



RHYTHM MASTER

CSEE4840 Embedded System

Presenters:

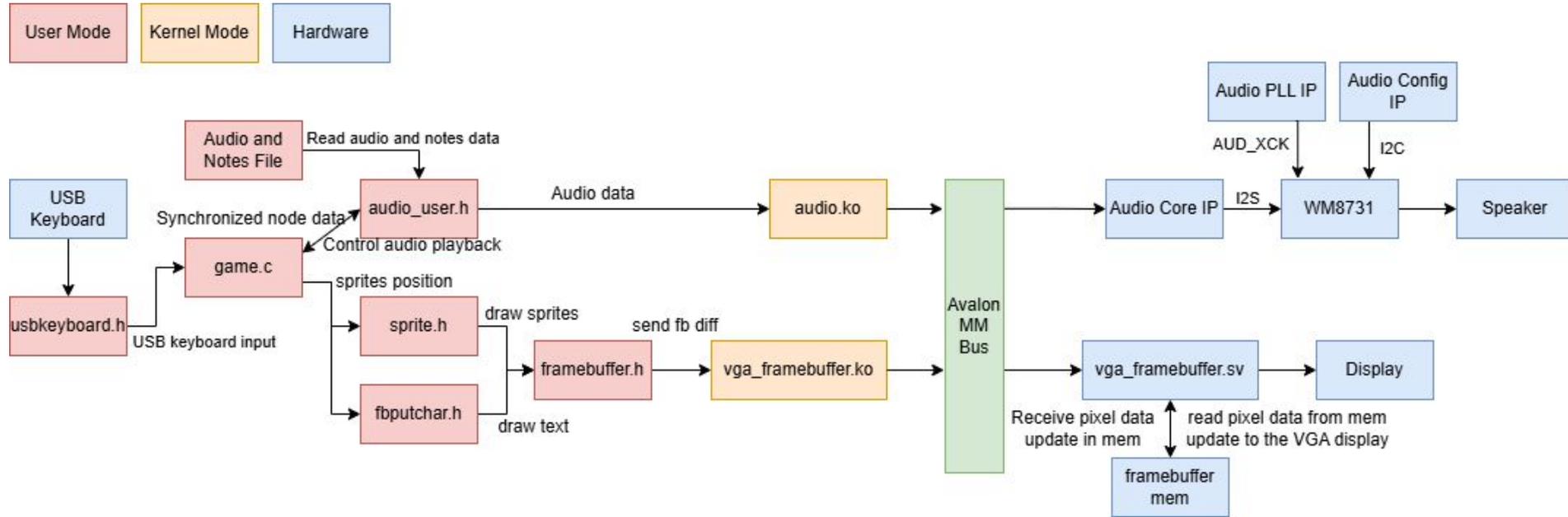
Yangyang Zhang

Junfeng Zou

Hongrui Huang

May 13, 2025

Design Overview



Audio HW Control

Use	Connections	Name	Description	Export	Clock	Base	End	IRQ
<input checked="" type="checkbox"/>		clk_0	Clock Source					
		clk_in	Clock Input	clk	exported			
		clk_in_reset	Reset Input	reset				
		clk	Clock Output	clk_0	clk_0			
		clk_reset	Reset Output	reset				
<input checked="" type="checkbox"/>		hps_0	Arria V/Cyclone V Hard Proce...					
		h2f_user1_clock	Clock Output	hps_ddr3	hps_0_h2...			
		memory	Conduit	hps				
		hps_io	Conduit					
		h2f_reset	Reset Output					
		h2f_axi_clock	Clock Input	clk_0	clk_0			
		h2f_axi_master	AXI Master	[h2f_axi_...				
		f2h_axi_clock	Clock Input	clk_0	clk_0			
		f2h_axi_slave	AXI Slave	[f2h_axi_...				
		h2f_lw_axi_clock	Clock Input	clk_0	clk_0			
		h2f_lw_axi_master	AXI Master	[h2f_lw_a...				
		f2h_irq0	Interrupt Receiver			IRQ 0	IRQ 31	
		f2h_irq1	Interrupt Receiver			IRQ 0	IRQ 31	
<input checked="" type="checkbox"/>		audio_pll_0	Audio Clock for DE-series Boa...					
		ref_clk	Clock Input	clk_0	clk_0			
		ref_reset	Reset Input					
		audio_clk	Clock Output	audio_pll...	audio_pll...			
		reset_source	Reset Output					
<input checked="" type="checkbox"/>		clock_bridge_0	Clock Bridge					
		in_clk	Clock Input	audio_p...	audio_p...			
		out_clk	Clock Output	clock_bri...	clock_bri...			
		out_clk_1	Clock Output					
<input checked="" type="checkbox"/>		audio_and_video_0	Audio and Video Config					
		clk	Clock Input	clk_0	clk_0			
		reset	Reset Input	[clk]	[clk]			
