```
#include <mbed.h>
#include <MMA7455.h>
#include <LM75B.h>
#include <display.h>

//Declare output object for LED1
DigitalOut led1(LED1);

// Initialise Joystick
typedef enum {JLT = 0, JRT, JUP, JDN, JCR} btnId_t;
static DigitalIn jsBtns[] = {P5_0, P5_4, P5_2, P5_1, P5_3}; // LFT, RGHT, UP, DWN, CTR
bool jsPrsdAndRlsd(btnId_t b);

//Object to manage the accelerometer
MMA7455 acc(P0_27, P0_28);
bool accInit(MMA7455& acc); //prototype of init routine
int32_t accVal[3];

//Input object for the potentiometer
AnalogIn pot(p15);
float potVal = 0.0;

Display *screen = Display::theDisplay();
//This is how you call a static method of class Display
//Returns a pointer to an object that manages the display screen

//Timer interrupt and handler
void timerHandler(); //prototype of handler function
int tickCt = 0;
```

```
	/**************************************************************************
**********/
			/***********************************STRUCTS AND
ENUMS************************************/

	/**************************************************************************
**********/
				typedef struct{
					int x, y;
					int vx, vy;
					int colour;
				} ball_t;

				typedef enum{ // the position of each wall with the right
cordinates
					LEFTWALL = 0,
					RIGHTWALL = 480,
					TOPWALL = 10,
					BOTTOMWALL = 272
				} walls_t;



	/**************************************************************************
**********/

	/***********************************VARIABLES**************************
**********/

	/**************************************************************************
**********/

					void InitMagic();
					void magicTimeCheck();
```

```
                                        int magicTime, ticker, magicTimeCt,
magicTimeInterval, magicTimeDuration;
                                        int paddle;
                                        int lineX;
                                        int lineLength;
                                        int score;
                                        int scrInc = 1;
                                        int total;
                                        int multiplier;
                                        int remainingBalls;



        /*************************************************************************
**********/
                        /*****************************FUNCTION
PROTOTYPES***********************************/

        /*************************************************************************
**********/

                                        ball_t initBallAndBat(ball_t b);
                                        void ballRendering(ball_t b);
                                        ball_t moveBall(ball_t b);
                                        ball_t readDevices(ball_t b);
                                        void renderPaddle(int x);
                                        void waitForStart();
                                        int initialiseGame();
                                        int chekcBtmWall(ball_t b);
                                        void initDevices();
                                        void displayUpdate(ball_t b);
                                        void scorePoint();
                                        void gameOver(ball_t b);
```

```
                ball_t ball;
                ball_t b;


        /*************************************************************************
**********/
                        /***********************************MAIN
FUNCTION*********************************/

        /*************************************************************************
**********/
                        int main() {
                                // Initialise the display
                                Ticker ticktock;
                                ticktock.attach(&timerHandler, 1);
                                ball_t ball;
                                int inPlay;
                                initDevices();
                                remainingBalls = initialiseGame();

                                while(true){
                                        while(remainingBalls > 0){

                                                waitForStart();
                                                inPlay = 1;
                                                InitMagic();
                                                ball = initBallAndBat(ball);
                                                displayUpdate(ball);
                                                ballRendering(ball);
                                                ball = moveBall(ball);
                                                while(inPlay){
                                                        magicTimeCheck();
                                                        displayUpdate(ball);
```

```
                                                ballRendering(ball);
                                                ball = readDevices(ball);
                                                ball = moveBall(ball);
                                                renderPaddle(paddle);
                                                inPlay = chekcBtmWall(ball);
                                                if(!inPlay){
                                                        gameOver(ball);
                                                }

                                                wait(0.005); //5
milliseconds
                                        }

                                }

                                        screen->setCursor(2,2);
                                        screen->printf("GAME OVER! Your Total
score :  %d , Press center on joystic to reset          ", total);
                                        waitForStart();
                                        total = 0;
                                        remainingBalls = initialiseGame();
                                }

                        }




        /*************************************************************************
**********/
                        /************************BALL AND PADDLE MOVEMENT +
OBSTACLE************************/
```

```
        /*************************************************************************
**********/


                                ball_t initBallAndBat(ball_t b){ // takes ball_t as @param
                                        //randomise b.vx's sign and b.vy using rand
                                        int random = rand()% 2;
                                        int random2 = rand()% 2;

                                        //randomise bat and ball starting positions
                                        paddle = 160 + rand() % (321 - 160); // randomises
the position fo the paddle/Bat
                                        b.x = 160 + rand() % (281 - 160); // randomises the
position of ball
                                        b.y = 40;
                                        if(random == 0){
                                                b.vx = 1;
                                        }else{
                                                b.vx = -1;
                                        }

                                        //randomise 2 different speeds for the ball, and pick
one.
