Kyty - PS4 & PS5 emulator

Contents

Using Worms W.M.D as example

1. Loadind the program

Tldr: All elf files and libaries are loaded, their exported/imported symbols are mapped and relocated. run_entry() and game_main_loop() run concurrently

Rough overview about the loading and symbol linking of the program/game. Ends with the jump into the program.

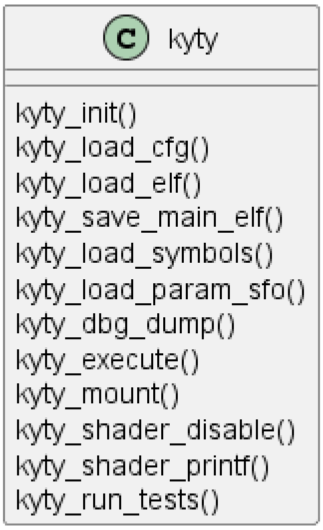


Figure 1. Kyty Lua functions (for the init process). Registerd by kyty_reg() in Kyty.cpp

1.1. kyty_load_elf_func()

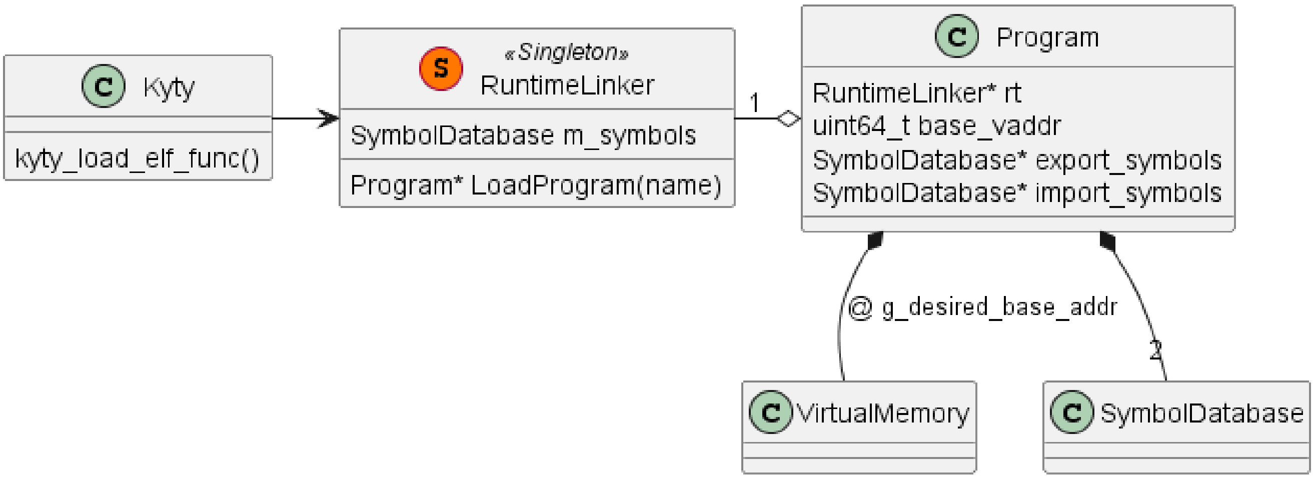


Figure 2. Class: RuntimeLinker

The selected files (here) eboot.bin, libc.prx and libSceFios2.prx are loaded to g_desired_base_addr by calling *RuntimeLinker::LoadProgram()*.

Hint: Loading/Parsing the program initially is done by "Emulator/Loader/elf64"

*"g_desired_base_addr += CODE_BASE_INCR * (1 + alloc_size / CODE_BASE_INCR)"*

The segments: code, data (read only) and data (read &write) are read and loaded to the programs virtualMemory afterwards.

todo: tls header patch?