		avalon_av_config...	Avalon Memory Mapped Slave	aud_config	[clk]			
		external_interface	Conduit					
<input checked="" type="checkbox"/>		audio_0	Audio					
		clk	Clock Input	clock_br...	clock_br...			
		reset	Reset Input	[clk]	[clk]			
		avalon_audio_sla...	Avalon Memory Mapped Slave	aud	[clk]	0x0000_0000	0x0000_000f	
		interrupt_interface	Interrupt Sender					
		external_interface	Conduit					

System: soc_system **Path:** audio_pll_0

Audio Clock for DE-series Boards
altera_up_avalon_audio_pll

Settings

Reference clock: 50.0 MHz

Audio Clock Frequency: 12.288 MHz

System: soc_system **Path:** audio_and_video_config_0

Audio and Video Config
altera_up_avalon_audio_and_video_config

Components

Audio/Video Device: On-Board Peripherals

DE Board: DE1-SoC

Auto Initialize Device(s)

Auto Initialization Parameters for Audio

Audio In Path: Line In to ADC

Audio Out - Enable DAC Output

Audio Out - Microphone Bypass

Audio Out - Line In Bypass

Data Format: Left Justified

Bit Length: 16

Sampling Rate: 48 kHz

System: soc_system **Path:** audio_0

Audio
altera_up_avalon_audio

Interface Settings

Avalon Type: Memory Mapped

Audio Direction

Audio In

Audio Out

Data Format

Data Width: 16

Audio Control & HW/SW Interface

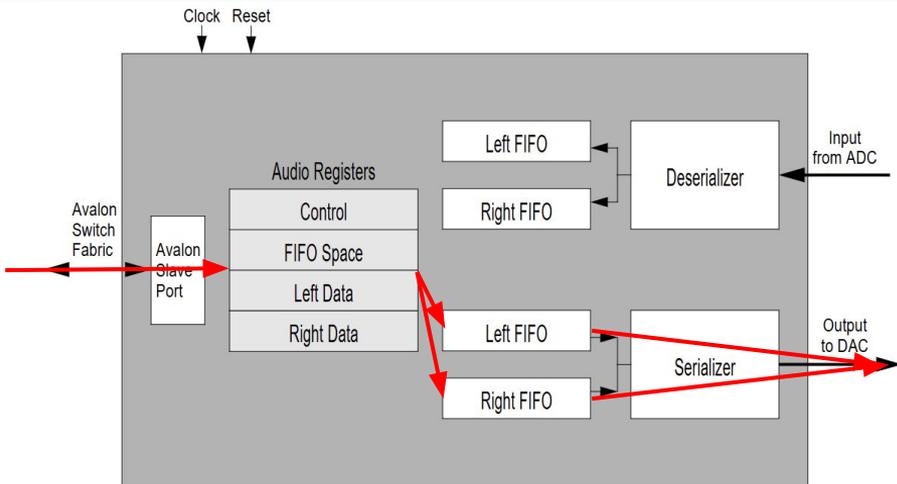


Figure 1. Block diagram for Audio core with Memory-Mapped Interface

Offset in bytes	Register Name	R/W	Bit Description									
			31...24	23...16	15...10	9	8	7...4	3	2	1	0
0	control	RW		(1)		WI	RI	(1)	CW	CR	WE	RE
4	fifospace	R	WSLC	WSRC		RALC				RARC		
8	leftdata	RW (2)	Left Data									
12	rightdata	RW (2)	Right Data									