                                        if(random2 == 0){
                                                b.vy = 2;
                                        }else if(random2 == 1){
                                                b.vy = 3;
                                        }

                                        //randomise the length and the postion of the extra
obstacle
                                        lineX = 0 + rand() % (0 - 281);
```

```
                lineLength = 40 + rand() % (40 - 201);
                b.colour = BLUE;
                return b;
        }

        void ballRendering(ball_t b){
                screen->fillCircle(b.x, b.y, 5, WHITE);
}

        /*************************************************************************
**********/
                /************************COLLISION CHECKS AND
MOVEMENT*******************************/

        /*************************************************************************
**********/

        ball_t moveBall(ball_t b){
                //side wall and top wall collision checks
                if(((b.x + b.vx) >= RIGHTWALL) || ((b.x + b.vx) <=
LEFTWALL)){
                        b.vx = -b.vx;
                }

                else if(((b.y + b.vy) <= TOPWALL)){
                        multiplier++;
                        scorePoint();
                        b.vy = -b.vy;
                }

                //bat collision checks
```

```
                else if(((b.y + b.vy) >= 258) && ((b.x + b.vx) >= paddle) &&
((b.x + b.vx)<= (paddle + 40))){
                        b.vy = -b.vy;
                }

                //obstacle collision checks - accounts for variable y velocity
(2 - 3)
                else if((((b.y + b.vy) == 60) || ((b.y + b.vy) == 61) || ((b.y +
b.vy) == 59)) &&
                                ((b.x + b.vx) >= lineX) &&
((b.x + b.vx) <= lineX + lineLength)){
                        b.vy = -b.vy;
                }

                if (magicTime) {
                        b.colour = RED;
                }else{
                        b.colour = BLUE;
                }


                b.x += b.vx;
                b.y += b.vy;
                screen->fillCircle(b.x, b.y, 5, b.colour);
                return b;
        }

        int chekcBtmWall(ball_t b){
                return b.y + b.vy < 272;
        }

        void renderPaddle(int x){
```

```
                screen->fillRect(paddle, 258, 40, 4, BLACK);
        }



    /**************************************************************************
**********/
                /************************DRAW OBJECTS ON SCREEN AND BAT
CONTROLS*********************/

    /**************************************************************************
**********/


        ball_t readDevices(ball_t b){
                acc.read(accVal[0], accVal[1], accVal[2]);
                screen->fillRect(paddle, 258, 40, 4, WHITE); // clear
old bat
                // check for inputs and possible boundary oversteps
                if (jsBtns[1].read() == 0) {
                        if((paddle - 1) >= LEFTWALL){
                                paddle -= 2;
                        }
                } else if (jsBtns[0].read() == 0) {
                        if(((paddle + 1) + 40) <= RIGHTWALL){
                                paddle += 2;
                        }
                } else if(accVal[0] < -10){
                        if((paddle - 1) >= LEFTWALL){
                                paddle -= 2;
                        }
                } else if(accVal[0] > 10){
                        if(((paddle + 1) + 40) <= RIGHTWALL){
                                paddle += 2;
```

```
                }
        }
        //check for potentiometer readings, change speed based on them
        potVal = pot.read();
        if(potVal < 0.33f){
                if(b.vy < 0){
                        b.vy = -1;
                }else{
                        b.vy = 1;
                }
        }else if((potVal > 0.33f)&&(potVal <=0.66f)){
                if(b.vy < 0){
                        b.vy = -2;
                }else{
                        b.vy = 2;
                }
        }else if((potVal > 0.66f)&&(potVal <= 1)){
                if(b.vy < 0){
                        b.vy = -3;
                }else{
                        b.vy = 3;
                }
        }
        return b;
    }

    /*************************************************************************
**********/
```

```
                    /*************************MAGIC TIME AND INTERRUPT AND
SCORE*************************/

        /*************************************************************************
**********/
                                        void timerHandler() {
                                                if(ticker){
                                                        tickCt++;
                                                        if(tickCt== 11){
                                                                tickCt = 0;
                                                        }
                                                }
                                                        else if (magicTime) {
                                                                magicTimeCt++;
                                                                if (magicTimeCt==11){
                                                                        magicTimeCt = 0;
