Eshkol Multimedia System Architecture

Technical Implementation Specification

Version: 1.0
Date: 2025-12-09
Status: Implementation Specification
Prerequisite: POINTER_CONSOLIDATION_IMPLEMENTATION.md must be completed first

Table of Contents

1. Overview

The Eshkol Multimedia System provides type-safe abstractions for:

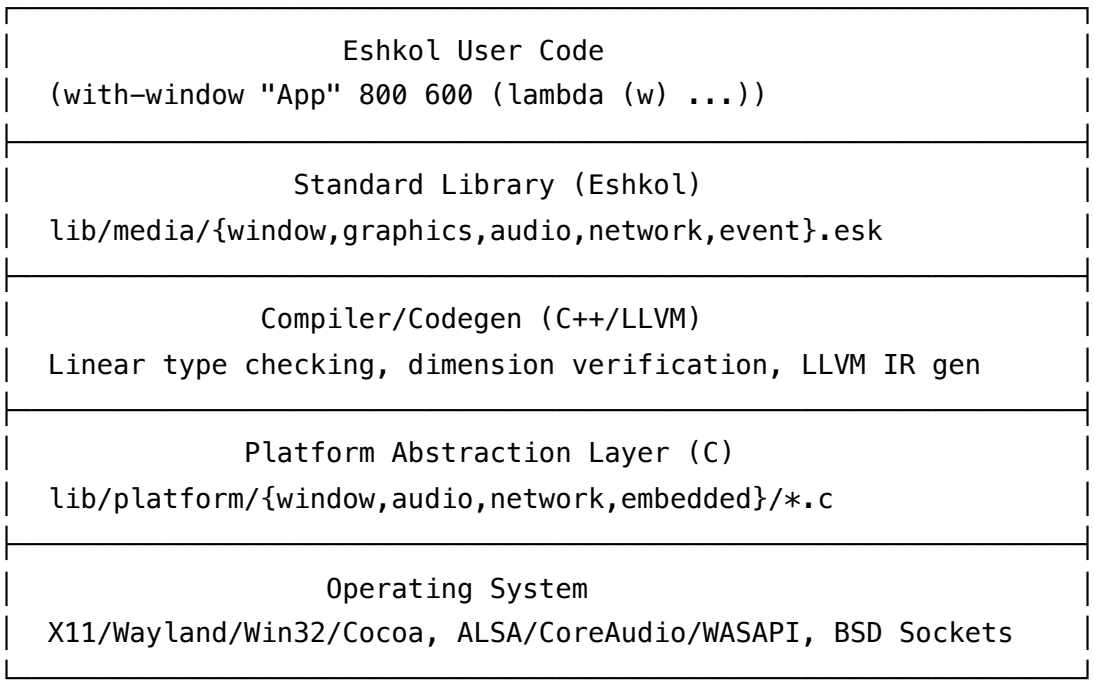
- **Graphics**: Windows, surfaces, pixel buffers, drawing primitives
- **Audio**: Playback, recording, synthesis, real-time processing
- **Networking**: TCP/UDP sockets, data transmission

- **Hardware I/O**: GPIO, I2C, SPI for embedded/robotics applications
- **Events**: Input handling, system notifications

1.1 Design Principles

1. **Type Safety**: All resources are typed; dimension and format errors caught at compile time
2. **Linear Resources**: Hardware handles use linear types for guaranteed cleanup
3. **Zero Overhead**: Proof erasure ensures no runtime cost for type information
4. **Platform Independence**: Same Eshkol code runs on Linux, macOS, Windows, embedded
5. **Integration**: Seamless interop with autodiff, tensors, and HoTT type system

1.2 Architecture Layers



2. Type System Integration

2.1 Multimedia Type Constants

These types are defined in `inc/eshkol/eshkol.h` after pointer consolidation:

```
// Multimedia types (from pointer consolidation)
ESHKOL_VALUE_HANDLE      = 16,  // Managed resource handles
ESHKOL_VALUE_BUFFER      = 17,  // Typed data buffers
ESHKOL_VALUE_STREAM      = 18,  // Lazy data streams
ESHKOL_VALUE_EVENT       = 19,  // Input/system events
```