Notes on Table 1:

- (1) Reserved. Read values are undefined. Write zero.
- (2) Only reads incoming audio data and writes outgoing audio data.

Bit number	Bit name	Read/Write	Description
0	RE	R/W	Interrupt-enable bit for read interrupts. If the RE bit is set to 1 and both the left and right channel read FIFOs contain data, the Audio core generates an interrupt request (IRQ).
1	WE	R/W	Interrupt-enable bit for write interrupts. If the WE bit is set to 1 and both the left and right channel write FIFOs have space available for more data, the Audio core generates an interrupt request (IRQ).
2	CR	R/W	Clears the Audio core's Input FIFOs, when the bit is 1. Clear remains active until specifically set to zero.
3	CW	R/W	Clears the Audio core's Output FIFOs, when the bit is 1. Clear remains active until specifically set to zero.
8	RI	R	Indicates that a read interrupt is pending.
9	WI	R	Indicates that a write interrupt is pending.

4.1.2 Fifospace Register

The `fifospace` register fields `WSLC` (b_{31-24}) and `WSRC` (b_{23-16}) indicate the number of words available (i.e., the amount of empty space) for outgoing data in the left and right channel FIFOs, respectively, while `RALC` (b_{15-8}) and `RARC` (b_{7-0}) indicate the number of words of incoming audio data in the left and right channel FIFOs, respectively. When all of the outgoing and incoming FIFOs are empty, the `fifospace` register will hold `WSLC` = `WSRC` = 128, and `RALC` = `RARC` = 0.

4.1.3 Leftdata Register

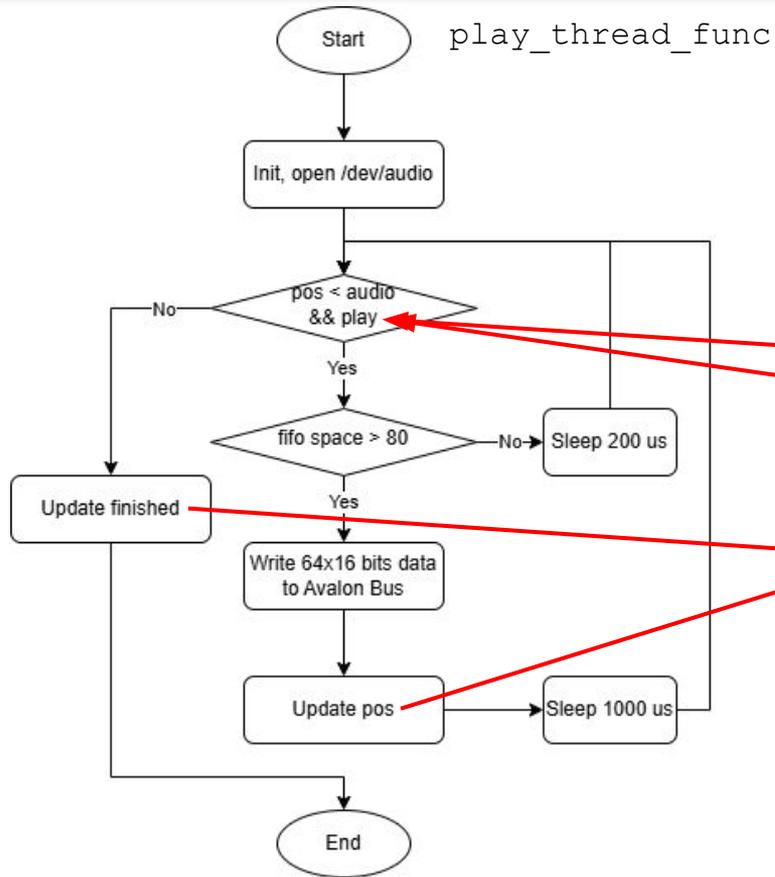
Each FIFO can store up to 128 32-bit words.
 $2 * 128 * 32 = 8192$ bits

