Dino Run Game

This is an Arduino game project that involves creating a simple game called Dino Run using the Adafruit_GFX and Adafruit_ST7735 libraries for the TFT display and the ADXL345_Unified library for the accelerometer. The player controls dinosaur and must jump over obstacles or other dinosaur's to avoid collision while advancing through the game. The score increases as the player collects the coins and the game ends if the player collides with any obstacles or other dinosaur. The game board also has extra features like Backlight Brightness adjustment, sound effects and the game board can be USB powered or battery powered.

Components Used

- Arduino UNO board
- 1.44inch SPI Module MSP1443 using ST7735S display driver
- ADXL345 I2C & SPI based accelerometer (used I2C in this project)
- Breadboard
- Jump wires /Hook wires

Optional components:

- Li-po battery
- Li-po charge discharge module
- 10K potentiometer
- 100Ω Resistor
- Small speaker

* if Low impedance speaker is used then add a sufficient series resistor to limit the max current drawing from the Digital pin

How to Install

1. Download and install the Arduino IDE.
2. Connect the Arduino board to the computer using a USB cable.
3. Open the Arduino IDE and select the correct board and port.
4. Download the required libraries.
5. Copy the code and paste it into the Arduino IDE.
6. Upload the code to the Arduino board.
7. Connect the TFT display and the accelerometer to the Arduino board according to the pin configuration.
8. Power up the Arduino board.
9. The game should start automatically.

Libraries Required

- Wire.h
- Adafruit_GFX.h
- Adafruit_ST7735.h
- Adafruit_ADXL345_U.h

Details of pins used in the code:

• TFT_CS: This is the chip select pin for the LCD display. It's used to select the display when sending commands or data to it.

• TFT_RST: This is the reset pin for the LCD display. It's used to reset the display when initializing it.

• TFT_DC: This is the data/command pin for the LCD display. It's used to distinguish between commands and data when sending information to the display.

• MOSI (Master Out Slave In): This pin is used to transmit data from the Arduino to the LCD display.

• MISO (Master In Slave Out): This pin is used to transmit data from the LCD display to the Arduino (Used in the displays which has inbuilt memory card slot to tranmit data from SD card to Arduino we don't be using this pin as the display used here not have any SD card slot).

• SCK (Serial Clock): This pin is used to synchronize the data transfer between the Arduino and the LCD display.

To power the LCD screen connect the LCD V_{CC} to 5V and GND to GND of Arduino and back LED pin to 3.3V (if you want to add brightness adjustment feature connect a resistor b/w the PWM pin and LED typically 100Ω-1KΩ will be fine.

• **Accelerometer:** The accelerometer used in this code is the ADXL345module. It communicates with the Arduino using the I2C and SPI protocol, but in this project we used I2C of ADXL345 to Communicate with Arduino. which uses the SDA (data) and SCL (clock) pins on the Arduino board.

• SDA (Serial Data) is the bidirectional line used for transmitting and receiving data between the master and slave devices.

• SCL (Serial Clock) is the unidirectional line used to synchronize the data transfer between the master and slave devices.

To power the sensor connect the V_{CC} to 3.3V and GND to GND of Arduino

*** V_{CC} must be connected to 3.3V If you accidentally connected to 5V the board may be permanently damaged (max voltage stated in the datasheet is 3.9V)**

In summary, the TFT_CS, TFT_RST, and TFT_DC pins are used to communicate with the LCD display, while the accelerometer module communicates with the Arduino board using the I2C protocol over the SDA and SCL pins, and requires power and ground connections.

Pin Configuration

Accelerometer:

- SDA: A4
- SCL: A5
- VCC: 3.3V
- GND: GND

LCD Display:

