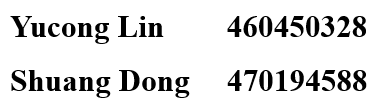
**3D Game Machine**

**To The Hell With Johnny**

****

# Abstract

In this milestone we designed and built a 3d game machine called *To Hell with Johnny*. This game machine is a human interaction game machine which can detect the movement of human and transfer data to the Arduino Due, and then to control the character to play this game. This game is displayed on an LCD and can transfer instruction from Android phones through a Bluetooth module, which means remote control to some extent. This game machine contains three different modes with different complexity and could run perfectly generally.

# Aim/Problem:

Game is always an attractive and popular field with great potential, that’s why we choose a game machine project. While great games do not only depend on good ideas, but also require solid programming ability. In the following project we tried to optimize our program and debug on code and peripherals.

# Design of hardware

For this project, the whole system includes a piece of Arduino, a 3D accelerator, a piece of LCD and a module of Bluetooth. The overall structure is show as figure.1 and each part is introduced respectively below.

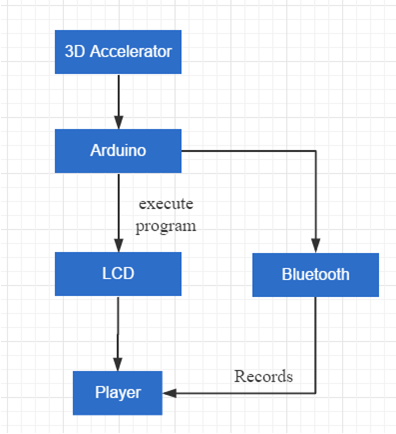


Fig.1 overall structure

## Arduino Due

We used the Arduino Due with the chip SAM3X8E, which is an open-source electronics platform and important for everyone to use, especially in embedded system. In this project we connected all the peripherals, including LCD, accelerometer and Bluetooth on it to make them work together. We can see all pins of Arduino Due as below in Figure.2.

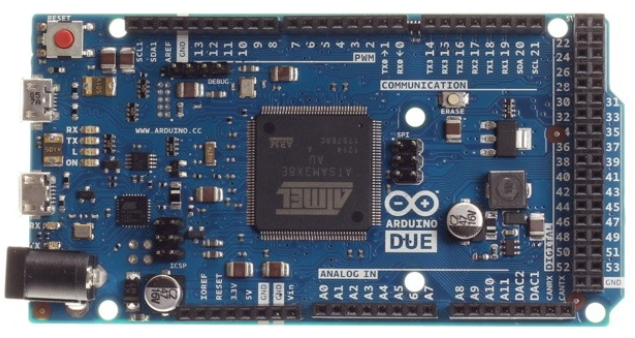


Fig.2 Arduino Due SAM3X8E

## LCD

We used a widely-used LCD, Nokia 5110, which contains 8 pins as shown below in Figure.3 with the size of is 48\*84.

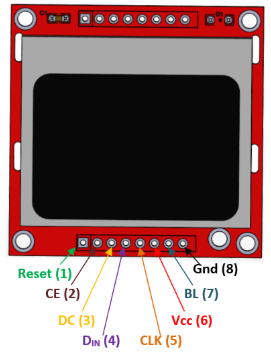


Fig.3.Nokia 5110 LCD

It usually works under 3.3V, which mean the VCC (pin 6) should be connected to 3.3V power supply on Arduino Due, while the BL (pin 7), background light, should be connected to 5.5V power supply and GND (pin 8) to GND on Arduino, without doubt. These are the power supply connection. RST (pin 1) is the reset pin, which can reset the module. CE means chip enable and is made low to select this LCD when we have more than one peripherals on Arduino. While DC means data/command to switch between data mode (high) and command mode(low). The input is Din (pin 4) and clock is CLK (pin 5). Pins from 1 to 5 were connected to the pins from 3 to 7 on Arduino, respectively as we set up in the code.

