDER);
  drawLimb(SKEL_RIGHT_SHOULDER, SKEL_RIGHT_ELBOW);
  drawLimb(SKEL_RIGHT_ELBOW, SKEL_RIGHT_HAND);

  drawLimb(SKEL_LEFT_SHOULDER, SKEL_TORSO);
  drawLimb(SKEL_RIGHT_SHOULDER, SKEL_TORSO);

  drawLimb(SKEL_TORSO, SKEL_LEFT_HIP);
  drawLimb(SKEL_LEFT_HIP, SKEL_LEFT_KNEE);
  drawLimb(SKEL_LEFT_KNEE, SKEL_LEFT_FOOT);

  drawLimb(SKEL_TORSO, SKEL_RIGHT_HIP);
  drawLimb(SKEL_RIGHT_HIP, SKEL_RIGHT_KNEE);
  drawLimb(SKEL_RIGHT_KNEE, SKEL_RIGHT_FOOT);

  drawLimb(SKEL_LEFT_HIP, SKEL_RIGHT_HIP);

  drawBodyPart(SKEL_LEFT_SHOULDER, SKEL_RIGHT_SHOULDER, SKEL_TORSO);
  drawBodyPart(SKEL_LEFT_HIP, SKEL_RIGHT_HIP, SKEL_TORSO);
}
```

Ultima problemă cu care m-am confruntat(care din păcate este cea mai evidentă) a fost refresh rate-ul foarte mic al display-ului LCD. Mai mult decât atât, plăcuța nu poate ține pasul cu mesajele trimise de Kinect. Din acest motiv nu am putut obține actualizare în timp real, folosind în schimb un timer pentru a limita intervalul de timp în care se trimit coordonate de schelet. O soluție posibilă ar fi putut să fie utilizarea tehnicii de double buffer și a algoritmului Bresenham pentru desenarea liniilor, însă memoria limitată a plăcuței nu a făcut posibilă alocarea unui buffer.

Rezultate Obţinute

Concluzii

Ma bucur că am descoperit cum funcționează tehnologiile folosite de Kinect (depth image, skeleton recognition) și am reușit să creez un proiect fun și relativ unic. Sper că acest proiect va motiva viitorii studenți să încerce și ei includerea unui Kinect în proiectul lor. Este mult mai ușor decât pare, iar eu am încercat în această documentație să menționez toate blocajele pe care le-am avut în timpul fazei de setup pentru a trece mult mai repede la partea creativă. Singurul regret pe care îl am este că display-ul meu nu a avut un refresh rate suficient de mare încât să pot reda imagini în timp real.

Download

- Proiect Github
- Driver Kinect pentru PC
- Processing IDE
  - SimpleOpenNI
- Arduino IDE
  - LCDWIKI_GUI și LCDWIKI_KBV

Jurnal

Am deci să folosesc această rubrică pentru a scoate în evidență blocajele pe care le-am avut și ce mi-aș fi dorit să știu înainte de a începe să lucrez la acest proiect, în speranța ca viitori studenți care folosesc Kinect pentru proiectul lor să treacă mult mai repede peste problemele de configurare. Dacă doriți să creați un proiect folosind Kinect și Arduino, următoarele idei s-ar putea să vă salveze destul de mult timp.

Setup Kinect:

- Pentru versiunile mai vechi de Kinect veți avea nevoie de power adapter. Cele mai noi pot fi direct conectate la calculator (e.g. Kinect for Windows);
- Dacă folosiți un power adapter, acesta trebuie obligatoriu conectat la laptop printr-un port USB 3.0.
- În funcție de calculator, este sau nu posibil ca Kinect-ul să pornească imediat (verificați LED-ul verde din față). Dacă nu, instalați driver-ul pentru Kinect (link la Downloads).
- La momentul scrierii, nu există o librărie open-source de Kinect pentru calculatoarele cu arhitectură ARM, care să ofere toate funcționalitățile disponibile. Recomand folosirea Windows, Linux sau OSX cu Intel.

Programare Kinect și comunicare cu Arduino:

- În literatura de specialitate se folosesc în mare parte versiuni de simple-openni dinainte de 1.96, care acum sunt outdated;
- Chiar dacă versiunile de SimpleOpenNI spun că sunt pentru o versiune specifică de Processing, este posibil ca totuși să puteți folosi și versiuni mai noi de processing (e.g. SimpleOpenNI for Processing 3.5.3 cu Processing 4.2);
- Atunci când stabiliți conexiunea serială între calculator și Arduino, verificați în codul de Processing dacă portul serial folosit este cel corect.
  - Toate tutorialele spun că numele portului care trebuie folosit este Serial.list()[0], dar în cazul meu acest port era ocupat de căștile cu Bluetooth.
  - Pentru mine, Serial.list()[2] a fost numele portului serial.

Alte probleme întâmpinate de mine:

- Display-ul LCD nu are un refresh rate suficient de mare pentru a capta imagini în timp real. La momentul scrierii display-ul afișeaza poziția utilizatorului o dată la 1.6 secunde.
- Deoarece comunicarea seriala este bazata octet pe octet, va fi nevoie sa folosesc valori intre 0 si 255 si sa imi creez minimal un protocol de comunicație.
- Memoria limitată a plăcuței Arduino face imposibilă utilizarea tehnicii double buffer pentru un display de 320×420.

Bibliografie/Resurse

Resurse Software:

- http://www.lcdwiki.com/3.5inch_Arduino_Display-UNO
- https://www.instructables.com/Kinect-controls-Arduino-wired-Servos-using-Visual-/
- https://github.com/gitcnd/LCDWIKI_GUI/blob/master/LCDWIKI_GUI.h
- https://github.com/totovr/SimpleOpenNI

Resurse Hardware:

- https://www.instructables.com/Kinect-controls-Arduino-wired-Servos-using-Visual-/
- https://www.aranacorp.com/en/using-a-tft-lcd-shield-with-arduino/
- https://www.youtube.com/watch?v=NhpXJlM7-7E&ab_channel=ZinTechIdeas
- http://www.lcdwiki.com/3.5inch_Arduino_Display-UNO#Product_Video

Resurse Kinect:

- https://www.academia.edu/18712369/Arduino_and_Kinect_Projects

Export to PDF