- CS: D10
- RST: D9
- DC: D8
- MOSI: D11
- SCK: D13
- LED: 3.3V
- VCC: 5V
- GND: GND

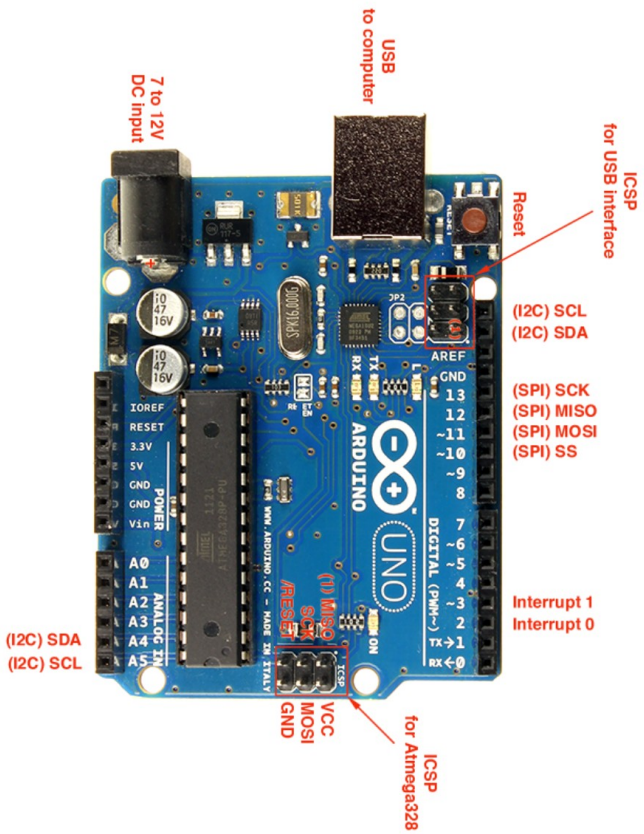

Image representing the Pin configurations in Arduino UNO

Optional connections for Brightness adjustment and External battery power

- A0 -----> potentiometer mid pin (wiper pin)
- Potentiometer -----> 5V & GND (end pins)
- A1 -----> Battery(+ve)

*** Refer circuit diagram for more detailes**

How to Play

- The game starts with the dinosaur at the left of the screen.
- The player controls the dinosaur with the accelerometer.
- If the accelerometer's y-value is greater than 9.0, the dinosaur jumps.
- The dinosaur must jump over obstacles to avoid collision and advance through the game.
- The score increases as the player progresses.
- The game ends if the dinosaur collides with any obstacle.

Summary of the Game code

This code is a simple game based on the classic Google Chrome dinosaur game where the player controls a dinosaur that needs to jump over obstacles. The game is designed to run on an Arduino board with a 1.8" TFT display and an ADXL345 accelerometer sensor to detect the dino's jump. The code includes libraries for the display and the sensor.

The game consists of a static baseline and a randomly generated obstacle, which can be a cactus or a group of cacti, that move from right to left across the screen. The player must control the dinosaur to avoid the obstacles by jumping over them.The code initializes the display, accelerometer, and other variables. The setup() function sets up the initial display and draws the baseline and the first obstacle. The loop() function contains the main game loop that runs continuously while the game is not over. Inside the loop() function, the code first moves the obstacle to the left by changing the x-coordinate of the obstacle. Then, the drawFrame() function is called to draw the new frame of the game.

The checkCollision() function is called to check if the dino collides with the obstacle. If a collision occurs, the gameOver variable is set to true, and the game ends.
The jump() function is called if the player makes a jump by tilting the accelerometer sensor above a certain threshold. The function moves the dinosaur up by changing the y-coordinate and redraws the dinosaur. If the dinosaur is not jumping, it moves down by increasing the y-coordinate. If the dinosaur lands on the ground, the isJumping variable is set to false. The score variable is incremented each frame, and the drawScore() function is called to display the score on the screen. When the obstacle moves off the screen, the treeType variable is randomly set to either 1 or 2, and a new obstacle is generated. The drawDino() function draws the dinosaur on the screen using the Adafruit_GFX library.

Finally, the drawGameOver() function is called when the game is over to display the "GAME OVER" message on the screen.

Brief explanation of the code:

1. Libraries:

The code includes the following libraries:

- Wire.h: This is a library that allows communication with I2C devices such as the ADXL345 accelerometer sensor.

- Adafruit_GFX.h: This is a graphics library that provides a common set of graphics primitives (points, lines, circles, etc.) for Adafruit display drivers.

- Adafruit_ST7735.h: This is a library for the ST7735 LCD display.

- Adafruit_ADXL345_U.h: This is a library for the ADXL345 accelerometer sensor.

2. Defines:

The code includes several constants that are defined using the #define preprocessor directive. These constants include:

• SCREEN_WIDTH: The width of the LCD display in pixels (128 pixels).

• SCREEN_HEIGHT: The height of the LCD display in pixels (128 pixels).

• TFT_CS, TFT_RST, TFT_DC: The pin numbers used for the chip select, reset, and data/command pins of the LCD display.

• DINO_WIDTH, DINO_HEIGHT: The width and height of the dinosaur character in pixels (30and 30 pixels, respectively).