The `leftdata` register is readable only for Audio In and writable only for Audio Out. It stores the data coming from or going to the left channel. The data is always flush right, i.e., the LSB is b_0 of the `leftdata` register.

4.1.4 Rightdata Register

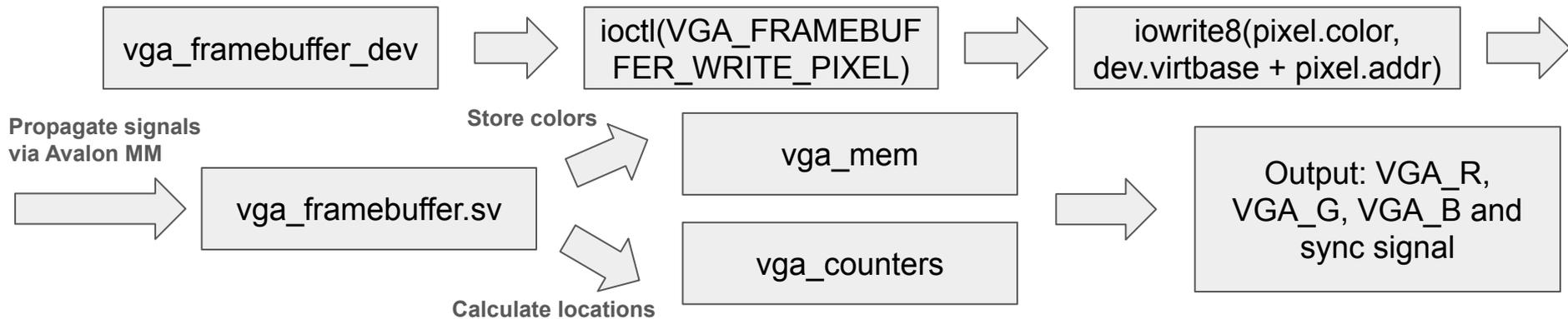
The `rightdata` register is readable only for Audio In and writable only for Audio Out. It stores the data coming from or going to the right channel. The data is always flush right, i.e., the LSB is b_0 of the `rightdata` register.

Audio SW Control



```
int load_audio(const char *filename);
int play_audio(int time_ms);
int pulse_audio(void);
int reset_audio(void);
int audio_time(void);
int is_audio_finished(void);
void free_audio(void);
```

VGA HW Control



System: soc_system Path: vga_framebuffer_0.avalon_slave_0

Use	Connections	Name	Description	Export	Clock	Base	End	IRQ
<input checked="" type="checkbox"/>		clk_0	Clock Source	clk	exported			
		clk_in	Clock Input	reset	clk_0			
		clk_in_reset	Reset Input	Double-click to				
		clk	Clock Output	Double-click to				
		clk_reset	Reset Output					
<input checked="" type="checkbox"/>		hps_0	Arria V/Cyclone V Hard Proce...					
		h2f_user1_clock	Clock Output	Double-click to	hps_0_h2...			
		memory	Conduit	hps_ddr3				
		hps_io	Conduit	hps				
		h2f_reset	Reset Output	Double-click to				
		h2f_axi_clock	Clock Input	Double-click to	clk_0			
		h2f_axi_master	AXI Master	Double-click to	[h2f_axi_...			
		f2h_axi_clock	Clock Input	Double-click to	clk_0			
		f2h_axi_slave	AXI Slave	Double-click to	[f2h_axi_...			
		h2f_lw_axi_clock	Clock Input	Double-click to	clk_0			
		h2f_lw_axi_master	AXI Master	Double-click to	[h2f_lw_a...			
		f2h_irq0	Interrupt Receiver	Double-click to		IRQ 0		IRQ 31
		f2h_irq1	Interrupt Receiver	Double-click to		IRQ 0		IRQ 31
<input checked="" type="checkbox"/>		vga_framebuffer...	vga_framebuffer					
		clock	Clock Input	Double-click to	clk_0			
		reset	Reset Input	Double-click to	[clock]			
		avalon_slave_0	Avalon Memory Mapped Slave		[clock]	0x0000_0000	0x0007_ffff	
		vga	Conduit	vga	[clock]			

