Bahria University,

Karachi Campus



CLASS: BSE- 3 (A)

Flappybird (Game)

Project Report

Dr. Samar Yazdani/ Engr. Ramsha Mashood

Signed Remarks: Score:

**Group Members:**

* + - Zain Shahid(Leader) (02-131202-032)
    - Aneesha Kumari (02-131202-052)
    - Khizra Tariq Syeda (02-131202-001)

**Table of Contents**

[**1.** **INTRODUCTION & PROBLEM** 2](#_Toc94640329)

[**2.** **TECHNOLOGY** 3](#_Toc94640330)

[**3.** **FUNCTIONALITIES** 3](#_Toc94640331)

[**4.** **MODULE DISTRIBUTION:** 3](#_Toc94640332)

[**5.** **CODE:** 4](#_Toc94640333)

[**6.** **INTERFACES:** 12](#_Toc94640334)

[**7.** **CONCLUSION:** 14](#_Toc94640335)

* + - 1. **INTRODUCTION & PROBLEM:**

Flappy Bird is an arcade-style game in which the player controls the bird, which moves persistently to the right. The player is tasked with navigating the bird through pairs of pipes that have equally sized gaps placed at random heights. It automatically descends and only ascends when the player taps the ‘f’ button on the keyboard.

* + - 1. **TECHNOLOGY:**

We used Mars simulator for our project. We made our project on bitmap of MIPS Assembly language.

* + - 1. **FUNCTIONALITIES:**

In our game, the image moving is the most important function we need to implement. Besides the background horizontal moving, the vertical jumping combining keyboard input is our second challenge. The bird briefly flaps upward each time the player taps the ‘f’ button; if the button is not tapped, the bird falls due to gravity.

# **MODULE DISTRIBUTION:**

**Zain Shahid:**

Implementation of pipe, coding of top refresh and bot refresh.

**Aneesha Kumari:**

Coding of bird, using loop and method of jump.

**Khizra Tariq Syeda:**

Designing of bitmap display, coding of reset of game and exit screen.

# **CODE:**

# Bitmap Display Configuration:

# - Unit width in pixels: 8

# - Unit height in pixels: 8

# - Display width in pixels: 256

# - Display height in pixels: 256

# - Base Address for Display: 0x10008000 ($gp)

#

#

#####################################################################

.data

displayAddress: .word 0x10008000

r: .word 4

green: .word 0x00ff44

blue: .word 0x00a2ff

yellow: .word 0xfbff00

firstval: .word 0xffff0000

secondval: .word 0xffff0004

asciif: .word 102

time: .word 250

.text

main:

lw $t0, displayAddress # $t0 stores the base address for display

lw $t1, blue # $t1 stores the yellow and blue colour code

lw $t2, green # $t2 stores the green colour code

lw $t3, time # $t3 sotees the time that syscall waits

move $t4, $zero # $t4 stores the counter for the units

li $t6, 1792 # $t6 stores the address for the corner of the bird

li $t7, 1020 # $t7 stores the edited address for the corner of the pipe

lw $s0, firstval # s0 stores address of the first val - whether there is input

lw $s1, secondval # s1 stores address of the second val - input

lw $s2, asciif # s2 stores ascii code for f

li $t8, 1024 # t8 stores the address for the corner of the pipe

li $t9, 8 # t9 stores the pipe offset

li $s7, 0 # stores the randomized value

li $s6, 128 # stores the length of the display

li $a3, 4 # stores a value to check in ending

while:

add $t5, $t4, $t0 # t5 gets updated with the next unit

sw $t1, 0($t5) # paint the first unit on the second row blue. Why +128?

add $t4, $t4, 4 # increment to next unit

bne $t4, 4096, while # jump back if the screen is not fully painted

Bird:

lw $t1, yellow

add $t6, $t6, 28

add $t5, $t6, $t0 # t5 gets updated with the corner of the bird

lw $s5, 0($t5)

sw $t1, 0($t5) # paint the bird

beq $s5, $t2, Exit

add $t6, $t6, 8

add $t5, $t6, $t0 # t5 gets updated with the corner of the bird

lw $s5, 0($t5)

sw $t1, 0($t5) # paint the bird

beq $s5, $t2, Exit

add $t6, $t6, 120

add $t5, $t6, $t0 # t5 gets updated with the corner of the bird

sw $t1, 0($t5) # paint the bird

add $t6, $t6, 4

add $t5, $t6, $t0 # t5 gets updated with the corner of the bird

sw $t1, 0($t5) # paint the bird

add $t6, $t6, 4

add $t5, $t6, $t0 # t5 gets updated with the corner of the bird

sw $t1, 0($t5) # paint the bird

add $t6, $t6, 4

add $t5, $t6, $t0 # t5 gets updated with the corner of the bird

lw $s5, 0($t5)

