FINAL PROJECT:

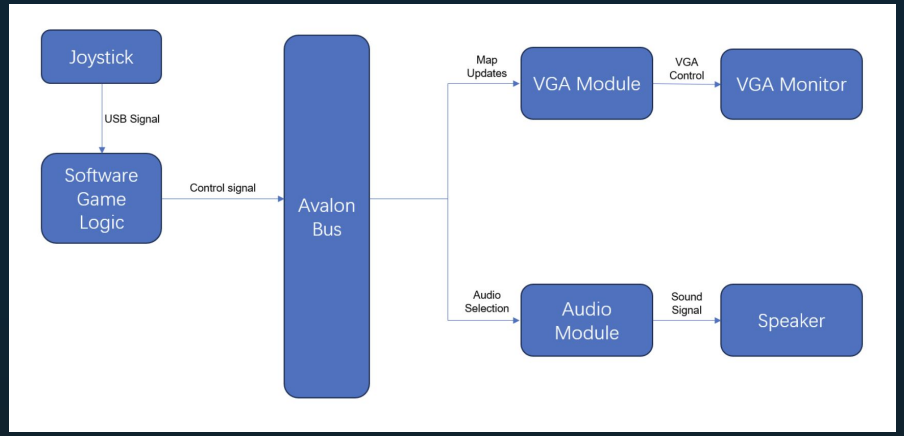
BOMBERMAN

Natalie Hughes, Qian Zhao, Shiyan Wang

PROJECT INTRODUCTION

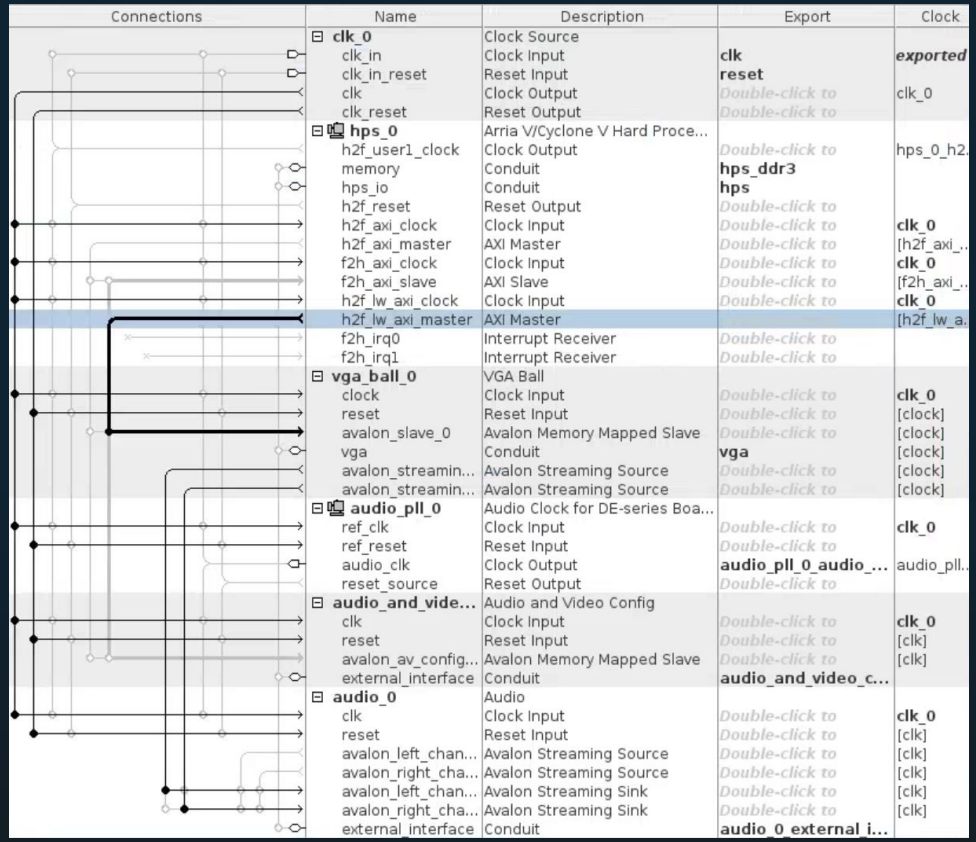
- Iconic 2D multiplayer game by Hudson Soft.
- Two players battle in grid-based mazes.
- Place bombs to:
  - Destroy obstacles.
  - Outmaneuver opponents.
- Mazes have destructible and indestructible walls.
- Bombs explode to trap or eliminate opponents.
- Game ends when one player eliminates the other.