2.2 Subtype Enumerations

```
// ═══════════════════════════════════════════════════════════
// HANDLE SUBTYPES (type = ESHKOL_VALUE_HANDLE = 16)
// ═══════════════════════════════════════════════════════════
typedef enum {
    // Graphics
    HANDLE_SUBTYPE_WINDOW       = 0,   // GUI window with surface
    HANDLE_SUBTYPE_SURFACE      = 1,   // Drawing surface (may be offscreen)
    HANDLE_SUBTYPE_DISPLAY      = 2,   // Display/monitor handle

    // Audio
    HANDLE_SUBTYPE_AUDIO_OUT    = 3,   // Audio output device
    HANDLE_SUBTYPE_AUDIO_IN     = 4,   // Audio input device
    HANDLE_SUBTYPE_MIDI_OUT     = 5,   // MIDI output
    HANDLE_SUBTYPE_MIDI_IN      = 6,   // MIDI input

    // Network
    HANDLE_SUBTYPE_SOCKET_TCP   = 7,   // TCP socket
    HANDLE_SUBTYPE_SOCKET_UDP   = 8,   // UDP socket
    HANDLE_SUBTYPE_SOCKET_UNIX  = 9,   // Unix domain socket

    // File I/O
    HANDLE_SUBTYPE_FILE         = 10,  // File handle
    HANDLE_SUBTYPE_DIRECTORY    = 11,  // Directory handle
    HANDLE_SUBTYPE_PIPE         = 12,  // Pipe handle

    // Embedded/Robotics
    HANDLE_SUBTYPE_GPIO         = 13,  // GPIO pin
    HANDLE_SUBTYPE_I2C          = 14,  // I2C bus
    HANDLE_SUBTYPE_SPI          = 15,  // SPI bus
    HANDLE_SUBTYPE_UART         = 16,  // UART/serial port
    HANDLE_SUBTYPE_PWM          = 17,  // PWM output
    HANDLE_SUBTYPE_ADC          = 18,  // Analog-to-digital converter
    HANDLE_SUBTYPE_DAC          = 19,  // Digital-to-analog converter

    // Robotics
    HANDLE_SUBTYPE_MOTOR        = 20,  // Motor/actuator
    HANDLE_SUBTYPE_SERVO        = 21,  // Servo motor
    HANDLE_SUBTYPE_SENSOR       = 22,  // Generic sensor
    HANDLE_SUBTYPE_CAMERA       = 23,  // Camera device
    HANDLE_SUBTYPE_LIDAR        = 24,  // LIDAR sensor
    HANDLE_SUBTYPE_IMU          = 25,  // Inertial measurement unit
```

```
    // System
    HANDLE_SUBTYPE_PROCESS      = 26,  // Child process
    HANDLE_SUBTYPE_THREAD       = 27,  // Thread handle
    HANDLE_SUBTYPE_TIMER        = 28,  // System timer
    HANDLE_SUBTYPE_SEMAPHORE    = 29,  // Semaphore
    HANDLE_SUBTYPE_MUTEX        = 30,  // Mutex
} handle_subtype_t;

// ═══════════════════════════════════════════════════════════
// BUFFER SUBTYPES (type = ESHKOL_VALUE_BUFFER = 17)
// ═══════════════════════════════════════════════════════════
typedef enum {
    // Raw data
    BUFFER_SUBTYPE_BYTE         = 0,   // uint8_t[]
    BUFFER_SUBTYPE_INT8         = 1,   // int8_t[]
    BUFFER_SUBTYPE_INT16        = 2,   // int16_t[]
    BUFFER_SUBTYPE_INT32        = 3,   // int32_t[]
    BUFFER_SUBTYPE_INT64        = 4,   // int64_t[]
    BUFFER_SUBTYPE_UINT8        = 5,   // uint8_t[]
    BUFFER_SUBTYPE_UINT16       = 6,   // uint16_t[]
    BUFFER_SUBTYPE_UINT32       = 7,   // uint32_t[]
    BUFFER_SUBTYPE_UINT64       = 8,   // uint64_t[]
    BUFFER_SUBTYPE_FLOAT32      = 9,   // float[]
    BUFFER_SUBTYPE_FLOAT64      = 10,  // double[]

    // Pixel formats
    BUFFER_SUBTYPE_PIXEL_RGBA8  = 11,  // 4 bytes per pixel (R8G8B8A8)
    BUFFER_SUBTYPE_PIXEL_BGRA8  = 12,  // 4 bytes per pixel (B8G8R8A8)
    BUFFER_SUBTYPE_PIXEL_RGB8   = 13,  // 3 bytes per pixel
    BUFFER_SUBTYPE_PIXEL_BGR8   = 14,  // 3 bytes per pixel
    BUFFER_SUBTYPE_PIXEL_GRAY8  = 15,  // 1 byte per pixel
    BUFFER_SUBTYPE_PIXEL_GRAY16 = 16,  // 2 bytes per pixel
    BUFFER_SUBTYPE_PIXEL_RGBA16 = 17,  // 8 bytes per pixel
    BUFFER_SUBTYPE_PIXEL_RGBAF  = 18,  // 16 bytes per pixel (float RGBA)
    BUFFER_SUBTYPE_PIXEL_GRAYF  = 19,  // 4 bytes per pixel (float gray)

    // Audio samples
    BUFFER_SUBTYPE_SAMPLE_I8    = 20,  // 8-bit signed PCM
    BUFFER_SUBTYPE_SAMPLE_I16   = 21,  // 16-bit signed PCM
    BUFFER_SUBTYPE_SAMPLE_I24   = 22,  // 24-bit signed PCM
    BUFFER_SUBTYPE_SAMPLE_I32   = 23,  // 32-bit signed PCM
    BUFFER_SUBTYPE_SAMPLE_F32   = 24,  // 32-bit float
```

```
    BUFFER_SUBTYPE_SAMPLE_F64   = 25,  // 64-bit float

    // Complex numbers (for FFT, etc.)
    BUFFER_SUBTYPE_COMPLEX64    = 26,  // complex float (8 bytes)
    BUFFER_SUBTYPE_COMPLEX128   = 27,  // complex double (16 bytes)

    // Packed formats
    BUFFER_SUBTYPE_BOOL_PACKED  = 28,  // 1 bit per element
    BUFFER_SUBTYPE_NIBBLE       = 29,  // 4 bits per element
} buffer_subtype_t;

// ============================================================
// STREAM SUBTYPES (type = ESHKOL_VALUE_STREAM = 18)
// ============================================================
typedef enum {
    STREAM_SUBTYPE_BYTE         = 0,   // Byte stream
    STREAM_SUBTYPE_CHAR         = 1,   // Character stream (UTF-8)
    STREAM_SUBTYPE_LINE         = 2,   // Line-buffered text
    STREAM_SUBTYPE_TOKEN        = 3,   // Tokenized stream
    STREAM_SUBTYPE_AUDIO        = 4,   // Audio sample stream
    STREAM_SUBTYPE_VIDEO        = 5,   // Video frame stream
    STREAM_SUBTYPE_SENSOR       = 6,   // Sensor data stream
    STREAM_SUBTYPE_EVENT        = 7,   // Event stream
    STREAM_SUBTYPE_TAGGED       = 8,   // Tagged value stream
    STREAM_SUBTYPE_NETWORK      = 9,   // Network packet stream
} stream_subtype_t;

// ============================================================
// EVENT SUBTYPES (type = ESHKOL_VALUE_EVENT = 19)
// ============================================================
typedef enum {
    // Keyboard
    EVENT_SUBTYPE_KEY_DOWN      = 0,   // Key pressed
    EVENT_SUBTYPE_KEY_UP        = 1,   // Key released
    EVENT_SUBTYPE_KEY_REPEAT    = 2,   // Key repeat
    EVENT_SUBTYPE_TEXT_INPUT    = 3,   // Text input (Unicode)

    // Mouse
    EVENT_SUBTYPE_MOUSE_MOVE    = 4,   // Mouse motion
    EVENT_SUBTYPE_MOUSE_DOWN    = 5,   // Mouse button pressed
    EVENT_SUBTYPE_MOUSE_UP      = 6,   // Mouse button released
    EVENT_SUBTYPE_MOUSE_SCROLL  = 7,   // Mouse scroll wheel
    EVENT_SUBTYPE_MOUSE_ENTER   = 8,   // Mouse entered window
```

```
    EVENT_SUBTYPE_MOUSE_LEAVE   = 9,   // Mouse left window

    // Touch
    EVENT_SUBTYPE_TOUCH_BEGIN   = 10,  // Touch started
    EVENT_SUBTYPE_TOUCH_MOVE    = 11,  // Touch moved
    EVENT_SUBTYPE_TOUCH_END     = 12,  // Touch ended
    EVENT_SUBTYPE_TOUCH_CANCEL  = 13,  // Touch cancelled

    // Window
    EVENT_SUBTYPE_WINDOW_RESIZE = 14,  // Window resized
    EVENT_SUBTYPE_WINDOW_MOVE   = 15,  // Window moved
    EVENT_SUBTYPE_WINDOW_CLOSE  = 16,  // Window close requested
    EVENT_SUBTYPE_WINDOW_FOCUS  = 17,  // Window gained focus
    EVENT_SUBTYPE_WINDOW_BLUR   = 18,  // Window lost focus
    EVENT_SUBTYPE_WINDOW_SHOW   = 19,  // Window shown
    EVENT_SUBTYPE_WINDOW_HIDE   = 20,  // Window hidden
    EVENT_SUBTYPE_WINDOW_EXPOSE = 21,  // Window needs redraw

    // System
    EVENT_SUBTYPE_QUIT          = 22,  // Application quit requested
    EVENT_SUBTYPE_TIMER         = 23,  // Timer fired
    EVENT_SUBTYPE_IDLE          = 24,  // Idle callback
    EVENT_SUBTYPE_SIGNAL        = 25,  // OS signal received

    // Network
    EVENT_SUBTYPE_SOCKET_READY  = 26,  // Socket ready for I/O
    EVENT_SUBTYPE_SOCKET_CLOSE  = 27,  // Socket closed
    EVENT_SUBTYPE_SOCKET_ERROR  = 28,  // Socket error

    // Hardware
    EVENT_SUBTYPE_SENSOR_DATA   = 29,  // Sensor data available
    EVENT_SUBTYPE_GPIO_CHANGE   = 30,  // GPIO pin changed

    // Custom
    EVENT_SUBTYPE_USER          = 31,  // User-defined event
} event_subtype_t;
```

2.3 Object Structures

```
// ═══════════════════════════════════════════════════════════
// HANDLE OBJECT STRUCTURE
// ═══════════════════════════════════════════════════════════
typedef struct eshkol_handle {
    eshkol_object_header_t header;    // type info, LINEAR flag, etc.
    void*    platform_handle;          // Platform-specific handle
    uint32_t state;                    // HANDLE_STATE_* constants
    uint32_t refcount;                 // Reference count (if shared)
    void     (*destructor)(void*);     // Cleanup function
    void*    user_data;                // User-attached data
} eshkol_handle_t;

// Handle states
#define HANDLE_STATE_INVALID    0
#define HANDLE_STATE_OPEN       1
#define HANDLE_STATE_CLOSED     2
#define HANDLE_STATE_ERROR      3
#define HANDLE_STATE_BORROWED   4

// ═══════════════════════════════════════════════════════════
// BUFFER OBJECT STRUCTURE
// ═══════════════════════════════════════════════════════════
typedef struct eshkol_buffer {
    eshkol_object_header_t header;    // subtype = element type
    uint64_t length;                   // Number of elements
    uint64_t capacity;                 // Allocated capacity
    uint32_t element_size;             // Bytes per element
    uint32_t ndims;                    // Number of dimensions (1-4)
    uint32_t dims[4];                  // Dimension sizes
    uint32_t strides[4];               // Byte strides per dimension
    uint32_t flags;                    // BUFFER_FLAG_* constants
    uint8_t* data;                     // Pointer to element data
} eshkol_buffer_t;

// Buffer flags
#define BUFFER_FLAG_NONE        0x00
#define BUFFER_FLAG_READONLY    0x01   // Buffer is read-only
#define BUFFER_FLAG_OWNED       0x02   // Buffer owns its data
#define BUFFER_FLAG_ALIGNED     0x04   // Data is SIMD-aligned
#define BUFFER_FLAG_PINNED      0x08   // Data is pinned (no GC move)
#define BUFFER_FLAG_MAPPED      0x10   // Data is memory-mapped
```