We used *Adafruit\_PCD8544* and *Adafruit\_GFX* libraries to draw the icons of the character that we were going to control in the game and printed the words at start and end of game.

## 3d Accelerometer

Model of this accelerometer is MPU6050, the sensing range of angular velocity that can be detected is ±250, ±500, ±1000 and ±2000 o/sec and it can track the fast and slow move accurately, working under 5V power supply.

We used the official library support *I2Cdev* to configure for this module to communication and read data. We used 4 of 8 pins, VCC, GND, SLA, SDA, as shown below in Figure.4. Its SCL and SDA pins were connected to SCL (pin 21) and SDA (pin 20) on Arduino Due.



Fig.4 MPU6050

Initially MPU 6050 output the 6 values, acceleration and angular velocity of 3 directions. In this project we just need to use the value of acceleration of x direction to decide the movement direction of the character, and when the acceleration is large enough, the character will move faster, that’s what we want when designing this game.

So, we used MPU6050 library to output pitch and roll angle of x direction and that was enough for us.

## Bluetooth

We used Bluetooth HC-05 to achieve remote control to this game. We firstly configure it into AT mode by powering it while pressing the black button on it.

In AT mode (Baud rate is 38400) we can rename its name and made it into slave mode so that we could search it through Android phones.

After connection we could receive data from phones to firstly select the mode of game, working under 5V power supply (Baud rate is 9600 when communicating).

We just used 4 pins to connect to Arduino Due. RXD to TX1 (pin 18) and TXD to RX1 (pin 19) respectively.

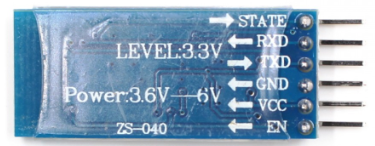


Fig.5 HC-05

# Design of Software

Because in our game machine, we provide three level to player to enjoy the game, so there must be an interface for players to choose the level. After initializing the game interface, there should be a subroutine to choose level. We realize it by the Bluetooth module so actually it is a process to read the data from player via the connection of Bluetooth and phone.

After finishing the selection, the game operating in steps. Firstly, calculate the movement of bricks and character. The bricks are generated randomly in terms of their location and kind. In order to move the character, the accelerometer measures the rotation angle of user and convert it into the change of coordinate of character. Thirdly, refresh the display on LCD to reflect all the movement to player. In the end of the game loop is to determine whether the character is on a brick and on which kind of brick if it is, then operate the correspond movement of character.

There are two ways to over the game: 1. the character touch the top or the bottom of the boundary; 2. the character touch the harsh brick. Whenever those judgement condition achieved, the game is over immediately and display the corresponding interface.

The flowchart of the whole software design is shown in figure.6.

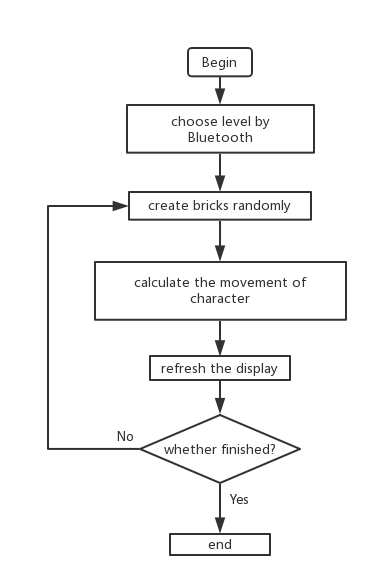


Fig.6 the whole software design

# How the design works

Android phones could be used to send data 1, 2, or 3 to select the mode of game in the menu page after successful connection.

Then we could start turning the 3-d accelerometer to control the character to move to left or right. The value of acceleration of x direction could be more stable and precious because of usage of *MPU6050* library.

We also have the termination condition when the character reaches the upper or the bottom, or when it moves onto the harsh block which is one of the three kinds of blocks.

When it is about to terminate, the screen will turn to next page showing GAME OVER and we need to press the RST button on Arduino to restart the game. Then this can be whole game.