SYSTEM ARCHITECTURE



- USB controllers interface with game logic
- vga.c driver relays control signals to FPGA, updating VGA monitor graphics; vga_display.sv handles sprite rendering.
- WM8731 CODEC manages audio output.
- FPGA registers track game state and ensure consistent rule application.

Qsys connection

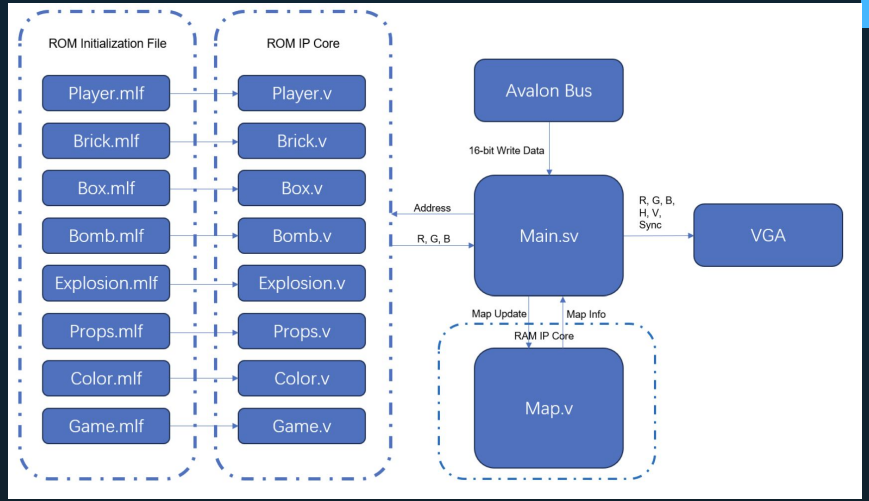


Connections	Name	Description	Export	Clock
	⊟ **clk_0**	Clock Source		
	clk_in	Clock Input	**clk**	***exported***
	clk_in_reset	Reset Input	**reset**	
	clk	Clock Output	*Double-click to*	clk_0
	clk_reset	Reset Output	*Double-click to*	
	⊟ **hps_0**	Arria V/Cyclone V Hard Proce...		
	h2f_user1_clock	Clock 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		[h2f_lw_a...
	f2h_irq0	Interrupt Receiver	*Double-click to*	
	f2h_irq1	Interrupt Receiver	*Double-click to*	
	⊟ **vga_ball_0**	VGA Ball		
	clock	Clock Input	*Double-click to*	**clk_0**
	reset	Reset Input	*Double-click to*	[clock]
	avalon_slave_0	Avalon Memory Mapped Slave	*Double-click to*	[clock]
	vga	Conduit	**vga**	[clock]
	avalon_streamin...	Avalon Streaming Source	*Double-click to*	[clock]
	avalon_streamin...	Avalon Streaming Source	*Double-click to*	[clock]
	⊟ **audio_pll_0**	Audio Clock for DE-series Boa...		
	ref_clk	Clock Input	*Double-click to*	**clk_0**
	ref_reset	Reset Input	*Double-click to*	
	audio_clk	Clock Output	**audio_pll_0_audio_...**	audio_pll...
	reset_source	Reset Output	*Double-click to*	
	⊟ **audio_and_vide...**	Audio and Video Config		
	clk	Clock Input	*Double-click to*	**clk_0**
	reset	Reset Input	*Double-click to*	[clk]
	avalon_av_config...	Avalon Memory Mapped Slave	*Double-click to*	[clk]
	external_interface	Conduit	**audio_and_video_c...**	
	⊟ **audio_0**	Audio		
	clk	Clock Input	*Double-click to*	**clk_0**
	reset	Reset Input	*Double-click to*	[clk]
	avalon_left_chan...	Avalon Streaming Source	*Double-click to*	[clk]
	avalon_right_cha...	Avalon Streaming Source	*Double-click to*	[clk]
	avalon_left_chan...	Avalon Streaming Sink	*Double-click to*	[clk]
	avalon_right_cha...	Avalon Streaming Sink	*Double-click to*	[clk]
	external_interface	Conduit	**audio_0_external_i...**	