Table 1. LoadProgram memory layout

file	address
eboot.bin	0x0000000900000000 **code**
	0x0000000900f60000 **data ro**
	0x0000000901010000 **data rw**
libc.prx	0x0000000920000000 **code**
	0x00000009200b8000 **data ro**
	0x00000009200c0000 **data rw**
libSceFios2.prx	0x0000000930000000 **code**
	0x0000000930058000 **data ro**
	0x000000093005c000 **data rw**

After loading a file to memory, its symbol database is created for the exported/imported functions by calling *CreateSymbolDatabase(Program)*.

Next step is to load libraries which provide the necessary symbols.

1.2. kyty_load_symbols()

Available libs are hardcoded currently. See emulator/src/Libs/Libs.cpp. Every library provides an *Init*-function* (e.g. InitVideoOut_1) and is directly called by *kyty_load_symbols()*. It adds its functions to the SymbolDatabase RuntimeLinker::m_symbols

⇒ Every program has its two own SymbolDatabase for export/import and RuntimeLinker holds all symbols from the provided libraries.

1.3. kyty_execute()

Runs RuntimeLinker::execute() on a newly created thread ("MainThread").

The process-thread calls WindowRun(), which initializes the window and vulkan first and then jumps to the game_main_loop(), afterwards.

Both are now running concurrently.

1.3.1. RuntimeLinker::execute()

Initializes the thread by calling PthreadAttrInit():

```
[9][00:00:00.114] libkernel::libkernel::PthreadAttrInit()
[9][00:00:00.114] libkernel::libkernel::PthreadAttrSetinheritsched()
[9][00:00:00.114] libkernel::libkernel::PthreadAttrSetschedparam()
[9][00:00:00.114] libkernel::libkernel::PthreadAttrSetschedpolicy()
[9][00:00:00.114] libkernel::libkernel::PthreadAttrSetdetachstate()
[9][00:00:00.114] libkernel::libkernel::PthreadAttrGetaffinity()
[9][00:00:00.114] libkernel::libkernel::PthreadAttrGetdetachstate()
[9][00:00:00.115] libkernel::libkernel::PthreadAttrGetguardsize()
[9][00:00:00.115] libkernel::libkernel::PthreadAttrGetinheritsched()
[9][00:00:00.115] libkernel::libkernel::PthreadAttrGetschedparam()
[9][00:00:00.115] libkernel::libkernel::PthreadAttrGetschedpolicy()
[9][00:00:00.115] libkernel::libkernel::PthreadAttrGetstackaddr()
[9][00:00:00.115] libkernel::libkernel::PthreadAttrGetstacksize()
    cpu_mask       = 0x7f
    detach_state   = 0
    guard_size     = 4096
    inherit_sched  = 4
    sched_priority = 700
    policy         = 1
    stack_addr     = 0x0000000000000000
    stack_size    = 0
```

Next, calls RelocateAll():

```
--- Relocate program: */eboot.bin ---
--- Relocate program: */sce_module/libc.prx ---
--- Relocate program: */sce_module/libSceFios2.prx ---
```

In short, it sets the symbols from the symbolDatabases.