• DINO_INIT_X, DINO_INIT_Y: The initial x and y coordinates of the dinosaur character when it first appears on the screen (10 and 99 pixels, respectively).

• BASE_LINE_X, BASE_LINE_Y, BASE_LINE_X1, BASE_LINE_Y1: The x and y coordinates of the baseline (ground) on which the dinosaur character runs (0, 117, 127, 117 pixels, respectively).

• TREE1_WIDTH, TREE1_HEIGHT: The width and height of the first type of obstacle (11 and 23 pixels, respectively).

• TREE2_WIDTH, TREE2_HEIGHT: The width and height of the second type of obstacle (22 and 23 pixels, respectively).

• TREE3_WIDTH, TREE3_HEIGHT: The width and height of the third type of obstacle (30 and 40 pixels, respectively).

• DINO_WIDTH, DINO_HEIGHT: The width and height of the 4^{th} type of obstacle (30 and 40 pixels, respectively).

• TREE_Y: The y-coordinate of the top of the obstacles (94 pixels).

• JUMP_PIXEL: The number of pixels the dinosaur character jumps when it jumps over an obstacle (22 pixels).

3. Global variables:

The code includes several global variables that are used throughout the code. These variables include:

• tft: An instance of the Adafruit_ST7735 class used to control the LCD display.

• accel: An instance of the Adafruit_ADXL345_Unified class used to read the acceleration data from the ADXL345 accelerometer sensor.

• gameOver: A boolean variable that is set to true when the game is over (i.e., the dinosaur character collides with an obstacle).

• isJumping: A boolean variable that is set to true when the dinosaur character is jumping over an obstacle.

• score: An integer variable that keeps track of the player's score.

• dinoX, dinoY: Integer variables that hold the current x and y coordinates of the dinosaur character on the screen.

• treeX: An integer variable that holds the current x-coordinate of the obstacle on the screen.

• treeType: An integer variable that holds the type of the obstacle (1 or 2).

4. Setup function:

• Initializes the ST7735 display using the tft.initR(INITR_BLACKTAB) and tft.setRotation(3) functions to set up the display and rotate it 270 degrees.

• Initializes the ADXL345 accelerometer sensor using the accel.begin(), accel.setRange(ADXL345_RANGE_2_G), and accel.setDataRate(ADXL345_DATARATE_100_HZ) functions to set the range and data rate of the sensor.

• Fills the screen with white using the tft.fillScreen(ST7735_WHITE) function.

• Draws the initial frame of the game by calling the drawDino(), drawBaseLine(), and drawTree() functions.

5. Loop :

•Reads the acceleration values from the ADXL345 sensor using the accel.getEvent(&event) function.
• Moves the tree to the left by decreasing its x-coordinate by 2 pixels.
• Checks for collisions between the dino and the tree using the checkCollision() function.
• Updates the position of the dino based on its jumping status and acceleration values.
• Increases the score by one for each frame.
• Calls the drawFrame() function to draw the updated game frame.
• Checks if the tree has gone off the screen, and if so, randomizes its type and resets its position.
• Draws the score on the screen using the drawScore() function.
• If the game is over due to a collision, calls the drawGameOver() function to display the "GAME OVER" message.

Extras:

Potentiometer to Control Backlight:

• Added a potentiometer to adjust the brightness of backlight.

• converted the analog input from the potentiometer into a PWM signal using the analogWrite() function.

• The PWM signal is then used to control the brightness of the LED connected to pin 3 of the Arduino.

• By adjusting the potentiometer, the analog input value will be changed and thereby change the PWM signal, which in turn changes the brightness of the LED.

• A current limiting resistor is kept between D3 and LED pin to limit the current going to Backlight

Limitations:

Hardware limitations: The game is designed to run on an Arduino board with limited processing power and memory, which may limit the complexity and graphics quality of the game.

1. Screen size limitations: The 1.44 inch TFT display used in this project is relatively small, which may limit the amount of detail and information that can be displayed on the screen.

2. Accelerometer accuracy: The ADXL345 accelerometer used in this project may not be as accurate as other more advanced sensors, which may affect the gameplay and control of the dinosaur character.

3. Limited obstacle variety: The game only includes two types of obstacles (cactus and group of cacti), which may make the game predictable and less challenging over time.

4. No high score tracking: The game does not include any mechanism to track and display high scores, which may reduce the competitive aspect of the game.

5. Limited player interaction: The game only allows the player to control the dinosaur character by jumping over obstacles, and does not include any additional actions or movements, which may limit the overall player engagement and immersion in the game.