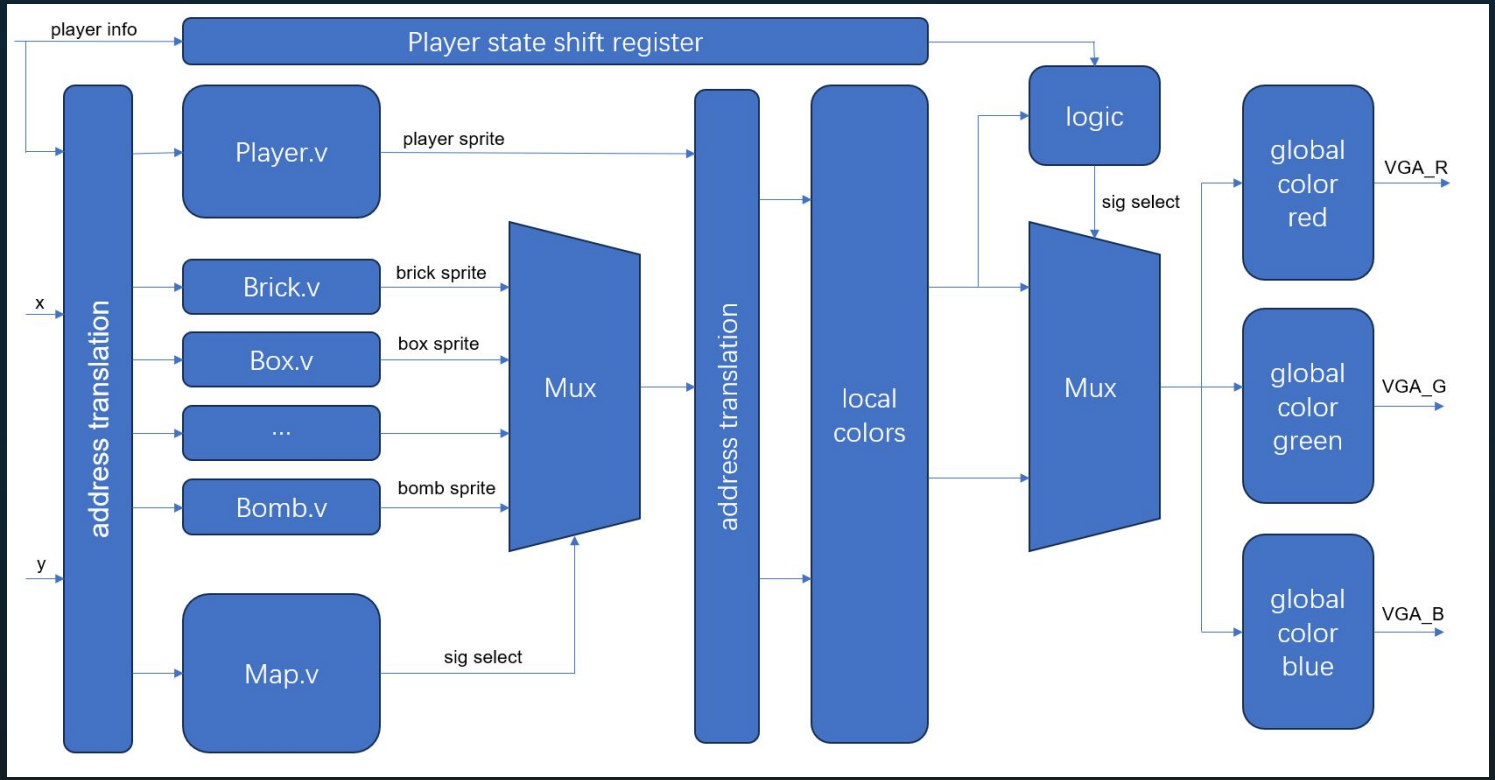
- audio_0: ip core to feed data to the codec
- audio_and_video_config: ip core to configure the codec (data width, sample rate, etc)
- audio_pll: provides 12.288MHz driver clock for codec
- vga_ball: add two streaming sources output to provide data to audio_0