# Trouble-shooting:

We meet several problems during our milestone. Fortunately, we solve them validly in the end. The main problems are illustrated below.

## The tracking of the character’s position

In our game, the movement of character depend on whether it is on a brick and if it is on a brick, what kind is that brick. We create a subroutine named whether\_on\_brick(); to check that whether the character is on the brick and return the kind of the brick which the character on. The algorithm of this subroutine is calculating the difference between the coordinate of the character and the top of the brick. If the difference equal to the height of the character, it means that the character is on the brick and the flag of on the brick is set.

In the test, we find that sometime this subroutine works well, but sometime the character fall down through the brick, even it should stop on the brick. In the first we are really confused because the method of checking whether on bricks is all the same but sometimes it works while sometimes not. We even print out the value of the flag in serial port in order to determine the operation of this function step by step. However, we find that the flag does not change when it should be.

Ultimately, we print out the coordinate of the character and bricks, we find the although it looks like that the character fall down from the top to the bottom, the coordination of it does not change consistently. The falling speed of the character is not 1 pixel per refresh time. The refresh of the LCD is so fast and that is why it looks like that the character once is on the brick exactly without overlapping but in fact the difference between the coordinate of the character and the brick is not equal to the height of the character.

After determining why does it not work, we solve this problem by figuring out the range of that the difference between the coordinate of the character and the brick could be, then set the flag when this condition happens. The subroutine works normally after this modifying.

## The overflow of the IMU

We try to use IMU module MPU6050 to determine the rotation angle and invert it into the movement of the character. In this way, the user could play the game using this senor instead of buttons and rocking bar.

When we finish the function of reading the rotation angle using the library given by the manufacturer and changing it into the change of the character, we connect MPU6050 to Arduino Due. Everything goes smoothly when debugging, but later find out that it wouldn't send new data for a while because the character does move left and right when we rotate the sensor.

The library given by the manufacturer has some guided word such as ‘connect failed’. In the serial port, the Arduino program reported a ‘FIFO overflow’.

We do some research on the internet and the solution is found. The last number in the MPU6050 library ‘MPU6050\_6Axis\_MotionApps20.h’ 305 lines is related to the frequency of IMU topic on the ros platform, so it depends on the performance of the hardware instead of being a constant. By testing several times, we figure out the suitable value of that number then the MPU6050 can operate normally.

1. Improvements

Because of the short of time, there are so many that could be improved in our milestone.

## The display of the character

As I mentioned in the chapter 4, the difference between the coordinate of character and bricks does not equal to the height of the character. We change the judging criteria that if the difference is within a certain range, it means that the character is on a brick. Whenever the software determines that the character is on a normal brick, the man will move up with that brick, looking like that the character is standing on the brick. However, after debugging the subroutine, the character hangs on the brick sometimes, shown in figure.7.

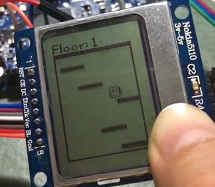


fig.7 when the character hang on the brick

This is not hard to be modified. Setting a more detailed determine criteria will help, like that when the difference equal to the height exactly, the character move up directly with the brick while the coordinate will be added or subtracted appropriately in other times. So that the display of character could be modified.

## The interaction with phone using Bluetooth

In our design of the milestone, we have three level to play. Because we do not have buttons, so we design to use our phone to choose the mode of the game.

We utilize Bluetooth module HC-05 to connect with the Arduino due to the phone and send data to the Arduino. However, the function of this Bluetooth is too simple that it seems that it is more complex to use this module. In the future we decide to realize more function with this Bluetooth module, such as upload the score of the game to the phone and store in a list of top ten marks or control the phone to send some message by Twitter or Facebook. We even want to use two Arduino due and two modules of Bluetooth to realize the communication between them.