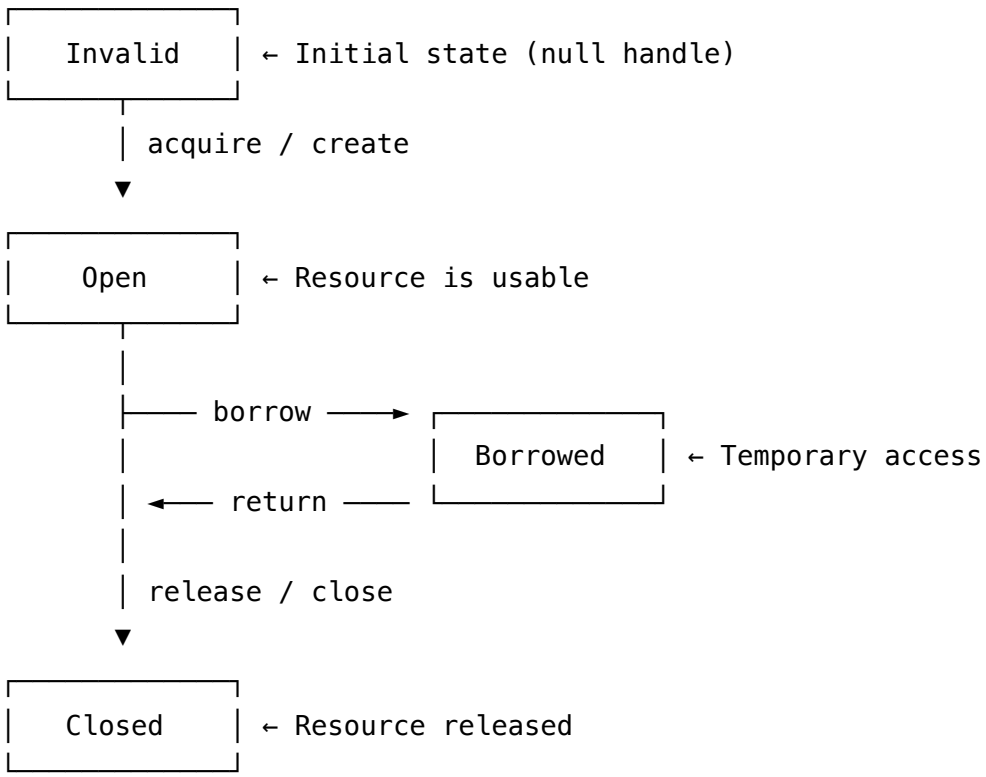
```c
#define BUFFER_FLAG_GPU         0x20   // Data is on GPU

// ============================================================
// STREAM OBJECT STRUCTURE
// ============================================================
typedef struct eshkol_stream {
    eshkol_object_header_t header;    // subtype = element type
    void*    source;                   // Underlying data source
    uint64_t position;                 // Current position
    uint64_t length;                   // Total length (-1 if unknown)
    uint32_t buffer_size;              // Internal buffer size
    uint32_t flags;                    // STREAM_FLAG_* constants

    // Function pointers for stream operations
    int64_t (*read)(struct eshkol_stream* s, void* buf, uint64_t count);
    int64_t (*write)(struct eshkol_stream* s, const void* buf, uint64_t count);
    int64_t (*seek)(struct eshkol_stream* s, int64_t offset, int whence);
    void    (*close)(struct eshkol_stream* s);
    int     (*eof)(struct eshkol_stream* s);
} eshkol_stream_t;

// Stream flags
#define STREAM_FLAG_NONE        0x00
#define STREAM_FLAG_READABLE    0x01
#define STREAM_FLAG_WRITABLE    0x02
#define STREAM_FLAG_SEEKABLE    0x04
#define STREAM_FLAG_BUFFERED    0x08
#define STREAM_FLAG_BINARY      0x10
#define STREAM_FLAG_EOF         0x20
#define STREAM_FLAG_ERROR       0x40

// ============================================================
// EVENT OBJECT STRUCTURE
// ============================================================
typedef struct eshkol_event {
    eshkol_object_header_t header;    // subtype = event type
    uint64_t timestamp;                // Event timestamp (microseconds)
    uint32_t window_id;                // Source window (if applicable)
    uint32_t device_id;                // Source device (if applicable)

    union {
        // Keyboard events
        struct {
```

```
            uint32_t keycode;          // Virtual key code
            uint32_t scancode;         // Physical scan code
            uint32_t modifiers;        // Modifier flags
            uint32_t unicode;          // Unicode codepoint (for text input)
        } key;

        // Mouse events
        struct {
            int32_t x, y;              // Position
            int32_t dx, dy;            // Delta (for motion)
            uint32_t button;           // Button number
            uint32_t buttons;          // Button state mask
            int32_t scroll_x, scroll_y; // Scroll deltas
        } mouse;

        // Touch events
        struct {
            uint32_t finger_id;        // Finger identifier
            float x, y;                // Position (normalized 0-1)
            float pressure;            // Pressure (0-1)
        } touch;

        // Window events
        struct {
            int32_t x, y;              // Position
            int32_t width, height;     // Size
        } window;

        // Timer events
        struct {
            uint32_t timer_id;         // Timer identifier
            uint64_t interval;         // Timer interval
        } timer;

        // Custom events
        struct {
            uint32_t code;             // User-defined code
            uint64_t data1, data2;     // User data
        } user;
    };
} eshkol_event_t;

// Modifier key flags
```

```c
#define EVENT_MOD_SHIFT     0x0001
#define EVENT_MOD_CTRL      0x0002
#define EVENT_MOD_ALT       0x0004
#define EVENT_MOD_META      0x0008
#define EVENT_MOD_CAPSLOCK  0x0010
#define EVENT_MOD_NUMLOCK   0x0020

// Mouse button flags
#define EVENT_BUTTON_LEFT   0x01
#define EVENT_BUTTON_MIDDLE 0x02
#define EVENT_BUTTON_RIGHT  0x04
#define EVENT_BUTTON_X1     0x08
#define EVENT_BUTTON_X2     0x10
```