Resources used:
vga_mem
 $307200 * 8 \text{ bits} = 2457600 \text{ bits}$

VGA HW/SW Interface

Offset	Register name	R/W	Description
0	pixel[0]	W	8-bit color index at pixel[0]
1	pixel[1]	W	8-bit color index at pixel[1]
2	pixel[2]	W	8-bit color index at pixel[2]
3	pixel[2]	W	8-bit color index at pixel[2]
...	pixel[...]	W	8-bit color index at pixel[...]
307198	pixel[307198]	W	8-bit color index at pixel[307198]
307199	pixel[307199]	W	8-bit color index at pixel[307199]

Sprite SW Implementation



```
sprite_t *sprite_load(const char *filename);  
void sprite_draw(sprite_t *sprite, int dst_x, int dst_y, int target_w, int target_h);  
void draw_bg (int combo, int score);  
void draw_combo(int combo);  
void draw_score(int score);  
void draw_score_final(int score);  
void split_digits(int num, int digits[5]);  
void sprite_free(sprite_t *sprite);
```

Framebuffer SW Control

fbputchar.h

```
void fbputchar(int x, int y, char c, int color_index);  
void fbputs(int x, int y, const char *s, int color_index);
```

sprite: logo

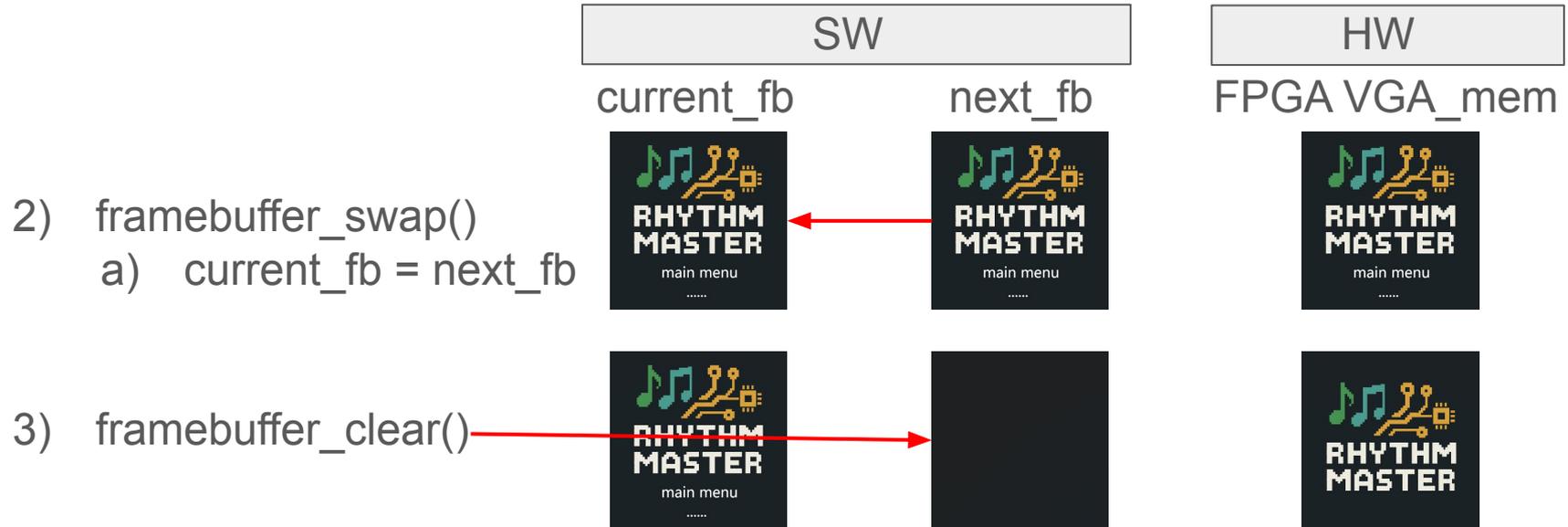


1) sprite_draw(...)
fbputs(...)

2) framebuffer_swap()
a) Send diff via Avalon



Framebuffer SW Control



Ready for next frame ...

Game Logic Key Functions

Game Menu

- Main Menu and Pause Menu
- Provide difficulty selection and song selection

Note Rendering

- Pre-processed beat maps
- Pre-processed note animation arrays

Hit Feedback

- Monitor keyboard for press detection
- Eliminate note and give feedback (MISS/GREAT/PERFECT)

Scoring

- $\text{Score} = \text{GREAT} * 30 + \text{PERFECT} * 50 + \text{combo bonus}$
- Level Feedback (S/A/B/C/D)