## Another way to control the game with MPU6050

We use MPU6050 to control the movement of the characters by reading and converting the rotation angle. But we still have another plan to build this game that using IMU MPU6050 to track the movements of the player to control the character, i.e. changing the coordinate of character by reading the change of coordinate of player but not the rotation angle.

In this way the players could join their whole body into the game instead of just rotating Arduino. They may get more fun from the game in this way and the game can also help them do some sport in a more relaxed and pleasant way.

MPU6050 is just an IMU, so we need to calculate the double integral of the reading from MPU6050 if we want to get the trace of the movement of players. However, the error of double integral is non-negligible, let alone there is offset of the data measured by MPU6050. Fortunately, after doing some research we learn something about Kalman filter, an algorithm that can reducing the error of this calculation.

Kalman Filter is an optimal recursive data processing algorithm.

It keeps track of the estimated state of the system and the variance or uncertainty of the estimate. The estimate is updated using a state transition model and measurements.

Its principle is shown below in figure.8. denotes the estimate of the system's state at time stepbefore themeasurementhas been taken into account; is the corresponding uncertainty.

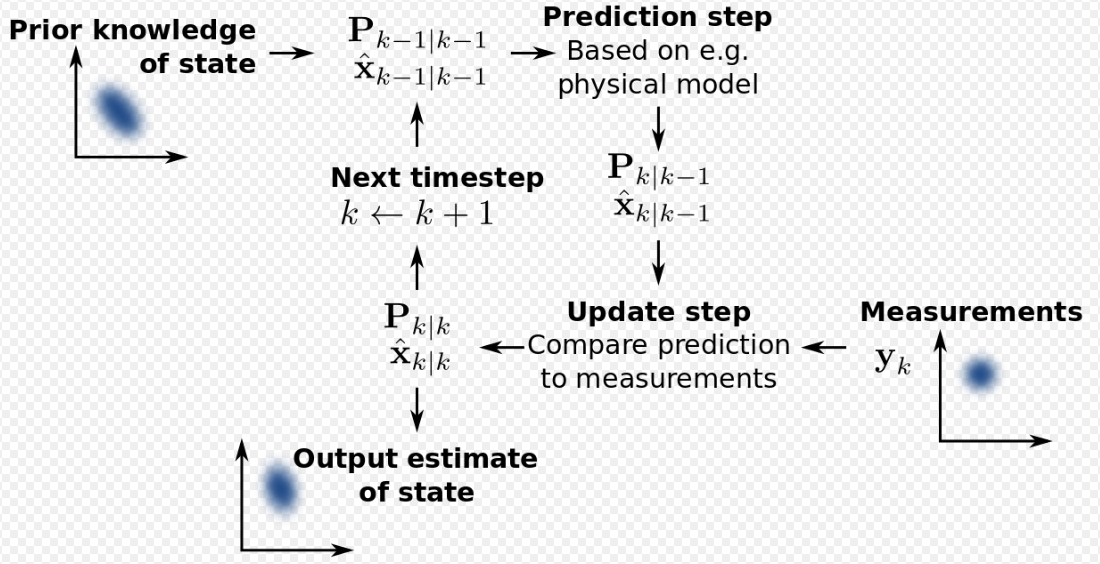


Fig.8 Principle of Kalman Filter

# Cost

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Quantity | Price ($) | Total |
| Arduino Due | 1 | 69.93 | 69.93 |
| Nokia5110LCD | 1 | 14.56 | 14.56 |
| MPU6050 3d accelerometer | 1 | 13.01 | 13.01 |
| HC05 Bluetooth | 1 | 10.37 | 10.37 |
| Total |  |  | 107.87 |

# Conclusion

In conclusion, we tried to achieve our goal of building a 3d game machine using an Arduino Due and play it on LCD, and we successfully did it. We have to admit it that there were still a lot of challenges even though this game looked simple and easy to operate, like stabilizing the value of 3d accelerometer, or achieving many interesting functions that we designed at first or trying to make the connection of Bluetooth stable.