3. Handle System

3.1 Handle Lifecycle



3.2 Handle Creation Functions

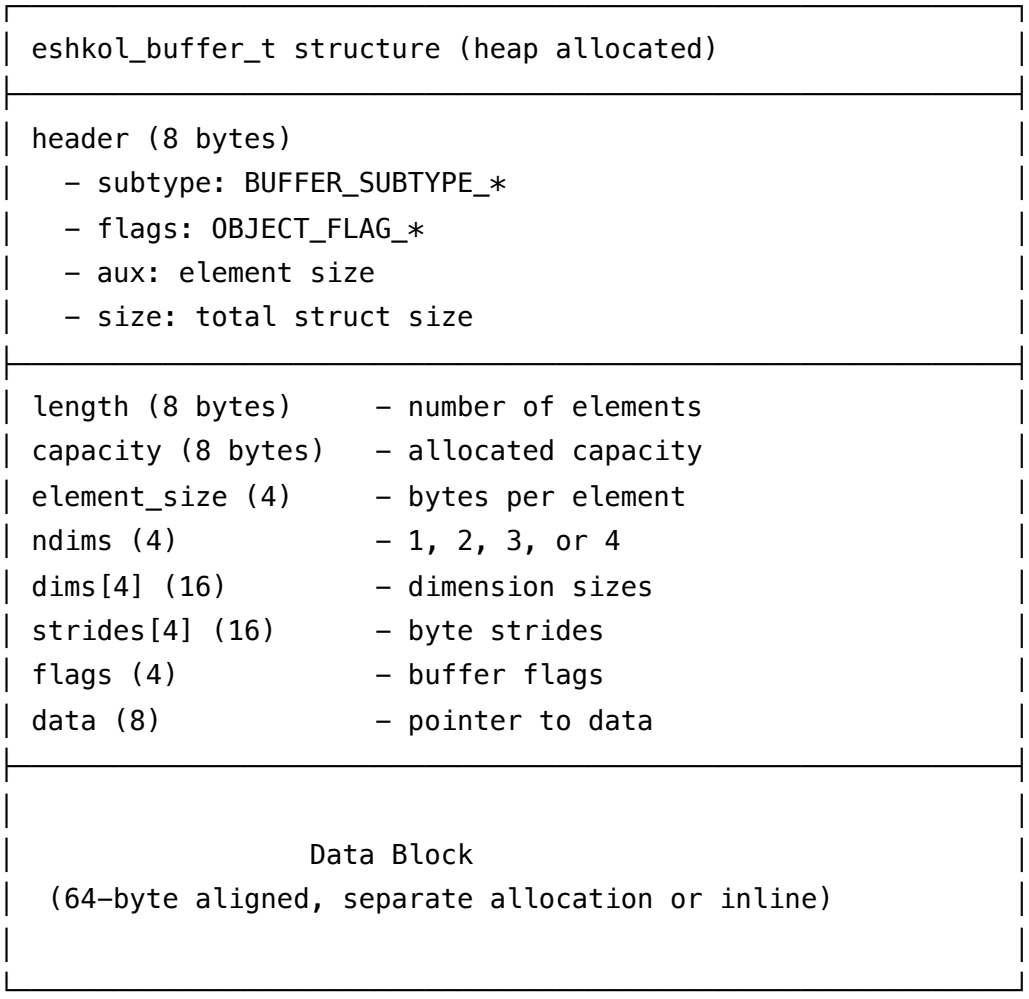
```
// In lib/core/arena_memory.cpp

eshkol_handle_t* arena_alloc_handle(handle_subtype_t subtype, void* platform_handle) {
    eshkol_handle_t* h = (eshkol_handle_t*)arena_alloc(sizeof(eshkol_handle_t));
    if (h) {
        h->header.subtype = subtype;
        h->header.flags = OBJECT_FLAG_LINEAR;  // Handles are linear by default
        h->header.aux = 0;
        h->header.size = sizeof(eshkol_handle_t);
        h->platform_handle = platform_handle;
        h->state = HANDLE_STATE_OPEN;
        h->refcount = 1;
        h->destructor = NULL;
        h->user_data = NULL;
    }
    return h;
}

// Tagged value creation
eshkol_tagged_value_t eshkol_make_handle(eshkol_handle_t* h) {
    eshkol_tagged_value_t result;
    result.type = ESHKOL_VALUE_HANDLE;
    result.flags = 0;
    result.reserved = 0;
    result.data.ptr_val = (uint64_t)h;
    return result;
}
```

3.3 Handle Operations API

```
;; === Window Handles ===
(: window-create (-> String Int Int (IO (Linear (Handle Window)))))
(: window-destroy (-> (Linear (Handle Window)) (IO Unit)))
(: window-get-size (-> (Borrow (Handle Window)) (IO (Pair Int Int))))
(: window-set-size (-> (Borrow (Handle Window)) Int Int (IO Unit)))
(: window-set-title (-> (Borrow (Handle Window)) String (IO Unit)))
(: window-get-surface (-> (Borrow (Handle Window)) (IO Surface)))
(: window-present (-> (Borrow (Handle Window)) (IO Unit)))
(: window-poll-event (-> (Borrow (Handle Window)) (IO (Maybe Event))))
(: window-wait-event (-> (Borrow (Handle Window)) (IO Event)))

;; === Audio Handles ===
(: audio-open-output (-> AudioConfig (IO (Linear (Handle AudioOutput)))))
(: audio-open-input (-> AudioConfig (IO (Linear (Handle AudioInput)))))
(: audio-close (-> (Linear (Handle AudioOutput)) (IO Unit)))
(: audio-start (-> (Borrow (Handle AudioOutput)) (IO Unit)))
(: audio-stop (-> (Borrow (Handle AudioOutput)) (IO Unit)))
(: audio-queue (-> (Borrow (Handle AudioOutput)) (Buffer SampleF32) (IO Bool)))

;; === Socket Handles ===
(: tcp-connect (-> String Int (IO (Linear (Handle SocketTCP)))))
(: tcp-listen (-> Int (IO (Linear (Handle SocketTCP)))))
(: tcp-accept (-> (Borrow (Handle SocketTCP)) (IO (Linear (Handle SocketTCP)))))
(: tcp-close (-> (Linear (Handle SocketTCP)) (IO Unit)))
(: socket-send (-> (Borrow (Handle SocketTCP)) (Buffer Byte) (IO Int)))
(: socket-recv (-> (Borrow (Handle SocketTCP)) Int (IO (Buffer Byte))))

;; === GPIO Handles ===
(: gpio-open (-> Int GPIOMode (IO (Linear (Handle GPIO)))))
(: gpio-close (-> (Linear (Handle GPIO)) (IO Unit)))
(: gpio-read (-> (Borrow (Handle GPIO)) (IO Bool)))
(: gpio-write (-> (Borrow (Handle GPIO)) Bool (IO Unit)))
```

4. Buffer System

4.1 Buffer Memory Layout



4.2 Buffer Creation Functions

```
// Allocate buffer with inline data
eshkol_buffer_t* arena_alloc_buffer(buffer_subtype_t subtype,
                                    uint64_t length,
                                    uint32_t element_size) {
    // Align data to 64 bytes
    size_t header_size = sizeof(eshkol_buffer_t);
    size_t padding = (64 - (header_size % 64)) % 64;
    size_t data_size = length * element_size;
    size_t total_size = header_size + padding + data_size;

    eshkol_buffer_t* b = (eshkol_buffer_t*)arena_alloc_aligned(total_size, 64);
    if (b) {
        b->header.subtype = subtype;
        b->header.flags = 0;
        b->header.aux = (uint16_t)element_size;
        b->header.size = (uint32_t)total_size;
        b->length = length;
        b->capacity = length;
        b->element_size = element_size;
        b->ndims = 1;
        b->dims[0] = (uint32_t)length;
        b->dims[1] = b->dims[2] = b->dims[3] = 1;
        b->strides[0] = element_size;
        b->strides[1] = b->strides[2] = b->strides[3] = 0;
        b->flags = BUFFER_FLAG_OWNED | BUFFER_FLAG_ALIGNED;
        b->data = ((uint8_t*)b) + header_size + padding;
    }
    return b;
}

// Allocate 2D buffer (for images)
eshkol_buffer_t* arena_alloc_buffer_2d(buffer_subtype_t subtype,
                                       uint32_t width,
                                       uint32_t height,
                                       uint32_t element_size) {
    uint64_t length = (uint64_t)width * height;
    eshkol_buffer_t* b = arena_alloc_buffer(subtype, length, element_size);
    if (b) {
        b->ndims = 2;
        b->dims[0] = width;
        b->dims[1] = height;
```

```
        // Row-major stride
        b->strides[0] = element_size;
        b->strides[1] = width * element_size;
    }
    return b;
}

// Create tagged value
eshkol_tagged_value_t eshkol_make_buffer(eshkol_buffer_t* b) {
    eshkol_tagged_value_t result;
    result.type = ESHKOL_VALUE_BUFFER;
    result.flags = 0;
    result.reserved = 0;
    result.data.ptr_val = (uint64_t)b;
    return result;
}
```