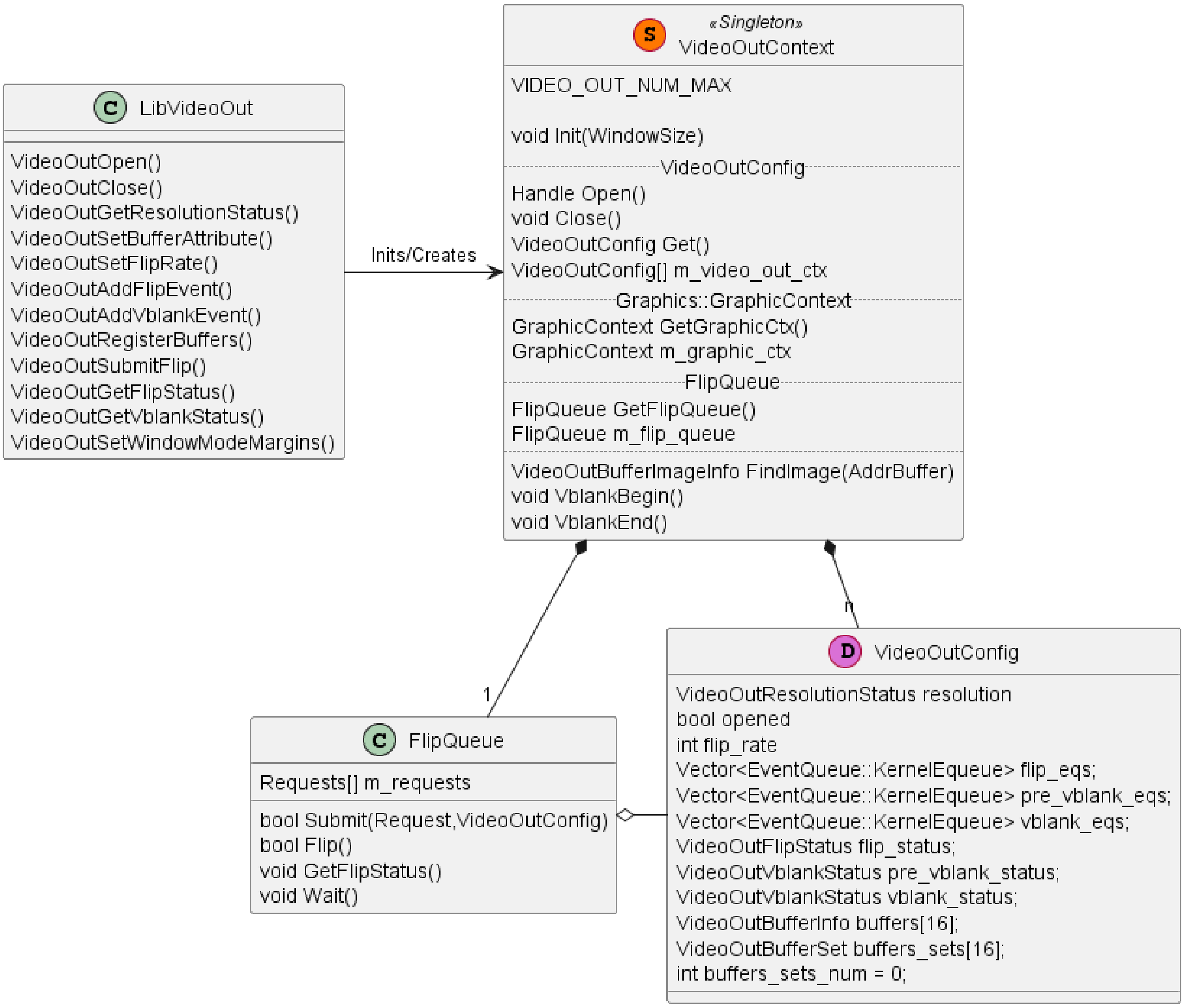
Before running the program, the modules have to be initialized- StartAllModules(). It calls the modules init-function.

The main-program is now ready to be called- run_entry()

