```
#include <LiquidCrystal.h>

#define PIN_BUTTON 2

#define PIN_READWRITE 10
#define PIN_CONTRAST 12

#define DINO_RUN1_CHAR_INDEX 1
#define DINO_RUN2_CHAR_INDEX 2
#define DINO_JUMP_CHAR_INDEX 3
#define DINO_JUMP_CHAR_INDEX_UPPER 8
#define DINO_JUMP_CHAR_INDEX_LOWER 4

#define CACTUS_CENTER_CHAR_INDEX 5
#define CACTUS_RIGHT_PART_CHAR_INDEX 6
#define CACTUS_LEFT_PART_CHAR_INDEX 7

#define SPACE ' '      // User the ' ' character
#define INDEX_1 1    // Horizontal position of dino on screen when starting the
game

#define ROW_WIDTH 16
#define ROW_EMPTY 0
#define ROW_LOWER_VALUE_1 1
#define ROW_UPPER_VALUE_2 2

#define DINO_POSITION_OFF 0          // Dino is invisible
#define DINO_POSITION_RUN_LOWER_ROW_POSITION_1 1  // Dino is running on lower
row (pose 1)
#define DINO_POSITION_RUN_LOWER_ROW_POSITION_2 2  //
(pose 2)

#define DINO_POSITION_JUMP_1 3       // Starting a jump
#define DINO_POSITION_JUMP_2 4       // Half-way up
#define DINO_POSITION_JUMP_3 5       // Jump is on upper row
#define DINO_POSITION_JUMP_4 6       // Jump is on upper row
#define DINO_POSITION_JUMP_5 7       // Jump is on upper row
#define DINO_POSITION_JUMP_6 8       // Jump is on upper row
#define DINO_POSITION_JUMP_7 9       // Half-way down
#define DINO_POSITION_JUMP_8 10      // About to land

#define DINO_POSITION_RUN_UPPER_ROW_POSITION_1 11 // Dino is running on upper
row (pose 1)
#define DINO_POSITION_RUN_UPPER_ROW_POSITION_2 12 //
(pose 2)

LiquidCrystal lcd(11, 9, 6, 5, 4, 3);

static char rowUpper[ROW_WIDTH + 1];
static char rowLower[ROW_WIDTH + 1];

static bool buttonPushed = false;

void initializeGraphics(){

  static byte graphics[] = {
    // Dino Run position 1
    B11110,
    B10111,
```

```
B11110,
B11100,
B11111,
B11100,
B10010,
B11011,
// Dino Run position 2
B11110,
B10111,
B11110,
B11100,
B11111,
B11100,
B01010,
B10011,
// Dino Jump
B11110,
B10111,
B11110,
B11100,
B11111,
B11110,
B10001,
B00000,
// Jump lower part [lower part of Dino while jumping]
B11111,
B11100,
B10010,
B11011,
B00000,
B00000,
B00000,
B00000,
// Cactus Center part
B00100,
B11111,
B00100,
B00101,
B11111,
B00100,
B00100,
B00100,
// Cactus right part
B00000,
B00011,
B00000,
B00000,
B00011,
B00000,
B00000,
B00000,
// Cactus left part
B00000,
B11000,
B00000,
B01000,
B11000,
B00000,
B00000,
```

```
    B00000,
    // Jump upper part [upper part of Dino while jumping]
    B00000,
    B00000,
    B00000,
    B00000,
    B11110,
    B10111,
    B11110,
    B11100,
  };
  int i;
  // Skip using character 0, this allows lcd.print() to be used to
  // quickly draw multiple characters
  for (i = 0; i < 8; ++i) {
    lcd.createChar(i + 1, &graphics[i * 8]); // characters getting generated are
1,2,3,4,5,6,7
  }
  for (i = 0; i < ROW_WIDTH; ++i) {      // ROW_WIDTH 16
    rowUpper[i] = SPACE; // SPACE ' '
    rowLower[i] = SPACE;
  }
}

// Slide the row to the left in half-character increments
// newRow = 5  or ' '
void advanceRow(char* row, byte newRow){
  for (int i = 0; i < ROW_WIDTH; ++i) {       //ROW_WIDTH 16
    char current = row[i];
    char next = (i == ROW_WIDTH-1) ? newRow : row[i+1];
    switch (current){
      case SPACE:
        row[i] = (next == CACTUS_CENTER_CHAR_INDEX) ?