HARDWARE VIDEO

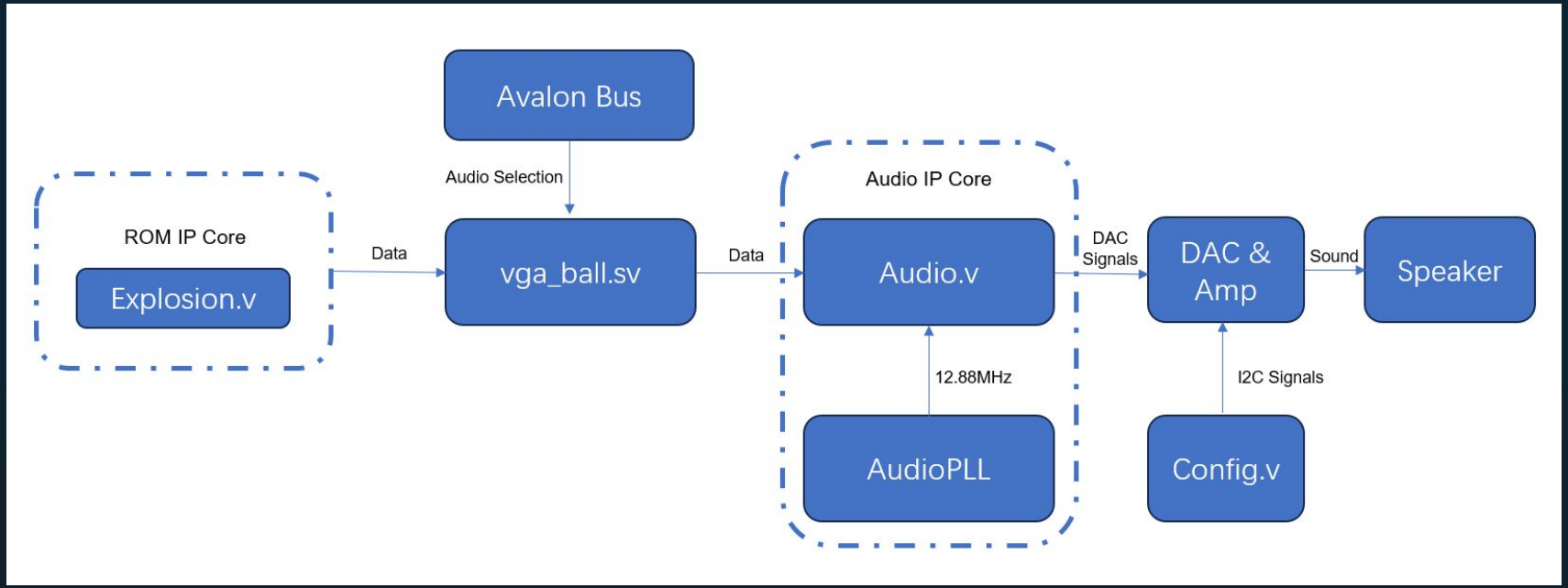
- Handles VGA display output and game graphics rendering using ROM modules for sprites and map elements, updating pixel positions
- submodule generates synchronization signals, managing timing for pixel updates.
- Player rendering uses central pixel location with four directional sprites and color maps, managing two display layers: players on top, map on bottom.