2. LibVideoOut

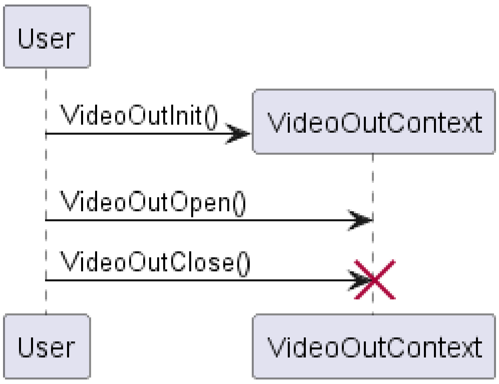
Init: Sets the resolution

Roughly describing the requirements for LibVideoOut API



C LibVideoOut
VideoOutOpen()
VideoOutClose()
VideoOutGetResolutionStatus()
VideoOutSetBufferAttribute()
VideoOutSetFlipRate()
VideoOutAddFlipEvent()
VideoOutAddVblankEvent()
VideoOutRegisterBuffers()
VideoOutSubmitFlip()
VideoOutGetFlipStatus()
VideoOutGetVblankStatus()
VideoOutSetWindowModeMargins()
Inits/Creates
«Singleton»
S VideoOutContext
VIDEO_OUT_NUM_MAX
void Init(WindowSize)
VideoOutConfig
Handle Open()
void Close()
VideoOutConfig Get()
VideoOutConfig[] m_video_out_ctx
Graphics::GraphicContext
GraphicContext GetGraphicCtx()
GraphicContext m_graphic_ctx
FlipQueue
FlipQueue GetFlipQueue()
FlipQueue m_flip_queue
VideoOutBufferImageInfo FindImage(AddrBuffer)
void VblankBegin()
void VblankEnd()
1
n
C FlipQueue
Requests[] m_requests
bool Submit(Request,VideoOutConfig)
bool Flip()
void GetFlipStatus()
void Wait()
D VideoOutConfig
VideoOutResolutionStatus resolution
bool opened
int flip_rate
Vector<EventQueue::KernelEqueue> flip_eqs;
Vector<EventQueue::KernelEqueue> pre_vblank_eqs;
Vector<EventQueue::KernelEqueue> vblank_eqs;
VideoOutFlipStatus flip_status;
VideoOutVblankStatus pre_vblank_status;
VideoOutVblankStatus vblank_status;
VideoOutBufferInfo buffers[16];
VideoOutBufferSet buffers_sets[16];
int buffers_sets_num = 0;

Figure 3. LibVideoOut exported Functions



```
[9][00:00:01.085] VideoOut::VideoOut::VideoOutOpen()
[9][00:00:01.085] VideoOut::VideoOut::VideoOutGetResolutionStatus()
[9][00:00:01.155] VideoOut::VideoOut::VideoOutAddFlipEvent()
[9][00:00:01.155] VideoOut::VideoOut::VideoOutSetBufferAttribute()
[9][00:00:01.155] VideoOut::VideoOut::VideoOutRegisterBuffers()
...
[22][00:00:05.122] VideoOut::VideoOut::VideoOutSubmitFlip()
...
[22][00:00:05.208] VideoOut::VideoOut::VideoOutSubmitFlip()
... // Repeats
```

2.1. VideoOutOpen()

```
Checks if a unused VideoOutConfig is available (1..VIDEO_OUT_NUM_MAX) and initializes
it. Array Index "starts" with 1. Index is returned.
```

2.2. VideoOutGetResolutionStatus(): RetByRef VideoOutResolutionStatus of the handle

2.3. VideoOutAddFlipEvent(udata)

```
Creates KernelEqueueEvent and add it to the KernelEventQueue and
VideoOutConfig::flip_eqs
```

```
Ident: VIDEO_OUT_EVENT_FLIP_Queue
Queue: KERNEL_EVFILT_VIDEO_OUT
   event.filter.data = videoOutConfig
   event.event.udata = udata

callbacks:
   flip_event_delete_func
   flip_event_reset_func
```

```
    flip_event_trigger_func
```

2.4. VideoOutSetBufferAttribute()

```
 Only wants the VideoOutBufferAttribute. Which is filled only with the information
provided as values.
-> No Internal changes
```

2.5. VideoOutRegisterBuffers()

```
start_index = 0
buffer_num = 2
pixel_format   = 0x80000000
tiling_mode    = 0
aspect_ratio   = 0
width          = 1920
height         = 1080
pitch_in_pixel = 1920
option = 0
```

Calls Graphics::WindowWaitForGraphicInitialized() and Graphics::GraphicsRenderCreateContext().

```
for start_index < buffer_num
    Graphics::GpuMemoryCreateObject()
```

→ Creates the VideoOutBuffers at the provided **addresses (Sysstem Memory)** and stores them in VideoOutConfig::buffers.

buffer := System Memory Address, buffer_vulkan := GpuMemoryCreateObject

2.6. VideoOutSubmitFlip()

g_video_out_context→GetFlipQueue().Submit(ctx, index, flip_arg)

→ Draws the vulkan_buffer to videoOut and increments the frame counter.