4.3 Buffer Operations API

```
;; === Buffer Creation ===
(: make-buffer (-> ElementType Int (Buffer e n)))
(: make-buffer-2d (-> ElementType Int Int (Buffer2D e w h)))
(: buffer-copy (-> (Buffer e n) (Buffer e n)))
(: buffer-view (-> (Buffer e n) Int Int (Buffer e m)))

;; === Buffer Access ===
(: buffer-ref (-> (Buffer e n) Int e))
(: buffer-set! (-> (Buffer e n) Int e (IO Unit)))
(: buffer-length (-> (Buffer e n) Int))
(: buffer-element-size (-> (Buffer e n) Int))

;; === 2D Buffer Access ===
(: buffer-ref-2d (-> (Buffer2D e w h) Int Int e))
(: buffer-set-2d! (-> (Buffer2D e w h) Int Int e (IO Unit)))
(: buffer-width (-> (Buffer2D e w h) Int))
(: buffer-height (-> (Buffer2D e w h) Int))

;; === Buffer Transformations ===
(: buffer-map (-> (-> a b) (Buffer a n) (Buffer b n)))
(: buffer-map! (-> (-> a a) (Buffer a n) (IO Unit)))
(: buffer-fold (-> (-> acc a acc) acc (Buffer a n) acc))
(: buffer-zip (-> (-> a b c) (Buffer a n) (Buffer b n) (Buffer c n)))
(: buffer-fill! (-> (Buffer a n) a (IO Unit)))

;; === Buffer I/O ===
(: buffer-to-file (-> (Buffer Byte n) String (IO Unit)))
(: buffer-from-file (-> String (IO (Buffer Byte))))
```

4.4 Element Size Table

Subtype	Element Size (bytes)	Description
BYTE	1	Unsigned byte
INT8	1	Signed byte
INT16	2	Signed 16-bit
INT32	4	Signed 32-bit

Subtype	Element Size (bytes)	Description
INT64	8	Signed 64-bit
FLOAT32	4	Single precision
FLOAT64	8	Double precision
PIXEL_RGBA8	4	32-bit color
PIXEL_RGB8	3	24-bit color
PIXEL_GRAY8	1	8-bit grayscale
SAMPLE_I16	2	16-bit audio
SAMPLE_F32	4	Float audio
COMPLEX64	8	Complex float
COMPLEX128	16	Complex double

5. Stream System

5.1 Stream Operations

```
// Stream creation from various sources
eshkol_stream_t* stream_from_file(const char* path, const char* mode);
eshkol_stream_t* stream_from_buffer(eshkol_buffer_t* buffer);
eshkol_stream_t* stream_from_socket(eshkol_handle_t* socket);
eshkol_stream_t* stream_from_string(const char* str);

// Stream operations
int64_t stream_read(eshkol_stream_t* s, void* buf, uint64_t count);
int64_t stream_write(eshkol_stream_t* s, const void* buf, uint64_t count);
int64_t stream_seek(eshkol_stream_t* s, int64_t offset, int whence);
void stream_close(eshkol_stream_t* s);
int stream_eof(eshkol_stream_t* s);

// High-level operations
char* stream_read_line(eshkol_stream_t* s);
eshkol_buffer_t* stream_read_all(eshkol_stream_t* s);
```

5.2 Stream API

```
;; === Stream Creation ===
(: file-stream (-> String (IO (Stream Byte))))
(: string-stream (-> String (Stream Char)))
(: buffer-stream (-> (Buffer e n) (Stream e)))
(: socket-stream (-> (Handle SocketTCP) (Stream Byte)))

;; === Stream Operations ===
(: stream-read (-> (Stream e) Int (IO (Buffer e))))
(: stream-read-line (-> (Stream Char) (IO String)))
(: stream-read-all (-> (Stream e) (IO (Buffer e))))
(: stream-write (-> (Stream e) (Buffer e) (IO Int)))
(: stream-seek (-> (Stream e) Int (IO Unit)))
(: stream-position (-> (Stream e) (IO Int)))
(: stream-eof? (-> (Stream e) (IO Bool)))
(: stream-close (-> (Stream e) (IO Unit)))

;; === Stream Combinators ===
(: stream-map (-> (-> a b) (Stream a) (Stream b)))
(: stream-filter (-> (-> a Bool) (Stream a) (Stream a)))
(: stream-take (-> Int (Stream a) (Stream a)))
(: stream-drop (-> Int (Stream a) (Stream a)))
(: stream-concat (-> (Stream a) (Stream a) (Stream a)))
```

6. Event System

6.1 Event Loop Pattern

```
;; Main event loop pattern
(define (main-loop window)
  (let loop ()
    (let ((event (window-poll-event window)))
      (match event
        ((Some (KeyDown key mods))
         (handle-key-down key mods)
         (loop))
        ((Some (MouseMove x y))
         (handle-mouse-move x y)
         (loop))
        ((Some (WindowClose))
         'quit)
        ((None)
         (render-frame window)
         (loop))))))
```

6.2 Event API

```
;; === Event Polling ===
(: poll-event (-> (IO (Maybe Event))))
(: wait-event (-> (IO Event)))
(: push-event (-> Event (IO Unit)))

;; === Event Queries ===
(: event-type (-> Event EventType))
(: event-timestamp (-> Event Int))
(: event-window (-> Event (Maybe WindowID)))

;; === Keyboard Events ===
(: key-event? (-> Event Bool))
(: key-code (-> Event Int))
(: key-modifiers (-> Event Int))
(: key-pressed? (-> Event Bool))

;; === Mouse Events ===
(: mouse-event? (-> Event Bool))
(: mouse-x (-> Event Int))
(: mouse-y (-> Event Int))
(: mouse-button (-> Event Int))

;; === Window Events ===
(: window-event? (-> Event Bool))
(: window-event-type (-> Event WindowEventType))
```

7. Linear Resource Management

7.1 Linearity Enforcement

The compiler tracks linear resource usage:

```
// In compiler type checker
class LinearityChecker {
    // Map from variable name to (type, consumed?)
    std::map<std::string, std::pair<Type, bool>> linear_vars;

public:
    void bindLinear(const std::string& name, const Type& type) {
        linear_vars[name] = {type, false};
    }

    void useLinear(const std::string& name) {
        auto it = linear_vars.find(name);
        if (it == linear_vars.end()) {
            error("Unknown linear variable: " + name);
        }
        if (it->second.second) {
            error("Linear variable already consumed: " + name);
        }
        it->second.second = true;  // Mark as consumed
    }

    void checkScopeEnd() {
        for (const auto& [name, info] : linear_vars) {
            if (!info.second) {
                error("Linear variable not consumed: " + name);
            }
        }
    }
};
```

7.2 Borrowing Pattern

```
;; Borrowing allows temporary access without consuming
(: borrow (-> (Linear a) (-> (Borrow a) b) (b * (Linear a))))

;; Example usage
(define (process-window window)
  (let-values (((result window')
                (borrow window
                  (lambda (w)
                    (window-get-size w)))))
    ;; window' is still linear, can be used
    (window-destroy window')
    result))

;; Convenience macro
(define-syntax with-borrow
  (syntax-rules ()
    ((_ handle var body ...)
     (let-values (((result handle') (borrow handle (lambda (var) body ...))))
       (values result handle')))))
```