HARDWARE VIDEO - CLOSER LOOK



HARDWARE - AUDIO



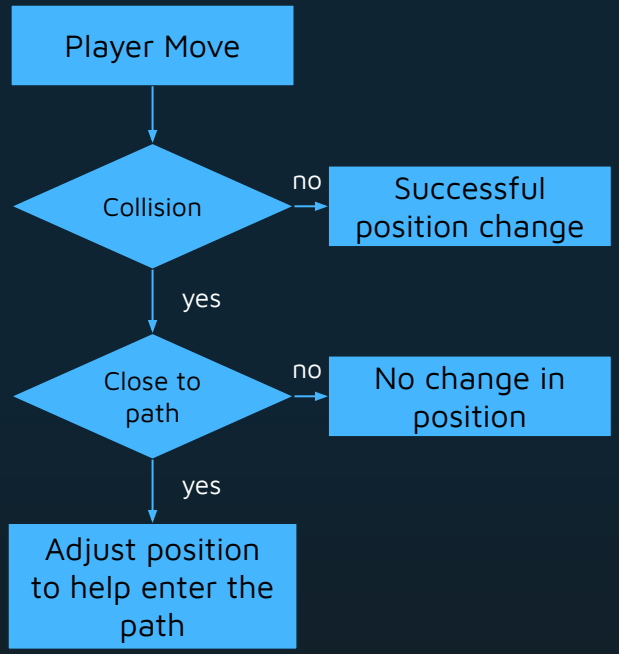
SOFTWARE - USER INPUT

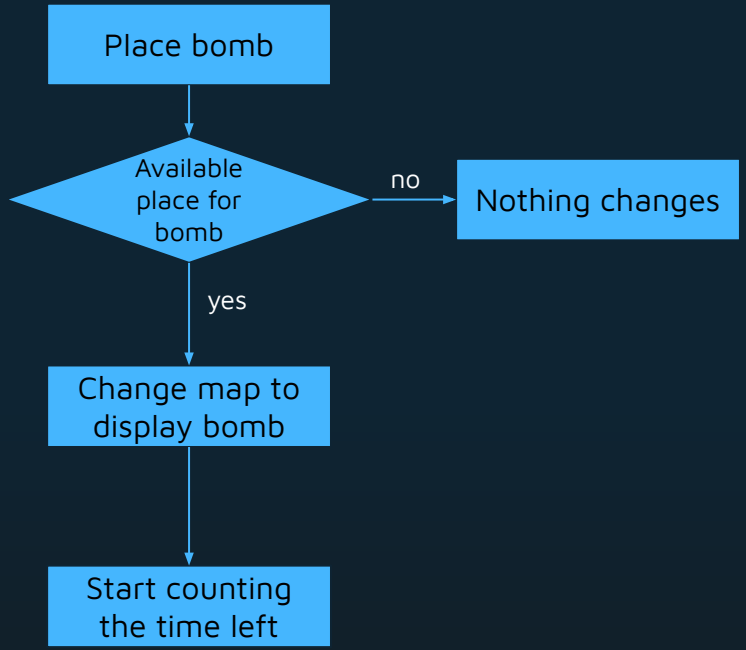
- Manages USB game controllers (idProduct 17) via libusb, handling input for movement and bomb placement.
- Debounce counters ensure single presses are registered; both controllers processed in a single loop using libusb_interrupt_transfer.
- Key functions detect and interact with controllers, reading 7-byte protocol messages to discern player actions in real-time.
- Place bombs with A and move with arrow keys

SOFTWARE - GAME LOGIC

- Main game loop processes:
  - Player movement and bomb placement
  - Power-up collection
  - Game state updates
  - Bomb detonation and explosion propagation
  - Map changes synchronization with display hardware
- Loop continues until a player dies
- Displays game over screen and cleans up resources at the end