3. LibGraphics

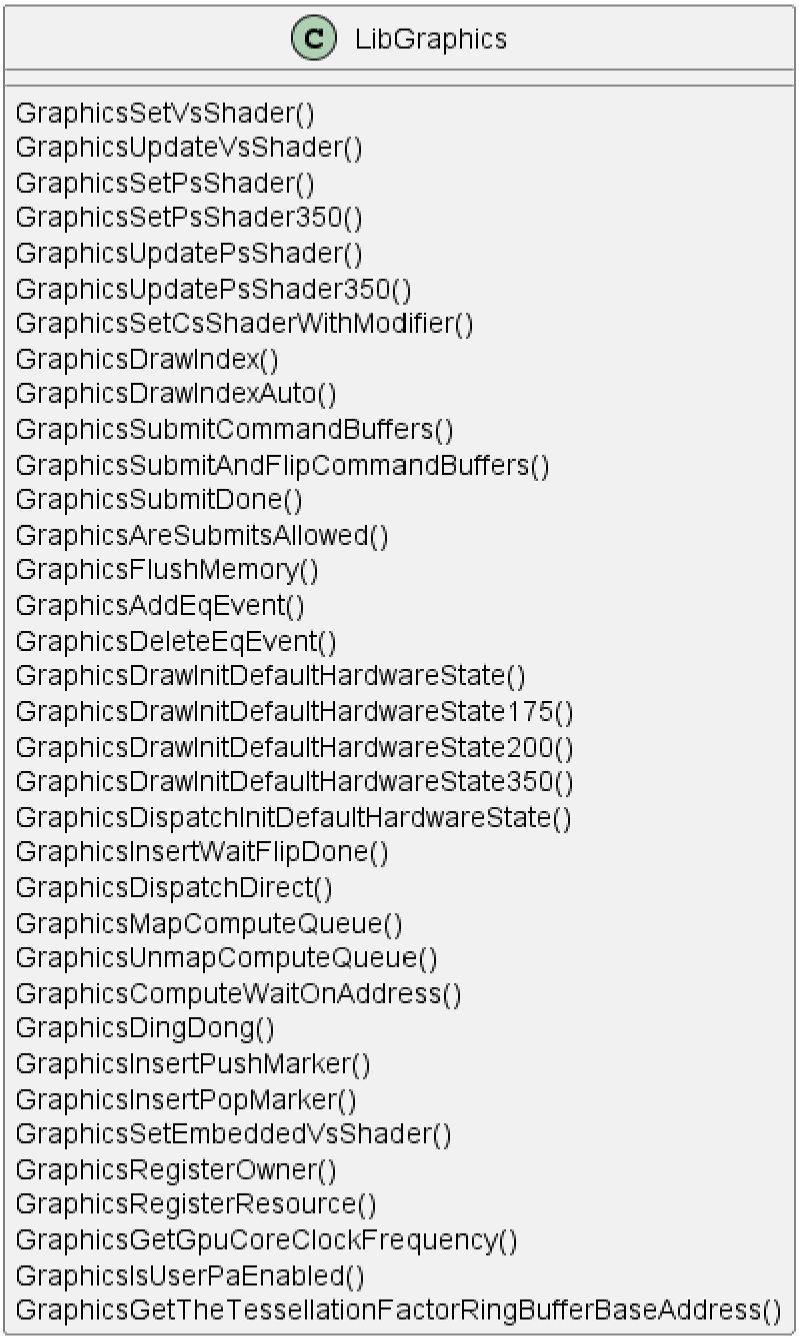

C
LibGraphics
GraphicsSetVsShader()
GraphicsUpdateVsShader()
GraphicsSetPsShader()
GraphicsSetPsShader350()
GraphicsUpdatePsShader()
GraphicsUpdatePsShader350()
GraphicsSetCsShaderWithModifier()
GraphicsDrawIndex()
GraphicsDrawIndexAuto()
GraphicsSubmitCommandBuffers()
GraphicsSubmitAndFlipCommandBuffers()
GraphicsSubmitDone()
GraphicsAreSubmitsAllowed()
GraphicsFlushMemory()
GraphicsAddEqEvent()
GraphicsDeleteEqEvent()
GraphicsDrawInitDefaultHardwareState()
GraphicsDrawInitDefaultHardwareState175()
GraphicsDrawInitDefaultHardwareState200()
GraphicsDrawInitDefaultHardwareState350()
GraphicsDispatchInitDefaultHardwareState()
GraphicsInsertWaitFlipDone()
GraphicsDispatchDirect()
GraphicsMapComputeQueue()
GraphicsUnmapComputeQueue()
GraphicsComputeWaitOnAddress()
GraphicsDingDong()
GraphicsInsertPushMarker()
GraphicsInsertPopMarker()
GraphicsSetEmbeddedVsShader()
GraphicsRegisterOwner()
GraphicsRegisterResource()
GraphicsGetGpuCoreClockFrequency()
GraphicsIsUserPaEnabled()
GraphicsGetTheTessellationFactorRingBufferBaseAddress()

Figure 4. LibGraphics exported Functions

User (Program) collects commands into one cmd buffer and calls GraphicsSubmitCommandBuffers() afterwards.

```
Functions info put into the cmd buffer:
    User -> Graphics: GraphicsDrawInitDefaultHardwareState200
    User -> Graphics: GraphicsSetVsShader
    User -> Graphics: GraphicsSetPsShader
    User -> Graphics: GraphicsUpdateVsShader
...
```

cmd consists of:

```
(uint32_t) cmd[0] = KYTY_PM4(size, Pm4::IT_NOP, Pm4::R_VS_UPDATE); // Always
(Completly custom?)
(uint32_t) cmd[1..n] // Dependent on command
```