This milestone did a lot of help because we build it by own hands step by step, including coding, finding and understanding libraries. Furthermore, this project also improved our abilities on teamwork and self-discipline, which can in turn help us overcome coming difficulties in the future.

**Reference**

[1] Arduino.cc. (2018). Arduino Reference. [online] Available at: <https://www.arduino.cc/reference/en/>

[2] Arduino.cc. (2018). Arduino - PinMappingSAM3X. [online] Available at: https://www.arduino.cc/en/Hacking/PinMappingSAM3X

[3] Ww1.microchip.com. (2018). [online] Available at: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-11057-32-bit-Cortex-M3-Microcontroller-SAM3X-SAM3A\_Datasheet.pdf

[4] How to clear FIFO buffer on MPU6050. [online] Arduino Stack Exchange. Available at: https://arduino.stackexchange.com/questions/10308/how-to-clear-fifo-buffer-on-mpu6050

[5] GitHub. (2018). MPU6050 connected to, FIFO overflow [online] Available at: https://github.com/jrowberg/i2cdevlib/issues/87

[6] YouTube. (2018). Arduino Tutorial: Menu on a Nokia 5110 LCD Display Tutorial. [online] Available at: https://www.youtube.com/watch?v=VtZvf5T98FI

[7] GitHub. (2018). adafruit/Adafruit-PCD8544-Nokia-5110-LCD-library. [online] Available at: https://github.com/adafruit/Adafruit-PCD8544-Nokia-5110-LCD-library

[8] Maker Pro. (2018). How to Interface Arduino and the MPU 6050 Sensor | Arduino. [online] Available at: https://maker.pro/arduino/tutorial/how-to-interface-arduino-and-the-mpu-6050-sensor

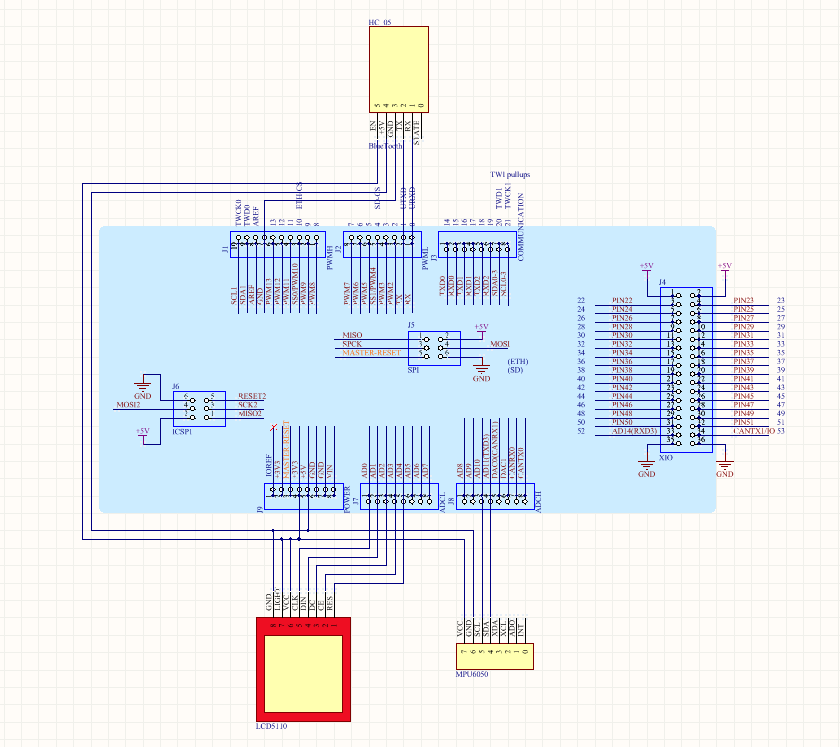
[9] Forum.arduino.cc. (2018). [Solved] MPU6050 and DUE. [online] Available at: https://forum.arduino.cc/index.php?topic=162727.0