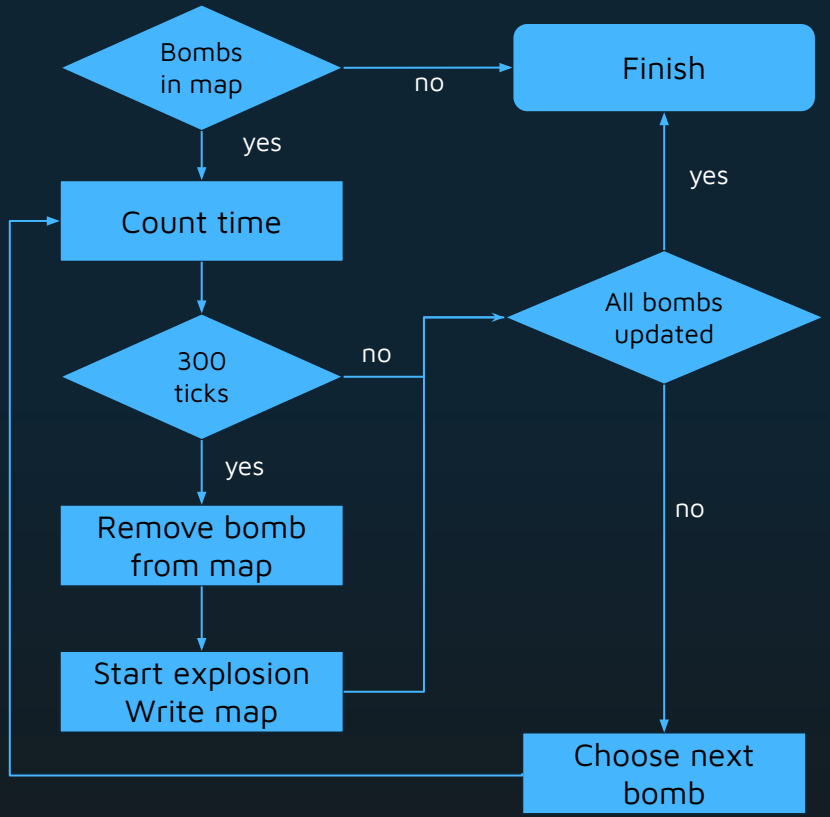
GAME LOGIC - OVERVIEW



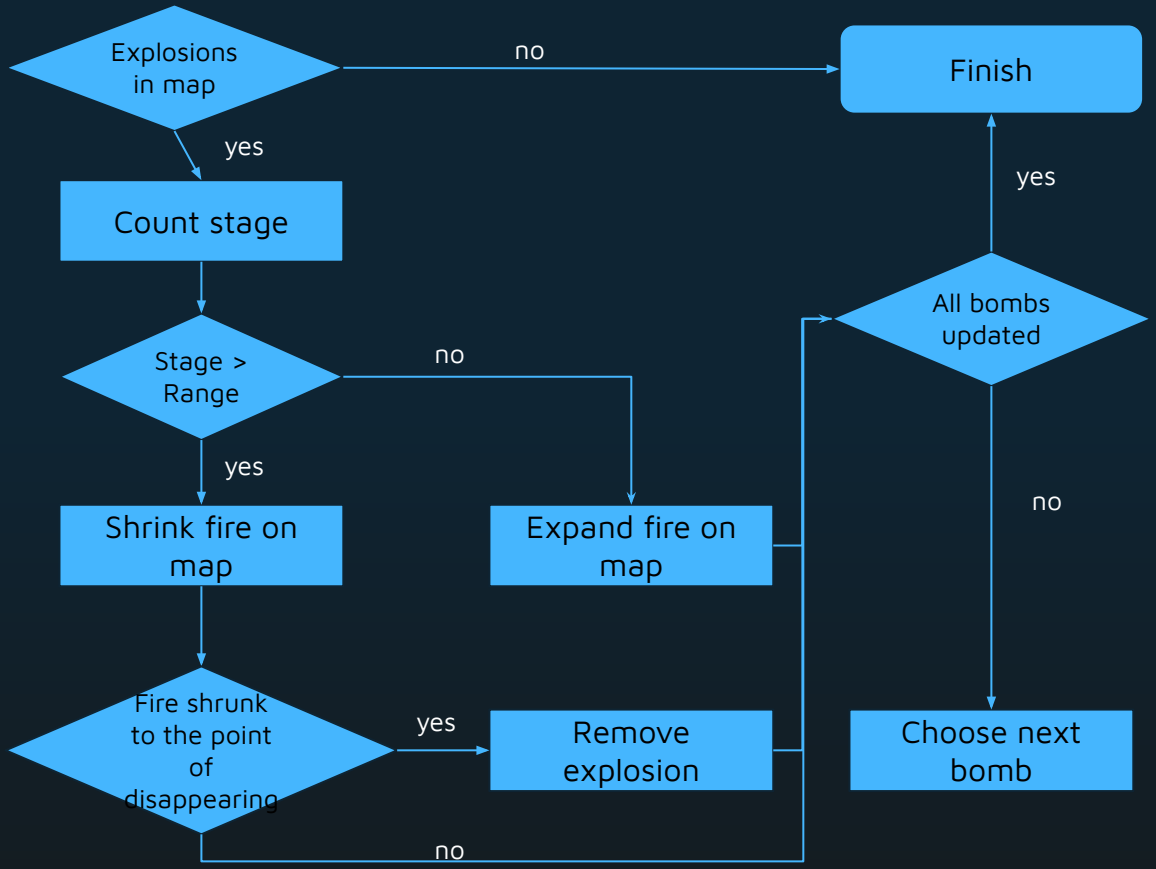


The game ends when a player collides with the flame and is eliminated, and the other player wins. Players dying at the same time is a tie.