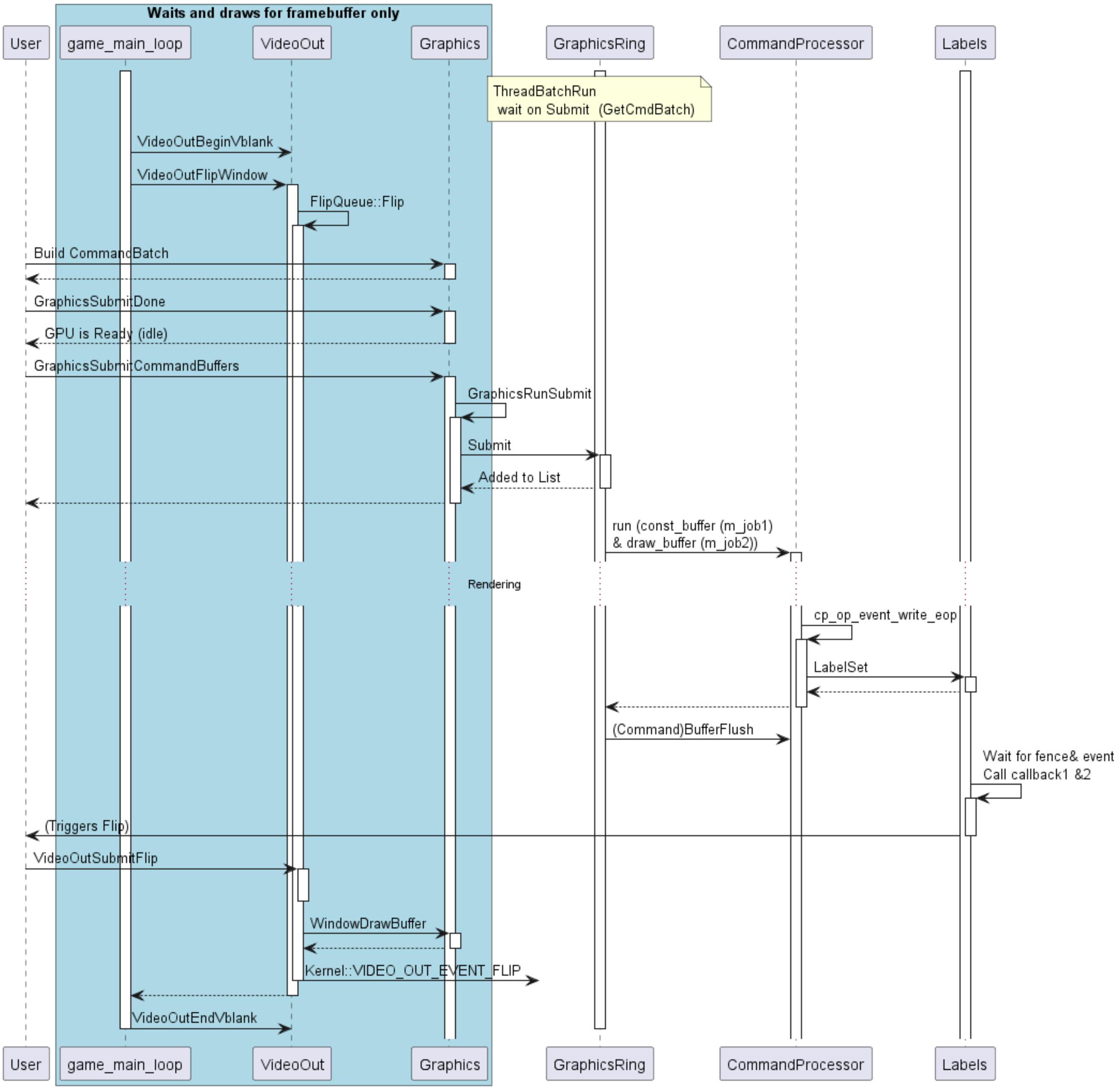


Figure 5. Sequence-Diagram Graphics-Window flip

GraphicsSubmitCommandBuffers() → GraphicsRing::ThreadBatchRun → CommandProcessor::Run() runs async for draw_buffer and const_buffer. *(const_buffer is mostly idle, draw_buffer does all the work)*

g_cp_op_func and g_hw_ctx_func, defined by graphics_init_jmp_tables(), forward these commands.

Currently defined g_cp_op_func

```
[Pm4::IT_NOP]                       :=cp_op_nop;
    Pm4::R_ZERO
    Pm4::R_VS: := hw_ctx_set_vs_shader()
    Pm4::R_PS: := hw_ctx_set_ps_shader()
    Pm4::R_CS: := hw_ctx_set_cs_shader()
    Pm4::R_DRAW_INDEX: := cp_op_draw_index()
    Pm4::R_DRAW_INDEX_AUTO: := cp_op_draw_index_auto()
    Pm4::R_DISPATCH_DIRECT: := cp_op_dispatch_direct()
    Pm4::R_DISPATCH_RESET: := cp_op_dispatch_reset()
    Pm4::R_DISPATCH_WAIT_MEM: := cp_op_wait_on_address()
    Pm4::R_DRAW_RESET: := cp_op_draw_reset()
    Pm4::R_WAIT_FLIP_DONE: := cp_op_wait_flip_done()
    Pm4::R_PUSH_MARKER: := cp_op_push_marker()
    Pm4::R_POP_MARKER: := cp_op_pop_marker()
    Pm4::R_VS_EMBEDDED: := hw_ctx_set_vs_embedded()
    Pm4::R_PS_EMBEDDED: := hw_ctx_set_ps_embedded()
    Pm4::R_VS_UPDATE: := hw_ctx_update_vs_shader()
    Pm4::R_PS_UPDATE: := hw_ctx_update_ps_shader()