[10] YouTube. (2018). How to interface Arduino Due with MPU6050: Arduino Tutorial Tamil. [online] Available at: https://www.youtube.com/watch?v=ylE1Ewyflh0

[11] HowToMechatronics. (2018). Arduino and HC-05 Bluetooth Module Tutorial - HowToMechatronics. [online] Available at: https://howtomechatronics.com/tutorials/arduino/arduino-and-hc-05-bluetooth-module-tutorial/

[12] Arduino-info.wikispaces.com. (2018). arduino-info - BlueTooth-HC05-HC06-Modules-How-To. [online] Available at: <https://arduino-info.wikispaces.com/BlueTooth-HC05-HC06-Modules-How-To>

[13]HowToMechatronics. (2018). Arduino and HC-05 Bluetooth Module Tutorial - HowToMechatronics. [online] Available at: https://howtomechatronics.com/tutorials/arduino/arduino-and-hc-05-bluetooth-module-tutorial/ **Appendix1**

****

**Appendix 2**

//This code is used for the ELEC9607 project in 2018

//3D game machine -- To Hell with Johnny

//This is a free software and we will be happy if it can help you.

//Actually the code is little messy and clumsy. orz...

//Created by Charles & Shuang on 2018/6/4.

//Copyright © 2018 Charles & Shuang All rights reserved.

#include <SPI.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_PCD8544.h>

#include "Wire.h"

#include "I2Cdev.h"

#include "MPU6050\_6Axis\_MotionApps20.h"

#define HC05 Serial1

MPU6050 accelgyro;

int16\_t ax, ay, az;

int16\_t gx, gy, gz;

//The kind of brick

#define normal 0

#define unstable 1

#define harsh 2

#define broken 3

Adafruit\_PCD8544 display = Adafruit\_PCD8544(7, 6, 5, 4, 3);

static const unsigned char PROGMEM floor\_bmp[] =

{

B11110100, B00000000, B00000000,

B10000100, B00000000, B00000000,

B10000100, B00000000, B00000000,

B11110100, B11000110, B01010100,

B10000101, B00101001, B01100000,

B10000101, B00101001, B01000100,

B10000100, B11000110, B01000000,

B00000000, B00000000, B00000000,

};

static const unsigned char PROGMEM man\_bmp[] =

{

B01111100,

B10000010,

B10101010,

B10101010,

B10000010,

B10000010,

B10101010,

B01010100,

};

static const unsigned char PROGMEM normal\_bmp[] =

{

B11111111, B11111110,

B11111111, B11111110,

B11111111, B11111110,

};

static const unsigned char PROGMEM harsh\_bmp[] =

{

B01000100, B01000100,

B10101010, B10101010,

B11111111, B11111110,

};

static const unsigned char PROGMEM unstable\_bmp[] =

{

B11111111, B11111110,

B10000000, B00000010,

B11111111, B11111110,

};

static const unsigned char PROGMEM broken\_bmp[] =

{

B11000000, B00000110,

B11000000, B00000110,

B11000000, B00000110,

};

int brick[5] = { 0, 2, 0, 0, 1 };

int xb0 = 17; //the coordinate of the bricks

int yb0 = 48;

int xb1 = 3;

int yb1 = 64;

int xb2 = 12;

int yb2 = 79;

int xb3 = 34;

int yb3 = 93;

int xb4 = 22;

int yb4 = 111;

int refreshtime = 150;

int count\_broken = 0; //time on unstable brick

int xm = 21; //the coordinate of the characters

int ym = 41;

int score = 0;

int on\_brick = 0;//whether on bricks

int start = 0;

int death = 0;

int the\_brick = 0;

void setup()

{

Serial.begin(9600);

display.begin();

//init done

//you can change the contrast around to adapt the display

//for the best viewing!

display.setContrast(53);

display.clearDisplay();

display.setRotation(1);

on\_brick = 0;

Wire.begin();

accelgyro.initialize();

Serial.println(accelgyro.testConnection() ? "MPU6050 connection successful" : "MPU6050 connection failed");