CACTUS_RIGHT_PART_CHAR_INDEX : SPACE;
        break;
      case CACTUS_CENTER_CHAR_INDEX:
        row[i] = (next == SPACE) ? CACTUS_LEFT_PART_CHAR_INDEX :
CACTUS_CENTER_CHAR_INDEX;
        break;
      case CACTUS_RIGHT_PART_CHAR_INDEX:
        row[i] = CACTUS_CENTER_CHAR_INDEX;
        break;
      case CACTUS_LEFT_PART_CHAR_INDEX:
        row[i] = SPACE;
        break;
    }
  }
}

bool drawDino(byte position, char* rowUpper, char* rowLower, unsigned int score)
{
  // position is 1 for the first time

  bool collide = false;
  char upperSave = rowUpper[INDEX_1];          // INDEX_1 1
  char lowerSave = rowLower[INDEX_1];
  byte upper, lower;
  switch (position) {
    case DINO_POSITION_OFF: // 0
```

```
      upper = lower = SPACE;       // SPACE ' '
      break;
    case DINO_POSITION_RUN_LOWER_ROW_POSITION_1:  // 1
      upper = SPACE;
      lower = DINO_RUN1_CHAR_INDEX;           // 1
      break;
    case DINO_POSITION_RUN_LOWER_ROW_POSITION_2: // 2
      upper = SPACE;
      lower = DINO_RUN2_CHAR_INDEX;          // 2
      break;
    case DINO_POSITION_JUMP_1:   // DINO_POSITION_JUMP_1 = 3 Starting a jump
    case DINO_POSITION_JUMP_8:   // DINO_POSITION_JUMP_8 = 10 About to land
      upper = SPACE;
      lower = DINO_JUMP_CHAR_INDEX;       // 3
      break;
    case DINO_POSITION_JUMP_2:   // DINO_POSITION_JUMP_2 = 4  Half-way up
    case DINO_POSITION_JUMP_7:   // DINO_POSITION_JUMP_7 = 9  Half-way down
      upper = DINO_JUMP_CHAR_INDEX_UPPER; // 8
      lower = DINO_JUMP_CHAR_INDEX_LOWER; // 4
      break;
    case DINO_POSITION_JUMP_3: // 5 Jump is on upper row
    case DINO_POSITION_JUMP_4: // 6 Jump is on upper row
    case DINO_POSITION_JUMP_5: // 7 Jump is on upper row
    case DINO_POSITION_JUMP_6: // 8 Jump is on upper row
      upper = DINO_JUMP_CHAR_INDEX;     // DINO_JUMP_CHAR_INDEX 3
      lower = SPACE;  // SPACE ' '
      break;
    case DINO_POSITION_RUN_UPPER_ROW_POSITION_1:    // 11  , Hero is running on
upper row (pose 1)
      upper = DINO_RUN1_CHAR_INDEX;             // 1
      lower = SPACE;    // SPACE ' '
      break;
    case DINO_POSITION_RUN_UPPER_ROW_POSITION_2:    // 12  , Hero is running on
upper row (pose 2)
      upper = DINO_RUN2_CHAR_INDEX;             // 2
      lower = SPACE;    // SPACE ' '
      break;
  }
  if (upper != ' ') {
    rowUpper[INDEX_1] = upper;               // INDEX_1 1
    collide = (upperSave == SPACE) ? false : true;
  }
  if (lower != ' ') {
    rowLower[INDEX_1] = lower;
    collide |= (lowerSave == SPACE) ? false : true;             // |= bitwise
OR, a = a | b
  }

  byte digits = (score > 9999) ? 5 : (score > 999) ? 4 : (score > 99) ? 3 :
(score > 9) ? 2 : 1;

  // Draw the scene
  rowUpper[ROW_WIDTH] = '\0';
  rowLower[ROW_WIDTH] = '\0';
  char temp = rowUpper[16-digits];
  rowUpper[16-digits] = '\0';
  lcd.setCursor(0,0);

  lcd.print(rowUpper);
```

```
  rowUpper[16-digits] = temp;
  lcd.setCursor(0,1);

  lcd.print(rowLower);

  lcd.setCursor(16 - digits,0);
  lcd.print(score);

  rowUpper[INDEX_1] = upperSave;
  rowLower[INDEX_1] = lowerSave;
  return collide;
}

// Handle the button push as an interrupt
void buttonPush() {
  buttonPushed = true;
}

void setup(){
  Serial.begin(9600);
  pinMode(PIN_READWRITE, OUTPUT);
  digitalWrite(PIN_READWRITE, LOW);
  pinMode(PIN_CONTRAST, OUTPUT);
  analogWrite(PIN_CONTRAST, 75);
  pinMode(PIN_BUTTON, INPUT);
  digitalWrite(PIN_BUTTON, HIGH);

  // Digital pin 2 maps to interrupt 0
  attachInterrupt(0/*PIN_BUTTON*/, buttonPush, FALLING); // FALLING for when the
pin goes from high to low.