7.3 Bracket Pattern for Automatic Cleanup

```
;; with-window ensures cleanup even on exception
(define-syntax with-window
  (syntax-rules ()
    ((_ title width height body-fn)
     (let ((window (window-create title width height)))
       (guard (exn
               (else
                (window-destroy window)
                (raise exn)))
         (let ((result (body-fn window)))
           (window-destroy window)
           result))))))

;; Usage
(with-window "My App" 800 600
  (lambda (w)
    (main-loop w)))
```

8. Platform Abstraction Layer

8.1 Directory Structure

```
lib/platform/
├── platform.h              # Common definitions, detection macros
├── platform_detect.c       # Runtime platform detection
│
├── window/
│   ├── window.h            # Window interface
│   ├── window_x11.c        # X11 implementation (Linux)
│   ├── window_wayland.c    # Wayland implementation (Linux)
│   ├── window_win32.c      # Win32 implementation (Windows)
│   ├── window_cocoa.m      # Cocoa implementation (macOS)
│   ├── window_fbdev.c      # Framebuffer implementation (embedded)
│   └── window_null.c       # Null implementation (headless)
│
├── graphics/
│   ├── graphics.h          # Graphics interface
│   ├── graphics_sw.c       # Software renderer
│   ├── graphics_gl.c       # OpenGL renderer (optional)
│   └── graphics_vk.c       # Vulkan renderer (optional)
│
├── audio/
│   ├── audio.h             # Audio interface
│   ├── audio_alsa.c        # ALSA implementation (Linux)
│   ├── audio_pulse.c       # PulseAudio implementation (Linux)
│   ├── audio_coreaudio.c   # Core Audio implementation (macOS)
│   ├── audio_wasapi.c      # WASAPI implementation (Windows)
│   └── audio_null.c        # Null implementation
│
├── network/
│   ├── network.h           # Network interface
│   ├── network_bsd.c       # BSD sockets (portable)
│   └── network_winsock.c   # Winsock (Windows)
│
├── embedded/
│   ├── gpio.h              # GPIO interface
│   ├── gpio_linux.c        # Linux sysfs GPIO
│   ├── gpio_rpi.c          # Raspberry Pi direct GPIO
│   ├── i2c.h               # I2C interface
│   ├── i2c_linux.c         # Linux I2C
```

```
│   ├── spi.h              # SPI interface
│   ├── spi_linux.c        # Linux SPI
│   └── uart.h             # UART interface
│
└── event/
    ├── event.h            # Event interface
    ├── event_queue.c      # Event queue implementation
    └── event_poll.c       # Platform event polling
```

8.2 Window Interface (platform/window/window.h)

```c
#ifndef ESHKOL_PLATFORM_WINDOW_H
#define ESHKOL_PLATFORM_WINDOW_H

#include <stdint.h>
#include <stdbool.h>

// Opaque window handle
typedef struct eshkol_window eshkol_window_t;

// Surface for drawing
typedef struct eshkol_surface {
    uint32_t width;
    uint32_t height;
    uint32_t pitch;           // Bytes per row
    uint32_t format;          // Pixel format
    uint8_t* pixels;          // Pixel data
} eshkol_surface_t;

// Window creation flags
#define WINDOW_FLAG_RESIZABLE   0x01
#define WINDOW_FLAG_BORDERLESS  0x02
#define WINDOW_FLAG_FULLSCREEN  0x04
#define WINDOW_FLAG_HIDDEN      0x08
#define WINDOW_FLAG_ALWAYS_TOP  0x10

// ============================================================
// WINDOW LIFECYCLE
// ============================================================
eshkol_window_t* eshkol_window_create(const char* title,
                                      int width, int height,
                                      uint32_t flags);
void eshkol_window_destroy(eshkol_window_t* window);

// ============================================================
// WINDOW PROPERTIES
// ============================================================
void eshkol_window_get_size(eshkol_window_t* window, int* w, int* h);
void eshkol_window_set_size(eshkol_window_t* window, int w, int h);
void eshkol_window_get_position(eshkol_window_t* window, int* x, int* y);
void eshkol_window_set_position(eshkol_window_t* window, int x, int y);
void eshkol_window_set_title(eshkol_window_t* window, const char* title);
```

```
bool eshkol_window_should_close(eshkol_window_t* window);
void eshkol_window_request_close(eshkol_window_t* window);

// ============================================================
// WINDOW SURFACE
// ============================================================
eshkol_surface_t* eshkol_window_get_surface(eshkol_window_t* window);
void eshkol_window_present(eshkol_window_t* window);

// ============================================================
// WINDOW EVENTS
// ============================================================
struct eshkol_event;
bool eshkol_window_poll_event(eshkol_window_t* window,
                              struct eshkol_event* event);
void eshkol_window_wait_event(eshkol_window_t* window,
                              struct eshkol_event* event);

// ============================================================
// PLATFORM INITIALIZATION
// ============================================================
int eshkol_platform_init(void);
void eshkol_platform_quit(void);

#endif // ESHKOL_PLATFORM_WINDOW_H
```

8.3 Audio Interface (platform/audio/audio.h)

```
#ifndef ESHKOL_PLATFORM_AUDIO_H
#define ESHKOL_PLATFORM_AUDIO_H

#include <stdint.h>
#include <stdbool.h>

// Audio device handle
typedef struct eshkol_audio_device eshkol_audio_device_t;

// Audio format
typedef enum {
    AUDIO_FORMAT_U8,      // Unsigned 8-bit
    AUDIO_FORMAT_S16,     // Signed 16-bit
    AUDIO_FORMAT_S24,     // Signed 24-bit
    AUDIO_FORMAT_S32,     // Signed 32-bit
    AUDIO_FORMAT_F32,     // Float 32-bit
} eshkol_audio_format_t;

// Audio configuration
typedef struct {
    uint32_t sample_rate;     // e.g., 44100, 48000
    uint8_t channels;         // 1 = mono, 2 = stereo
    eshkol_audio_format_t format;
    uint32_t buffer_frames;   // Frames per buffer
} eshkol_audio_config_t;

// Audio callback type
typedef void (*eshkol_audio_callback_t)(void* userdata,
                                        uint8_t* buffer,
                                        uint32_t frames);

// ═══════════════════════════════════════════════════════════
// AUDIO DEVICE MANAGEMENT
// ═══════════════════════════════════════════════════════════
eshkol_audio_device_t* eshkol_audio_open_output(
    const eshkol_audio_config_t* config,
    eshkol_audio_callback_t callback,
    void* userdata);

eshkol_audio_device_t* eshkol_audio_open_input(
    const eshkol_audio_config_t* config,
```

```
    eshkol_audio_callback_t callback,
    void* userdata);

void eshkol_audio_close(eshkol_audio_device_t* device);

// ============================================================
// AUDIO PLAYBACK CONTROL
// ============================================================
void eshkol_audio_start(eshkol_audio_device_t* device);
void eshkol_audio_stop(eshkol_audio_device_t* device);
void eshkol_audio_pause(eshkol_audio_device_t* device);

// ============================================================
// QUEUE-BASED AUDIO (alternative to callback)
// ============================================================
bool eshkol_audio_queue(eshkol_audio_device_t* device,
                        const void* data,
                        uint32_t frames);
uint32_t eshkol_audio_queued_frames(eshkol_audio_device_t* device);

#endif // ESHKOL_PLATFORM_AUDIO_H
```