sw $t1, 0($t5) # paint the bird

beq $s5, $t2, Exit

add $t6, $t6, 116

add $t5, $t6, $t0 # t5 gets updated with the corner of the bird

lw $s5, 0($t5)

sw $t1, 0($t5) # paint the bird

beq $s5, $t2, Exit

add $t6, $t6, 8

add $t5, $t6, $t0 # t5 gets updated with the corner of the bird

lw $s5, 0($t5)

sw $t1, 0($t5) # paint the bird

beq $s5, $t2, Exit

sub $t6, $t6, 292

j Loop

Loop:

jal Pipe

lw $t1, blue

lw $s3, 0($s0)

bne $s3, $zero, Jump

add $t6, $t6, 28

add $t5, $t6, $t0 # t5 gets updated with the corner of the bird

sw $t1, 0($t5) # paint the bird

add $t6, $t6, 8

add $t5, $t6, $t0 # t5 gets updated with the corner of the bird

sw $t1, 0($t5) # paint the bird

add $t6, $t6, 124

add $t5, $t6, $t0 # t5 gets updated with the corner of the bird

sw $t1, 0($t5) # paint the bird

add $t6, $t6, 8

add $t5, $t6, $t0 # t5 gets updated with the corner of the bird

sw $t1, 0($t5) # paint the bird

sub $t6, $t6, 40

bge $t6, 3900, Exit

j Bird

Jump:

lw $s3, 0($s1)

bne $s3, $s2, Loop

beq $t6, $zero, Loop

li $a0, 25 # set sleep to 100

li $v0, 32 # set syscall to sleep

syscall # sleep 100

lw $t1, blue

add $t6, $t6, 156

add $t5, $t6, $t0 # t5 gets updated with the corner of the bird

sw $t1, 0($t5) # paint the bird

add $t6, $t6, 4

add $t5, $t6, $t0 # t5 gets updated with the corner of the bird

sw $t1, 0($t5) # paint the bird

add $t6, $t6, 4

add $t5, $t6, $t0 # t5 gets updated with the corner of the bird

sw $t1, 0($t5) # paint the bird

add $t6, $t6, 4

add $t5, $t6, $t0 # t5 gets updated with the corner of the bird

sw $t1, 0($t5) # paint the bird

add $t6, $t6, 116

add $t5, $t6, $t0 # t5 gets updated with the corner of the bird

sw $t1, 0($t5) # paint the bird

add $t6, $t6, 8

add $t5, $t6, $t0 # t5 gets updated with the corner of the bird

sw $t1, 0($t5) # paint the bird

sub $t6, $t6, 548

j Bird

Pipe:

move $a0, $t3 # set sleep to 100

li $v0, 32 # set syscall to sleep

syscall # sleep 100

sub $t7, $t8, $t9

add $t7, $t7, $s7

lw $t1, blue

add $t7 $t7, 8

add $t7 $t7, 512

TopRefresh:

add $t5, $t7, $t0 # t5 gets updated with a piece of the pipe

sw $t1, 0($t5) # paint out the pipe

sub $t7, $t7, 128

bge $t7, $zero, TopRefresh

add $t7, $t7, 2048

BotRefresh:

add $t5, $t7, $t0 # t5 gets updated with a piece of the pipe

sw $t1, 0($t5) # paint out the pipe

add $t7, $t7, 128

ble $t7, 4096, BotRefresh

sub $t7, $t8, $t9

add $t7, $t7, $s7

j TopPipe

Clear:

li $t7, 128

Clear1:

add $t5, $t7, $t0 # t5 gets updated with a piece of the pipe

sw $t1, 0($t5) # paint out the pipe

add $t7, $t7, 128

bne $t7, 4096, Clear1

li $t7, 124

Clear2:

add $t5, $t7, $t0 # t5 gets updated with a piece of the pipe

sw $t1, 0($t5) # paint out the pipe

add $t7, $t7, 128

bne $t7, 4092, Clear2

li $t7, 100

add $t5, $t7, $t0 # t5 gets updated with a piece of the pipe

sw $t2, 0($t5) # paint the pipe

sub $t7, $t7, 4

add $t5, $t7, $t0 # t5 gets updated with a piece of the pipe

sw $t2, 0($t5) # paint the pipe

sub $t7, $t7, 4

add $t5, $t7, $t0 # t5 gets updated with a piece of the pipe

sw $t2, 0($t5) # paint the pipe

sub $t7, $t7, 120

add $t7, $t7, 2040

add $t7, $t7, $s7

j BotPipe

TopPipe:

beq $t7, 100, Clear

add $t5, $t7, $t0 # t5 gets updated with a piece of the pipe

sw $t2, 0($t5) # paint the pipe

sub $t7, $t7, 4

add $t5, $t7, $t0 # t5 gets updated with a piece of the pipe

sw $t2, 0($t5) # paint the pipe

sub $t7, $t7, 4

add $t5, $t7, $t0 # t5 gets updated with a piece of the pipe

sw $t2, 0($t5) # paint the pipe

sub $t7, $t7, 120

bge $t7, 4, TopPipe

add $t7, $t7, 2040

add $t7, $t7, $s7

BotPipe:

add $t5, $t7, $t0 # t5 gets updated with a piece of the pipe

sw $t2, 0($t5) # paint the bird

add $t7, $t7, 4

add $t5, $t7, $t0 # t5 gets updated with a piece of the pipe

sw $t2, 0($t5) # paint the bird

add $t7, $t7, 4

add $t5, $t7, $t0 # t5 gets updated with a piece of the pipe

sw $t2, 0($t5) # paint the bird

add $t7, $t7, 120

ble $t7, 4094, BotPipe

add $t9, $t9, 4

bge $t9, 132, Reset

jr $ra

Reset:

beq $t3, $zero, continue

sub $t3, $t3, 25

continue:

li $a1, 12 # set max to 8

li $v0, 42 # set syscall to random int

syscall # get random int

li $t7,3968

reset1:

add $t5, $t7, $t0 # t5 gets updated with a piece of the pipe

sw $t1, 0($t5) # paint the bird

sub $t7, $t7, 128

bge $t7, $zero, reset1

li $t7, 3972

reset2:

add $t5, $t7, $t0 # t5 gets updated with a piece of the pipe

sw $t1, 0($t5) # paint the bird

sub $t7, $t7, 128

bge $t7, 4, reset2

li $t7, 3976

reset3:

add $t5, $t7, $t0 # t5 gets updated with a piece of the pipe

sw $t1, 0($t5) # paint the bird

sub $t7, $t7, 128

bge $t7, 8, reset3

mult $a0 $s6

mflo $s7

sub $t7, $t7, $s7

li $t9, 0

jr $ra

Exit:

lw $t1, blue

move $t4, $zero # $t4 stores the counter for the units

End:

add $t5, $t4, $t0 # t5 gets updated with the next unit

sw $t1, 0($t5) # paint the first unit on the second row blue. Why +128?

add $t4, $t4, 4 # increment to next unit

bne $t4, 4096, End # jump back if the screen is not fully painted

li $t7, 1960

lw $t1, yellow

G:

add $t5, $t7, $t0 # t5 gets updated with a piece of the pipe

sw $t1, 0($t5) # paint the bird

add $t7, $t7, 4

add $t5, $t7, $t0 # t5 gets updated with a piece of the pipe

sw $t1, 0($t5) # paint the bird

add $t7, $t7, 4

add $t5, $t7, $t0 # t5 gets updated with a piece of the pipe

sw $t1, 0($t5) # paint the bird

add $t7, $t7, 4

add $t5, $t7, $t0 # t5 gets updated with a piece of the pipe

sw $t1, 0($t5) # paint the bird

add $t7, $t7, 116

add $t5, $t7, $t0 # t5 gets updated with a piece of the pipe

sw $t1, 0($t5) # paint the bird

add $t7, $t7, 128

add $t5, $t7, $t0 # t5 gets updated with a piece of the pipe

sw $t1, 0($t5) # paint the bird

add $t7, $t7, 8

add $t5, $t7, $t0 # t5 gets updated with a piece of the pipe

sw $t1, 0($t5) # paint the bird

add $t7, $t7, 4

add $t5, $t7, $t0 # t5 gets updated with a piece of the pipe

sw $t1, 0($t5) # paint the bird

add $t7, $t7, 116

add $t5, $t7, $t0 # t5 gets updated with a piece of the pipe

sw $t1, 0($t5) # paint the bird

add $t7, $t7, 12

add $t5, $t7, $t0 # t5 gets updated with a piece of the pipe

sw $t1, 0($t5) # paint the bird

add $t7, $t7, 116

add $t5, $t7, $t0 # t5 gets updated with a piece of the pipe

sw $t1, 0($t5) # paint the bird

add $t7, $t7, 4

add $t5, $t7, $t0 # t5 gets updated with a piece of the pipe

sw $t1, 0($t5) # paint the bird

add $t7, $t7, 4

add $t5, $t7, $t0 # t5 gets updated with a piece of the pipe

sw $t1, 0($t5) # paint the bird

add $t7, $t7, 4

add $t5, $t7, $t0 # t5 gets updated with a piece of the pipe

sw $t1, 0($t5) # paint the bird

beq $a3, $zero, done

move $a3, $zero

li $t7, 1984

j G

done:

li $v0, 10 # terminate the program gracefully

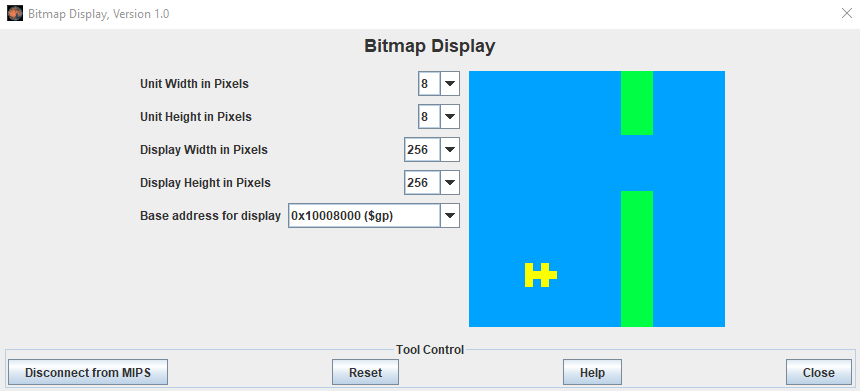
syscall

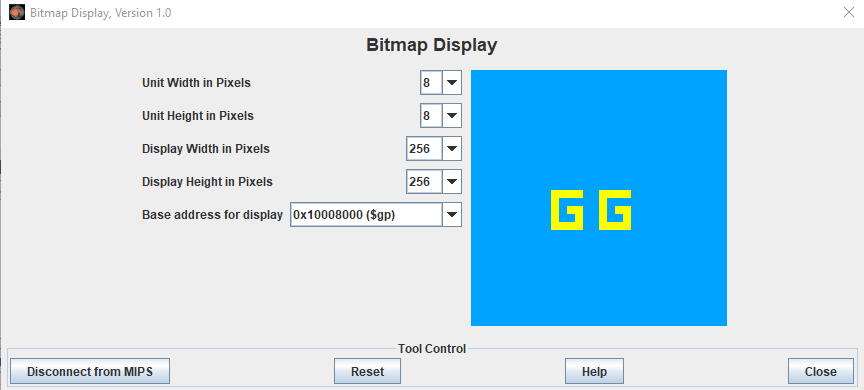
# **INTERFACES:**

We used bitmap display for the output of the program. To control the movements of the bird we used a keyboard simulator. The background, the pipes and the bird as well as their movements with respect to the bird and the keys entered are the most significant part of the game.

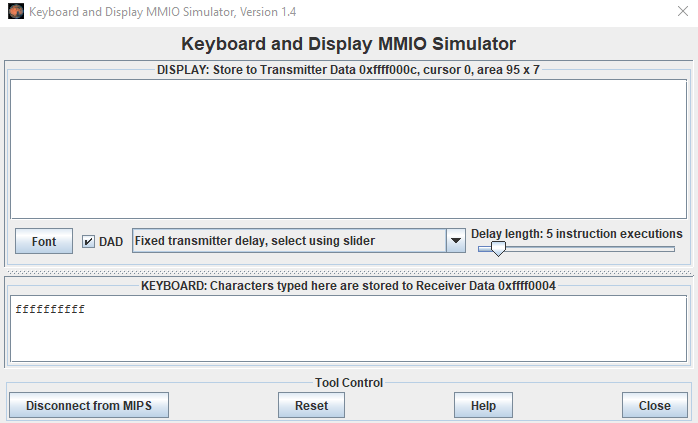
**Output:**

**Bitmap Display:**

****

****

**Keyboard Simulator:**

****

# **CONCLUSION:**

Flappybird is a simple game of the infinite level type. People of almost every age can play this game. It can be played in their leisure time for entertainment. It is also an easy-to-play game which does not require any complex procedure to play it. It is an addictive game and a person can play it for hours. It is possible to play this game on android.