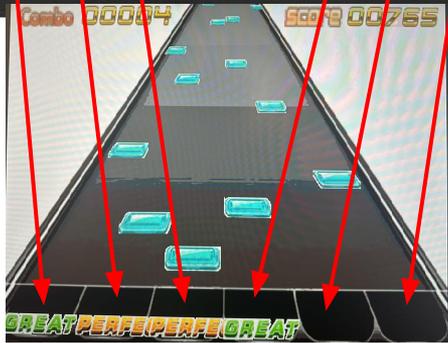
Game Controller

- USB Keyboard

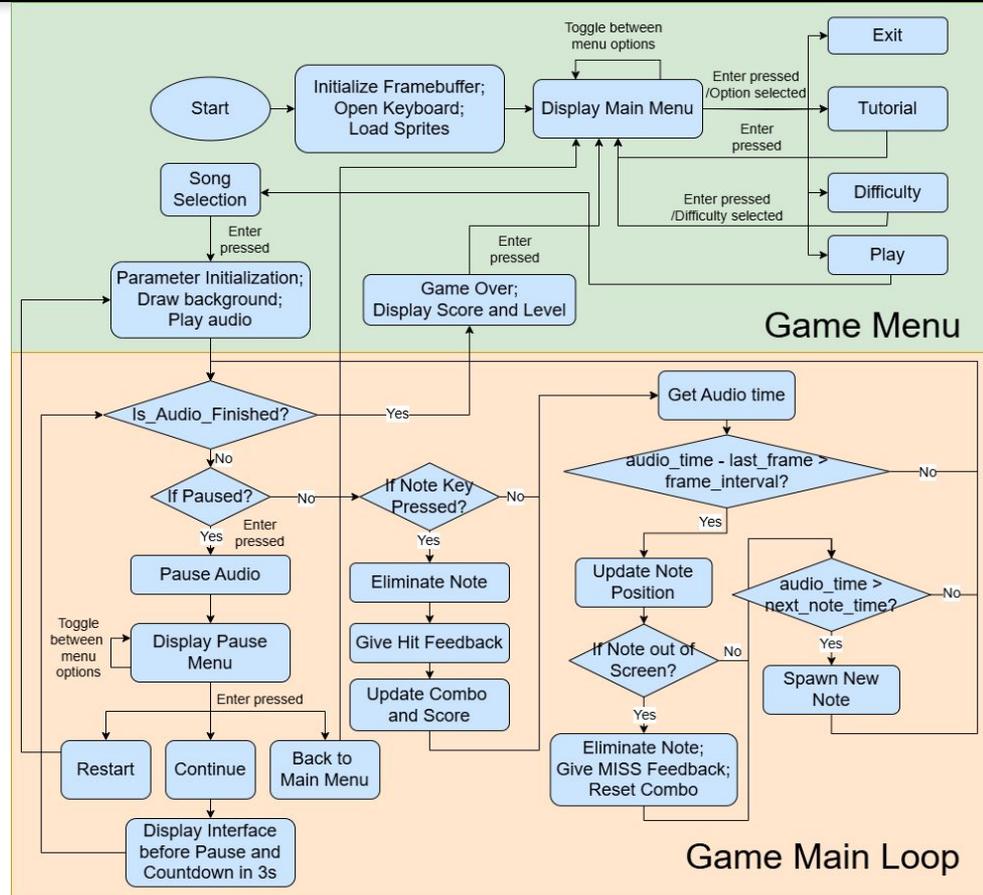