                                                                }
                                                        }
                                                }



                                        void InitMagic(){
                                                ticker = 1;
                                                magicTime = 0;
                                                magicTimeInterval = 5 + rand() % (11-5);
                                                magicTimeDuration = 2 + rand() % (11-2);
                                        }

                                        void magicTimeCheck(){
                                                if (tickCt == magicTimeInterval){
                                                        ticker = 0;
                                                        magicTime = 1;
```

```
                tickCt = 0;
            }else if (magicTimeCt == magicTimeDuration){
                ticker = 1;
                magicTime = 0;
                magicTimeCt = 0 ;
                magicTimeInterval = 5 + rand() % (11-5);
                magicTimeDuration = 2 + rand() % (11-2);
            }
        }


        void scorePoint(){
            if ((multiplier % 5 ) == 0){
                scrInc = 1 + (multiplier / 5);
            }
            if (magicTime) {
                score += (2 * scrInc);
            }
            else{
                score += scrInc;
            }
        }

    /**************************************************************************
**********/
                /************************DEVICE INITIALISATION AND GAME
INIT************************/

    /**************************************************************************
**********/
        int initialiseGame(){
            // sets balls left to 5
```

```
                return 5;
        }

        void initDevices(){
                screen->fillScreen(WHITE);
                screen->setTextColor(BLACK, WHITE);
                // Initialise accelerometer and temperature sensor
        }



        /**********************************************************************
**********/

        /*************************************DISPLAY*************************
**********/

        /**********************************************************************
**********/
        void displayUpdate(ball_t b){
                screen->setCursor(2,2);
                screen->printf("Score: [%d]   |    multiplier: [%d]   |
Total: [%d]   |   Balls remaining: [%d]       ",
                score, multiplier, total, remainingBalls);
                screen->drawLine(lineX, 60, lineX + lineLength, 60,
GREEN);
        }



        /**********************************************************************
**********/
                /************************START ANG GAME
OVER*************************************/
```

```
        /*************************************************************************
**********/
                                        void waitForStart(){
                                                while (!jsPrsdAndRlsd(JCR)){
                                                        //do nothing!
                                                }
                                        }
                                        void gameOver(ball_t b){
                                                remainingBalls--;
                                                screen->fillCircle(b.x, b.y, 5, WHITE);
                                                screen->fillRect(paddle, 258, 40, 4, WHITE);
                                                screen->drawLine(lineX, 60, lineX + lineLength, 60,
WHITE);
                                                multiplier = 0;
                                                total += score;
                                                score = 0;
                                                scrInc = 1;
                                                magicTime = 0;
                                                ticker = 0;
                                                tickCt = 0;
                                                magicTimeCt = 0;

                                        }

/* Definition of Joystick press capture function
 * b is one of JLEFT, JRIGHT, JUP, JDOWN - from enum, 0, 1, 2, 3
 * Returns true if this Joystick pressed then released, false otherwise.
 *
 * If the value of the button's pin is 0 then the button is being pressed,
 * just remember this in savedState.
 * If the value of the button's pin is 1 then the button is released, so
```

```
 * if the savedState of the button is 0, then the result is true, otherwise
 * the result is false. */
bool jsPrsdAndRlsd(btnId_t b) {
        bool result = false;
        uint32_t state;
        static uint32_t savedState[5] = {1,1,1,1,1};
      //initially all 1s: nothing pressed
        state = jsBtns[b].read();
  if ((savedState[b] == 0) && (state == 1)) {
                result = true;
        }
        savedState[b] = state;
        return result;
}

bool accInit(MMA7455& acc) {
bool result = true;
if (!acc.setMode(MMA7455::ModeMeasurement)) {
// screen->printf("Unable to set mode for MMA7455!\n");
result = false;
}
if (!acc.calibrate()) {
// screen->printf("Failed to calibrate MMA7455!\n");
result = false;
}
// screen->printf("MMA7455 initialised\n");
return result;
}
```