[Pm4::IT_DRAW_INDEX_2]            :=cp_op_draw_index;
[Pm4::IT_INDEX_TYPE]              :=cp_op_index_type;
[Pm4::IT_NUM_INSTANCES]           :=cp_op_num_instances;
[Pm4::IT_DRAW_INDEX_AUTO]         :=cp_op_draw_index_auto;
[Pm4::IT_WAIT_REG_MEM]            :=cp_op_wait_reg_mem;
[Pm4::IT_WRITE_DATA]              :=cp_op_write_data;
[Pm4::IT_INDIRECT_BUFFER]         :=cp_op_indirect_buffer;
[Pm4::IT_EVENT_WRITE]             :=cp_op_event_write;
[Pm4::IT_EVENT_WRITE_EOP]         :=cp_op_event_write_eop;
[Pm4::IT_EVENT_WRITE_EOS]         :=cp_op_event_write_eos;
[Pm4::IT_RELEASE_MEM]             :=cp_op_release_mem;
[Pm4::IT_DMA_DATA]                :=cp_op_dma_data;
[Pm4::IT_ACQUIRE_MEM]             :=cp_op_acquire_mem;
[Pm4::IT_SET_CONTEXT_REG]         :=cp_op_set_context_reg;  => forwards to
*g_hw_ctx_func
[Pm4::IT_SET_SH_REG]              :=cp_op_set_shader_reg;
[Pm4::IT_SET_UCONFIG_REG]         :=cp_op_set_uconfig_reg;
[Pm4::IT_WRITE_CONST_RAM]         :=cp_op_write_const_ram;
[Pm4::IT_DUMP_CONST_RAM]          :=cp_op_dump_const_ram;
[Pm4::IT_INCREMENT_CE_COUNTER]    :=cp_op_increment_ce_counter;
[Pm4::IT_INCREMENT_DE_COUNTER]    :=cp_op_increment_de_counter;
[Pm4::IT_WAIT_ON_CE_COUNTER]      :=cp_op_wait_on_ce_counter;
[Pm4::IT_WAIT_ON_DE_COUNTER_DIFF] :=cp_op_wait_on_de_counter_diff;
```