  initializeGraphics();

  lcd.begin(16, 2);
}

void loop(){
  static byte dinoPos = DINO_POSITION_RUN_LOWER_ROW_POSITION_1; // value 1
  static byte newRowType = ROW_EMPTY; // value 0
  static byte newRowDuration = 1;
  static bool playing = false;
  static bool blink = false;
  static unsigned int distance = 0;

  if (!playing) {
    drawDino((blink) ? DINO_POSITION_OFF : dinoPos, rowUpper, rowLower, distance
>> 3);  // DINO_POSITION_OFF = 0 ,Hero is invisible
    if (blink) {
      lcd.setCursor(0,0);
      lcd.print("Press Start");
    }
    delay(250);
    blink = !blink;
    if (buttonPushed) {
      initializeGraphics();
      dinoPos = DINO_POSITION_RUN_LOWER_ROW_POSITION_1;
      playing = true;
      buttonPushed = false;
      distance = 0;
```

```
    }
    return;
  }

  // Shift the row to the left
  advanceRow(rowLower, newRowType == ROW_LOWER_VALUE_1 ?
CACTUS_CENTER_CHAR_INDEX : SPACE); // CACTUS_CENTER_CHAR_INDEX 5 and
ROW_LOWER_VALUE_1 1


  // Make new row to enter on the right
  if (--newRowDuration == 0) {
    if (newRowType == ROW_EMPTY) {
      newRowType = ROW_LOWER_VALUE_1;
      newRowDuration = 2 + random(5);
    } else {
      newRowType = ROW_EMPTY;
      newRowDuration = 10 + random(5);
    }
  }

  if (buttonPushed) {
    if (dinoPos <= DINO_POSITION_RUN_LOWER_ROW_POSITION_2)
          dinoPos = DINO_POSITION_JUMP_1;
    buttonPushed = false;
  }

  if (drawDino(dinoPos, rowUpper, rowLower, distance >> 3)) {
    playing = false; // The dino collided with a cactus.
  } else {
    if (dinoPos == DINO_POSITION_RUN_LOWER_ROW_POSITION_2 || dinoPos ==
DINO_POSITION_JUMP_8) {
      dinoPos = DINO_POSITION_RUN_LOWER_ROW_POSITION_1;
    } else if ((dinoPos >= DINO_POSITION_JUMP_3 && dinoPos <=
DINO_POSITION_JUMP_5) && rowLower[INDEX_1] != SPACE) {
      dinoPos = DINO_POSITION_RUN_UPPER_ROW_POSITION_1;
    } else if (dinoPos >= DINO_POSITION_RUN_UPPER_ROW_POSITION_1 && rowLower
[INDEX_1] == SPACE) {
      dinoPos = DINO_POSITION_JUMP_5;
    } else if (dinoPos == DINO_POSITION_RUN_UPPER_ROW_POSITION_2) {
      dinoPos = DINO_POSITION_RUN_UPPER_ROW_POSITION_1;
    } else {
      ++dinoPos;
    }
    ++distance;

  }
  delay(100);
}
```