9. Compiler Extensions

9.1 Type Checking for Multimedia Types

```
// In lib/types/type_checker.cpp

class MultimediaTypeChecker {
public:
    // Check handle type
    bool checkHandleType(const Type& type, handle_subtype_t expected) {
        if (!type.isHandle()) return false;
        return type.getHandleSubtype() == expected;
    }

    // Check buffer element type
    bool checkBufferElementType(const Type& type, buffer_subtype_t expected) {
        if (!type.isBuffer()) return false;
        return type.getBufferSubtype() == expected;
    }

    // Check buffer dimensions
    bool checkBufferDimensions(const Type& type,
                               const std::vector<uint64_t>& dims) {
        if (!type.isBuffer()) return false;
        if (type.getDimensions().size() != dims.size()) return false;

        for (size_t i = 0; i < dims.size(); i++) {
            if (dims[i] != 0 && type.getDimension(i) != dims[i]) {
                return false;  // Dimension mismatch
            }
        }
        return true;
    }

    // Infer result type of buffer operations
    Type inferBufferMapResult(const Type& input, const Type& fn) {
        // buffer-map : (-> a b) -> Buffer a n -> Buffer b n
        if (!input.isBuffer()) {
            error("Expected buffer type");
        }
        Type elemType = fn.getReturnType();
        buffer_subtype_t subtype = elementTypeToSubtype(elemType);
```

```
        return Type::Buffer(subtype, input.getDimensions());
    }
};
```

9.2 LLVM Codegen for Multimedia

```
// In lib/backend/multimedia_codegen.cpp

class MultimediaCodegen {
    LLVMContext& ctx;
    IRBuilder<>& builder;
    Module& module;

public:
    // Generate buffer creation
    Value* codegenMakeBuffer(buffer_subtype_t subtype, uint64_t length) {
        // Call arena_alloc_buffer
        Function* alloc_fn = module.getFunction("arena_alloc_buffer");
        Value* subtype_val = builder.getInt8(subtype);
        Value* length_val = builder.getInt64(length);
        Value* elem_size = builder.getInt32(getElementSize(subtype));

        Value* buffer_ptr = builder.CreateCall(
            alloc_fn, {subtype_val, length_val, elem_size});

        // Wrap in tagged value
        return packPtrToTaggedValue(buffer_ptr, ESHKOL_VALUE_BUFFER);
    }

    // Generate buffer access
    Value* codegenBufferRef(Value* buffer, Value* index) {
        // Extract buffer pointer
        Value* buf_ptr = extractPtrFromTaggedValue(buffer);

        // Get data pointer from buffer struct
        Value* data_ptr = builder.CreateStructGEP(
            buffer_type, buf_ptr, BUFFER_FIELD_DATA);
        Value* data = builder.CreateLoad(ptr_type, data_ptr);

        // Get element size
        Value* elem_size_ptr = builder.CreateStructGEP(
            buffer_type, buf_ptr, BUFFER_FIELD_ELEMENT_SIZE);
        Value* elem_size = builder.CreateLoad(i32_type, elem_size_ptr);

        // Calculate offset
        Value* offset = builder.CreateMul(index,
            builder.CreateZExt(elem_size, i64_type));
```

```
        Value* elem_ptr = builder.CreateGEP(i8_type, data, offset);

        return elem_ptr;
    }

    // Generate handle creation
    Value* codegenWindowCreate(Value* title, Value* width, Value* height) {
        Function* create_fn = module.getFunction("eshkol_window_create");
        Value* flags = builder.getInt32(0);

        Value* window_ptr = builder.CreateCall(
            create_fn, {title, width, height, flags});

        // Wrap platform handle in eshkol_handle_t
        Function* wrap_fn = module.getFunction("arena_alloc_handle");
        Value* subtype = builder.getInt8(HANDLE_SUBTYPE_WINDOW);
        Value* handle = builder.CreateCall(wrap_fn, {subtype, window_ptr});

        // Create tagged value with LINEAR flag
        Value* tagged = packPtrToTaggedValue(handle, ESHKOL_VALUE_HANDLE);

        // Mark as linear in the type system
        markAsLinear(tagged);

        return tagged;
    }
};
```

10. Standard Library

10.1 Graphics Library (lib/media/graphics.esk)

```
;;; lib/media/graphics.esk
;;; High-level graphics operations

;; Color type (32-bit RGBA)
(define (rgba r g b a)
  (bitwise-ior
    (bitwise-and r #xFF)
    (arithmetic-shift (bitwise-and g #xFF) 8)
    (arithmetic-shift (bitwise-and b #xFF) 16)
    (arithmetic-shift (bitwise-and a #xFF) 24)))

(define (rgb r g b) (rgba r g b 255))

;; Color extraction
(define (color-r c) (bitwise-and c #xFF))
(define (color-g c) (bitwise-and (arithmetic-shift c -8) #xFF))
(define (color-b c) (bitwise-and (arithmetic-shift c -16) #xFF))
(define (color-a c) (bitwise-and (arithmetic-shift c -24) #xFF))

;; Predefined colors
(define black   (rgb 0 0 0))
(define white   (rgb 255 255 255))
(define red     (rgb 255 0 0))
(define green   (rgb 0 255 0))
(define blue    (rgb 0 0 255))
(define yellow  (rgb 255 255 0))
(define cyan    (rgb 0 255 255))
(define magenta (rgb 255 0 255))

;; Drawing primitives (pure Eshkol implementations)
(define (draw-line surface x0 y0 x1 y1 color)
  ;; Bresenham's line algorithm
  (let* ((dx (abs (- x1 x0)))
         (dy (- (abs (- y1 y0))))
         (sx (if (< x0 x1) 1 -1))
         (sy (if (< y0 y1) 1 -1))
         (err (+ dx dy)))
    (let loop ((x x0) (y y0) (e err))
```

```
      (surface-set-pixel! surface x y color)
      (unless (and (= x x1) (= y y1))
        (let ((e2 (* 2 e)))
          (let ((new-x (if (>= e2 dy) (+ x sx) x))
                (new-y (if (<= e2 dx) (+ y sy) y))
                (new-e (+ e
                          (if (>= e2 dy) dy 0)
                          (if (<= e2 dx) dx 0))))
            (loop new-x new-y new-e)))))))

(define (draw-rect surface x y w h color)
  (draw-line surface x y (+ x w) y color)
  (draw-line surface (+ x w) y (+ x w) (+ y h) color)
  (draw-line surface (+ x w) (+ y h) x (+ y h) color)
  (draw-line surface x (+ y h) x y color))

(define (fill-rect surface x y w h color)
  (do ((row y (+ row 1)))
      ((>= row (+ y h)))
    (do ((col x (+ col 1)))
        ((>= col (+ x w)))
      (surface-set-pixel! surface col row color))))

(define (draw-circle surface cx cy r color)
  ;; Midpoint circle algorithm
  (let loop ((x r) (y 0) (err (- 1 r)))
    (when (<= y x)
      (surface-set-pixel! surface (+ cx x) (+ cy y) color)
      (surface-set-pixel! surface (+ cx y) (+ cy x) color)
      (surface-set-pixel! surface (- cx y) (+ cy x) color)
      (surface-set-pixel! surface (- cx x) (+ cy y) color)
      (surface-set-pixel! surface (- cx x) (- cy y) color)
      (surface-set-pixel! surface (- cx y) (- cy x) color)
      (surface-set-pixel! surface (+ cx y) (- cy x) color)
      (surface-set-pixel! surface (+ cx x) (- cy y) color)
      (if (< err 0)
          (loop x (+ y 1) (+ err (* 2 y) 3))
          (loop (- x 1) (+ y 1) (+ err (* 2 (- y x)) 5))))))
```