Currently defined g_hw_ctx_func

```
[Pm4::DB_RENDER_CONTROL]                 = hw_ctx_set_render_control;
[Pm4::DB_STENCIL_CLEAR]                  = hw_ctx_set_stencil_clear;
[Pm4::DB_DEPTH_CLEAR]                    = hw_ctx_set_depth_clear;
[Pm4::PA_SC_SCREEN_SCISSOR_TL]           = hw_ctx_set_screen_scissor;
[Pm4::DB_Z_INFO]                         = hw_ctx_set_depth_render_target;
[Pm4::DB_STENCIL_INFO]                   = hw_ctx_set_stencil_info;
[0x08d]                                  = hw_ctx_hardware_screen_offset;
[0x08e]                                  = hw_ctx_set_render_target_mask;
```

```
[Pm4::PA_SC_GENERIC_SCISSOR_TL]          = hw_ctx_set_generic_scissor;
[Pm4::CB_BLEND_RED]                      = hw_ctx_set_blend_color;
[Pm4::DB_STENCIL_CONTROL]                = hw_ctx_set_stencil_control;
[Pm4::DB_STENCILREFMASK]                 = hw_ctx_set_stencil_mask;
[Pm4::SPI_PS_INPUT_CNTL_0]               = hw_ctx_set_ps_input;
[Pm4::DB_DEPTH_CONTROL]                  = hw_ctx_set_depth_control;
[Pm4::DB_EQAA]                           = hw_ctx_set_eqaa_control;
[Pm4::CB_COLOR_CONTROL]                  = hw_ctx_set_color_control;
[0x204]                                  = hw_ctx_set_clip_control;
[Pm4::PA_SU_SC_MODE_CNTL]                = hw_ctx_set_mode_control;
[0x206]                                  = hw_ctx_set_viewport_transform_control;
[Pm4::PA_SU_LINE_CNTL]                   = hw_ctx_set_line_control;
[Pm4::PA_SC_MODE_CNTL_0]                 = hw_ctx_set_scan_mode_control;
[Pm4::PA_SC_AA_CONFIG]                   = hw_ctx_set_aa_config;
[Pm4::PA_SC_AA_SAMPLE_LOCS_PIXEL_X0Y0_0] = hw_ctx_set_aa_sample_control;
[Pm4::VGT_SHADER_STAGES_EN]              = hw_ctx_set_shader_stages;
[0x2fa]                                  = hw_ctx_set_guard_bands;
```