HC05.begin(38400);

}

void loop()

{

if (start == 0)

{

gamestart();

if (HC05.available() > 0)

{

char data = HC05.read();

if (data == '1')

{

refreshtime = 300;

start = 1;

}

if (data == '2')

{

refreshtime = 150;

start = 1;

}

if (data == '3')

{

refreshtime = 77;

start = 1;

}

}

}

accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

if (start == 1 && death == 0)

{

initiate\_game();

whetheronbrick();

brick\_happen();

movement\_of\_man();

display.drawBitmap(xm, ym, man\_bmp, 7, 8, BLACK);

display.display();

determine\_death();

delay(refreshtime);

on\_brick = 0;

Serial.println(ax);

}

if (death != 0)

{

gameover();

}

}

void initiate\_game()

{

display.drawBitmap(0, 0, floor\_bmp, 24, 8, BLACK);

display.drawRect(0, 8, 48, 76, BLACK);

display.drawRect(0, 9, 48, 76, BLACK);

display.drawRect(0, 10, 48, 76, BLACK);

display.setTextSize(1);

display.setTextColor(BLACK);

display.setCursor(24, 0);

display.println(score);

display.display();

}

//whether on brick

void whetheronbrick()

{

if (xm >= (xb0 - 5) && xm <= (xb0 + 13) && (yb0 - ym) <= 9 && (yb0 - ym) >= 6)

{

on\_brick = 1;

the\_brick = 0;

}

if (xm >= (xb1 - 5) && xm <= (xb1 + 13) && (yb1 - ym) <= 9 && (yb1 - ym) >= 6)

{

on\_brick = 1;

the\_brick = 1;

}

if (xm >= (xb2 - 5) && xm <= (xb2 + 13) && (yb2 - ym) <= 9 && (yb2 - ym) >= 6)

{

on\_brick = 1;

the\_brick = 2;

}

if (xm >= (xb3 - 5) && xm <= (xb3 + 13) && (yb3 - ym) <= 9 && (yb3 - ym) >= 6)

{

on\_brick = 1;

the\_brick = 3;

}

if (xm >= (xb4 - 5) && xm <= (xb4 + 13) && (yb4 - ym) <= 9 && (yb4 - ym) >= 6)

{

on\_brick = 1;

the\_brick = 4;

}

}

//generate and move bricks

void brick\_happen()

{

display.clearDisplay();

initiate\_game();

if (yb0 >= 9)

{

switch (brick[0])

{

case 0: display.drawBitmap(xb0, yb0, normal\_bmp, 15, 3, BLACK);

case 1: display.drawBitmap(xb0, yb0, unstable\_bmp, 15, 3, BLACK);

case 2: display.drawBitmap(xb0, yb0, harsh\_bmp, 15, 3, BLACK);

case 3: display.drawBitmap(xb0, yb0, broken\_bmp, 15, 3, BLACK);

}

yb0--;

}

else

{

yb0 = 83;

xb0 = rand() % 33;

brick[0] = rand() % 3;

if (brick[2] | brick[3] | brick[4] != 0)

brick[0] = 0;

score++;

}

if (yb1 >= 9)

{

switch (brick[1])

{

case 0: display.drawBitmap(xb1, yb1, normal\_bmp, 15, 3, BLACK);

case 1: display.drawBitmap(xb1, yb1, unstable\_bmp, 15, 3, BLACK);

case 2: display.drawBitmap(xb1, yb1, harsh\_bmp, 15, 3, BLACK);

case 3: display.drawBitmap(xb1, yb1, broken\_bmp, 15, 3, BLACK);

}

yb1--;

}

else

{

yb1 = 83;

xb1 = rand() % 33;

brick[1] = rand() % 3;

if (brick[0] | brick[3] | brick[4] != 0)

brick[1] = 0;

score++;

}

if (yb2 >= 9)

{

switch (brick[2])