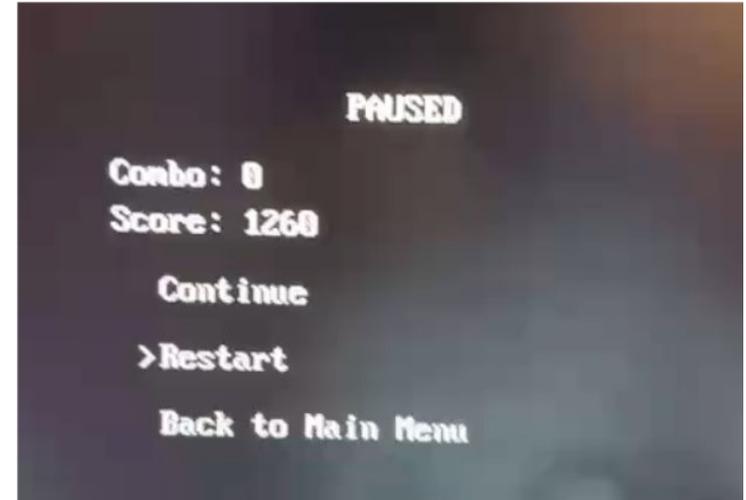
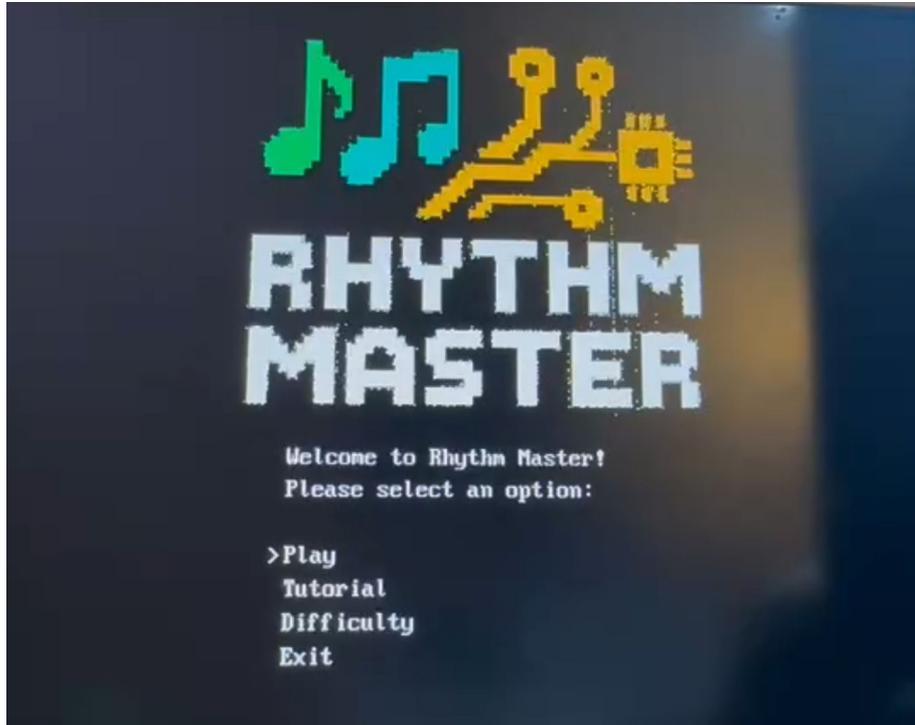
Confirm selection/Pause game