10.2 Audio Library (lib/media/audio.esk)

```
;;; lib/media/audio.esk
;;; Audio synthesis and processing

;; Standard audio configuration
(define (audio-config-default)
  (make-audio-config 44100 2 'float32 1024))

;; Basic oscillators
(define (sine-osc freq sample-rate)
  (let ((phase 0.0)
        (delta (/ (* 2.0 pi freq) sample-rate)))
    (lambda ()
      (let ((sample (sin phase)))
        (set! phase (+ phase delta))
        (when (>= phase (* 2.0 pi))
          (set! phase (- phase (* 2.0 pi))))
        sample))))

(define (square-osc freq sample-rate)
  (let ((sine (sine-osc freq sample-rate)))
    (lambda ()
      (if (>= (sine) 0.0) 1.0 -1.0))))

(define (sawtooth-osc freq sample-rate)
  (let ((phase 0.0)
        (delta (/ freq sample-rate)))
    (lambda ()
      (let ((sample (- (* 2.0 phase) 1.0)))
        (set! phase (+ phase delta))
        (when (>= phase 1.0)
          (set! phase (- phase 1.0)))
        sample))))

(define (triangle-osc freq sample-rate)
  (let ((phase 0.0)
        (delta (/ freq sample-rate)))
    (lambda ()
      (let ((sample (- (* 4.0 (abs (- phase 0.5))) 1.0)))
        (set! phase (+ phase delta))
        (when (>= phase 1.0)
          (set! phase (- phase 1.0)))
```

```
        sample))))

;; ADSR envelope
(define (make-adsr attack decay sustain release sample-rate)
  (let ((a-samples (* attack sample-rate))
        (d-samples (* decay sample-rate))
        (r-samples (* release sample-rate)))
    (lambda (time gate)
      (cond
        ((< time a-samples)
         ;; Attack phase
         (/ time a-samples))
        ((< time (+ a-samples d-samples))
         ;; Decay phase
         (let ((t (- time a-samples)))
           (- 1.0 (* (- 1.0 sustain) (/ t d-samples)))))
        (gate
         ;; Sustain phase
         sustain)
        (else
         ;; Release phase
         (* sustain (- 1.0 (/ (- time (+ a-samples d-samples)) r-samples))))))))

;; Simple synthesizer
(define (synth-note osc envelope duration sample-rate)
  (let ((num-samples (floor (* duration sample-rate))))
    (let ((buffer (make-buffer 'sample-f32 num-samples)))
      (do ((i 0 (+ i 1)))
          ((>= i num-samples) buffer)
        (let ((sample (* (osc) (envelope i #t))))
          (buffer-set! buffer i sample))))))
```

11. Implementation Phases

Phase 1: Core Infrastructure (Week 1)

Prerequisites: Pointer consolidation complete

Tasks:

1. Add multimedia type constants to eshkol.h
2. Add subtype enumerations (handle, buffer, stream, event)
3. Add object structures with headers
4. Add type checking macros

Files:

- `inc/eshkol/eshkol.h`

Validation:

- Build succeeds
- Existing tests pass

Phase 2: Buffer System (Week 2)

Tasks:

1. Implement buffer allocation functions
2. Implement buffer codegen (create, ref, set)
3. Implement buffer operations (map, fold, zip)
4. Add buffer tests

Files:

- `lib/core/arena_memory.cpp`
- `lib/backend/buffer_codegen.cpp` (new)
- `tests/features/buffer_test.esk` (new)

Validation:

- Buffer creation works
- Buffer access works
- Buffer transformations work

Phase 3: Handle System (Week 3)

Tasks:

1. Implement handle allocation functions
2. Implement linear type tracking in compiler
3. Add platform abstraction headers
4. Implement null platform (for testing)

Files:

- `lib/core/arena_memory.cpp`
- `lib/types/linearity_checker.cpp` (new)
- `lib/platform/window/window.h`
- `lib/platform/window/window_null.c`

Validation:

- Handle creation works
- Linear type errors caught
- Handle cleanup works

Phase 4: Window System (Week 4-5)

Tasks:

1. Implement X11 window backend
2. Implement surface operations
3. Implement event polling
4. Add window tests

Files:

- `lib/platform/window/window_x11.c`
- `lib/platform/event/event.c`
- `lib/media/window.esk`
- `tests/features/window_test.esk` (new)

Validation:

- Window opens
- Drawing works
- Events received

Phase 5: Audio System (Week 6)

Tasks:

1. Implement ALSA audio backend
2. Implement audio queue/callback
3. Add audio synthesis library
4. Add audio tests

Files:

- `lib/platform/audio/audio_alsa.c`
- `lib/media/audio.esk`
- `tests/features/audio_test.esk` (new)

Validation:

- Audio output works
- Audio input works
- Synthesis works

Phase 6: Network System (Week 7)

Tasks:

1. Implement BSD socket wrapper
2. Add TCP/UDP support
3. Add network tests

Files:

- `lib/platform/network/network_bsd.c`
- `lib/media/network.esk`
- `tests/features/network_test.esk` (new)

Validation:

- TCP connect/accept works
- UDP send/recv works

- Streaming works

Phase 7: Embedded I/O (Week 8)

Tasks:

1. Implement Linux GPIO
2. Implement I2C/SPI
3. Add embedded tests

Files:

- `lib/platform/embedded/gpio_linux.c`
- `lib/platform/embedded/i2c_linux.c`
- `lib/platform/embedded/spi_linux.c`

Validation:

- GPIO read/write works
- I2C communication works
- SPI communication works

Phase 8: Integration and Polish (Week 9-10)

Tasks:

1. Add macOS and Windows backends
2. Optimize buffer operations
3. Add comprehensive documentation
4. Performance testing

Validation:

- Cross-platform works
- Performance acceptable
- Documentation complete

12. Testing Strategy

12.1 Unit Tests

```
;;; tests/features/buffer_test.esk

;; Test buffer creation
(define buf1 (make-buffer 'byte 100))
(assert (= (buffer-length buf1) 100))

;; Test buffer access
(buffer-set! buf1 50 42)
(assert (= (buffer-ref buf1 50) 42))

;; Test buffer map
(define buf2 (make-buffer 'int32 10))
(buffer-fill! buf2 5)
(define buf3 (buffer-map (lambda (x) (* x 2)) buf2))
(assert (= (buffer-ref buf3 0) 10))

(display "Buffer tests passed!")
(newline)
```

12.2 Integration Tests

```
;;; tests/integration/graphics_test.esk

(with-window "Test Window" 640 480
  (lambda (window)
    ;; Test surface access
    (let ((surface (window-get-surface window)))
      (surface-clear surface black)
      (fill-rect surface 100 100 200 150 red)
      (draw-circle surface 320 240 50 green)
      (window-present window))

    ;; Test events
    (let ((event (window-wait-event window)))
      (assert (event? event)))

    (display "Graphics test passed!")
    (newline)))
```

12.3 Performance Tests

```
;;; tests/performance/buffer_perf.esk

(define (benchmark name thunk iterations)
  (let ((start (current-milliseconds)))
    (do ((i 0 (+ i 1)))
        ((>= i iterations))
      (thunk))
    (let ((elapsed (- (current-milliseconds) start)))
      (display name)
      (display ": ")
      (display (/ elapsed iterations))
      (display " ms/iteration")
      (newline))))

;; Buffer allocation benchmark
(benchmark "buffer-alloc-1M"
  (lambda () (make-buffer 'float32 1000000))
  1000)

;; Buffer map benchmark
(define test-buf (make-buffer 'float32 1000000))
(benchmark "buffer-map-1M"
  (lambda () (buffer-map (lambda (x) (* x 2.0)) test-buf))
  100)
```

Appendix A: Quick Reference

A.1 Type Constants

Type	Value	Description
ESHKOL_VALUE_HANDLE	16	Resource handles
ESHKOL_VALUE_BUFFER	17	Data buffers
ESHKOL_VALUE_STREAM	18	Data streams
ESHKOL_VALUE_EVENT	19	Input events

A.2 Common Handle Subtypes

Subtype	Value	Description
HANDLE_SUBTYPE_WINDOW	0	GUI window
HANDLE_SUBTYPE_AUDIO_OUT	3	Audio output
HANDLE_SUBTYPE_SOCKET_TCP	7	TCP socket
HANDLE_SUBTYPE_GPIO	13	GPIO pin

A.3 Common Buffer Subtypes

Subtype	Value	Element Size
BUFFER_SUBTYPE_BYTE	0	1
BUFFER_SUBTYPE_FLOAT32	9	4
BUFFER_SUBTYPE_PIXEL_RGBA8	11	4
BUFFER_SUBTYPE_SAMPLE_F32	24	4

End of Multimedia System Architecture Document