{

case 0: display.drawBitmap(xb2, yb2, normal\_bmp, 15, 3, BLACK);

case 1: display.drawBitmap(xb2, yb2, unstable\_bmp, 15, 3, BLACK);

case 2: display.drawBitmap(xb2, yb2, harsh\_bmp, 15, 3, BLACK);

case 3: display.drawBitmap(xb2, yb2, broken\_bmp, 15, 3, BLACK);

}

yb2--;

}

else

{

yb2 = 83;

xb2 = rand() % 33;

brick[2] = rand() % 3;

score++;

if (brick[0] | brick[1] | brick[4] != 0)

brick[2] = 0;

}

if (yb3 >= 9)

{

switch (brick[3])

{

case 0: display.drawBitmap(xb3, yb3, normal\_bmp, 15, 3, BLACK);

case 1: display.drawBitmap(xb3, yb3, unstable\_bmp, 15, 3, BLACK);

case 2: display.drawBitmap(xb3, yb3, harsh\_bmp, 15, 3, BLACK);

case 3: display.drawBitmap(xb3, yb3, broken\_bmp, 15, 3, BLACK);

}

yb3--;

}

else

{

yb3 = 83;

xb3 = rand() % 33;

brick[3] = rand() % 3;

score++;

if (brick[0] | brick[1] | brick[2] != 0)

brick[3] = 0;

}

if (yb4 >= 9)

{

switch (brick[4])

{

case 0: display.drawBitmap(xb4, yb4, normal\_bmp, 15, 3, BLACK);

case 1: display.drawBitmap(xb4, yb4, unstable\_bmp, 15, 3, BLACK);

case 2: display.drawBitmap(xb4, yb4, harsh\_bmp, 15, 3, BLACK);

case 3: display.drawBitmap(xb4, yb4, broken\_bmp, 15, 3, BLACK);

}

yb4--;

}

else

{

yb4 = 83;

xb4 = rand() % 33;

brick[4] = rand() % 3;

score++;

if (brick[1] | brick[2] | brick[3] != 0)

brick[4] = 0;

}

}

//movement of man

int movement\_of\_man()

{

if (ax >= 5555)

{

xm = xm - 2;

}

if (ax <= 5555 && ax >= 0)

{

xm = xm - 1;

}

if (ax <= -5555)

{

xm = xm + 2;

}

if (ax >= -5555 && ax <= 0)

{

xm = xm + 1;

}

if (on\_brick) //if on bricks

{

switch (brick[the\_brick])

{

case 0:

{

ym--;

break;

}

case 1:

{

if (count\_broken <= 3)

{

ym--;

count\_broken++;

}

else

{

on\_brick = 0;

brick[the\_brick] = 3;

count\_broken = 0;

}

break;

}

}

}

else

{

ym = ym + 3;

}

if (xm <= 1)

{

xm = 1;

}

if (xm >= 40)

{

xm = 40;

}

}

int determine\_death()

{

if (ym <= 9 || ym >= 78 || brick[the\_brick] == 2)

{

death = 1;

display.clearDisplay();

score = 0;

}

}

void gamestart()

{

display.setTextSize(1);

display.setTextColor(BLACK);

display.setCursor(3, 8);

display.println("TO HELL");

display.setCursor(12, 16);

display.println("WITH");

display.setCursor(6, 24);

display.println("JOHNNY");

display.setTextSize(2);

display.setTextColor(BLACK);

display.setTextSize(1);

display.setTextColor(BLACK);

display.drawBitmap(20, 36, man\_bmp, 7, 8, BLACK);

display.setCursor(0, 49);

display.println("1.EASY");

display.setCursor(0, 58);

display.println("2.NORMAL");

display.setCursor(0, 67);

display.println("3.HARD");

display.display();

}

void gameover()

{

display.setTextSize(2);

display.setTextColor(BLACK);

display.setCursor(0, 16);

display.println("GAME");

display.setTextSize(2);

display.setTextColor(BLACK);

display.setCursor(0, 32);

display.println("OVER");

display.display();

}