GAME LOGIC - BOMBS



GAME LOGIC - EXPLOSIONS

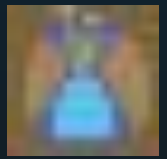


GAME LOGIC - PROPS

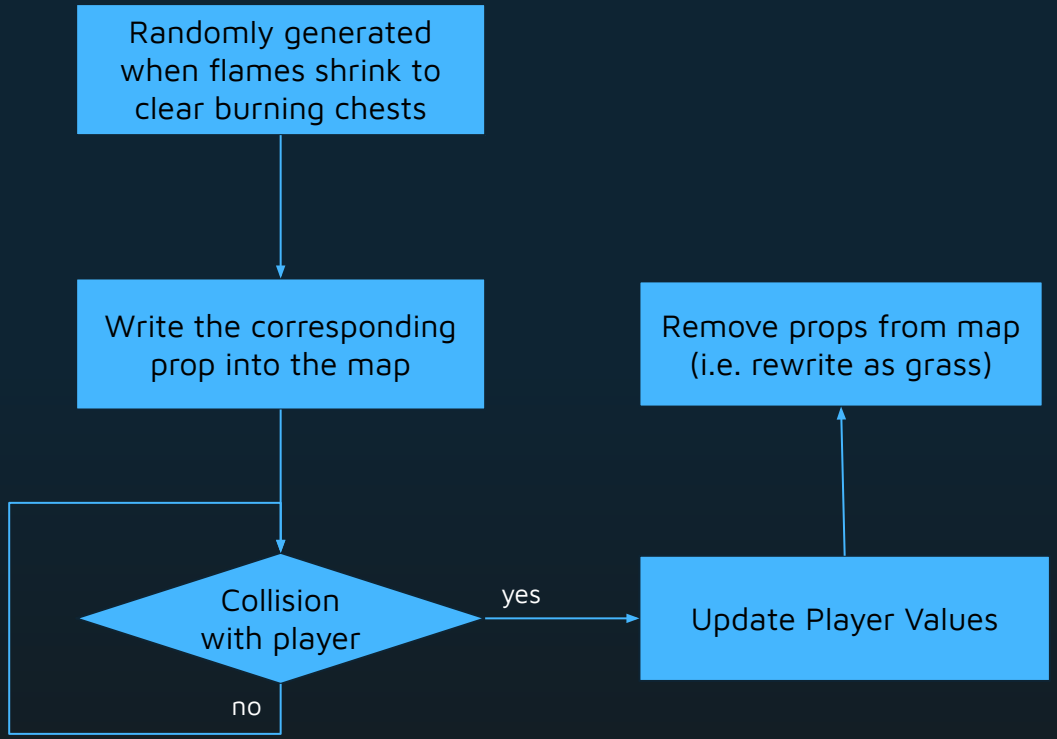
Type 1: Increase the number of bombs the player can place

Type 2: Expand the blast range of bombs placed by player

Type 3: Increase player movement speed



GAME LOGIC - PROPS



HARDWARE-SOFTWARE INTERFACE

Register	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	P0_pose					P0_moving	P0_x									
1						P0_facing		P0_y								
2	P1_pose					P1_moving	P1_x									
3						P1_facing		P1_y								
4	en	Tile_type				Position										
5	en	Tile_type				Position										
6	en	Tile_type				Position										
7	en	Tile_type				Position										
8	en	Tile_type				Position										
9	en	Tile_type				Position										
10	en	Tile_type				Position										
11	en	Tile_type				Position										
12	en	Tile_type				Position										
13	en	Tile_type				Position										
14	en	Tile_type				Position										
15	en	Tile_type				Position										
16	en	Tile_type				Position										
17																Audio_play

THANK YOU!