Move cursor up/down



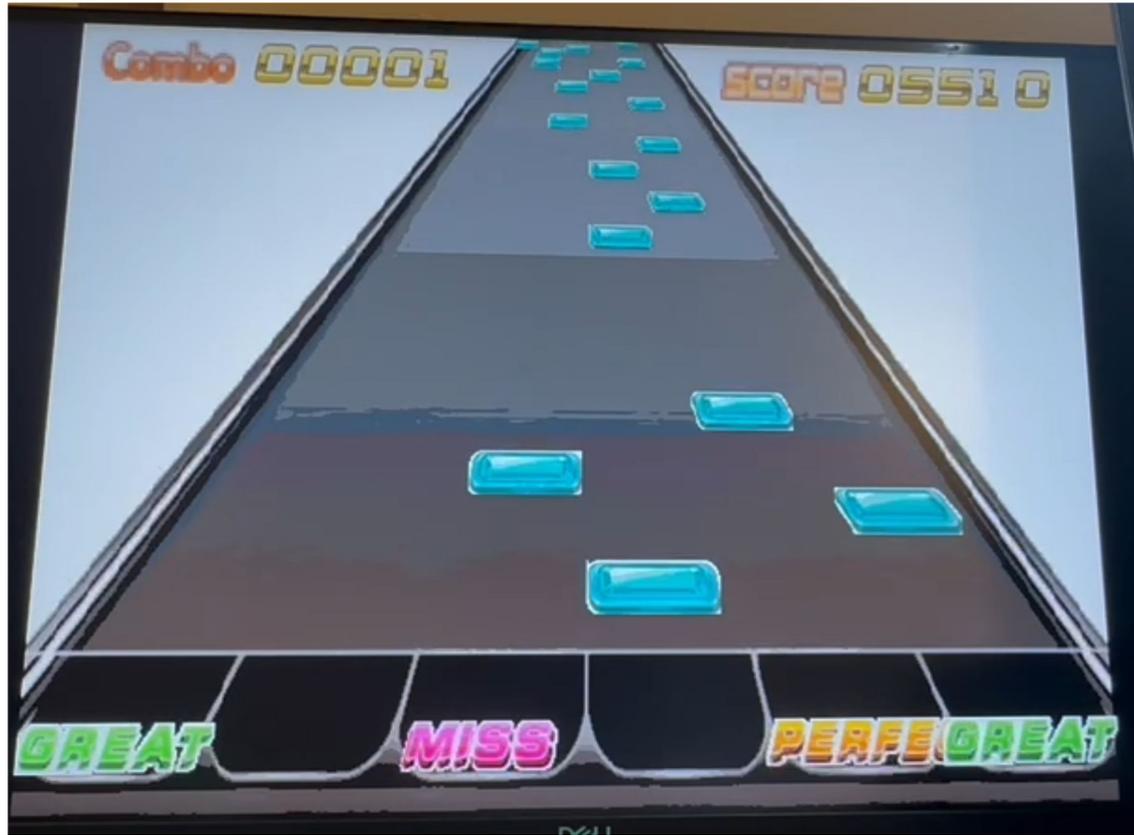
Game Logic Flowchart





Main Menu, Song Selection
Menu, and Pause Menu

Demo



Playing Interface and
Final Score Level