ChipAte Requirements

Last updated: July 17, 2019

Prepared by: Eric Scrivner

1. Overview

ChipAte is a program, written in Rust, that emulates the Chip8 and SuperChip8 virtual machines.

2. The Chip8 Virtual Machine

The Chip8 virtual machine, insofar as it concerns ChipAte, involves the classes and objects show in Figure 1.

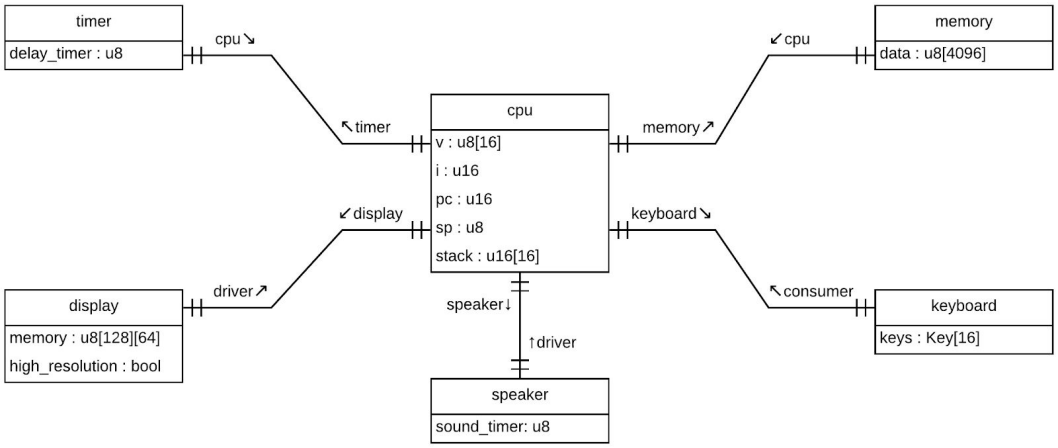

Figure 1 Classes and objects of concern to ChipAte

Class	Description	See Section
timer	Timer that counts down at a rate of 60hz.	2.2
cpu	Central processing unit responsible for executing Chip8 instructions	2.1
memory	Addressable memory store used for large and long-lived data	2.1
display	64x32 monochrome display where data can be rendered.	2.3
speaker	Single tone speaker with its own 60hz countdown timer.	2.2
keyboard	16 key hexadecimal keypad.	2.2

2.1. CPU and Memory

2.1.1. CPU

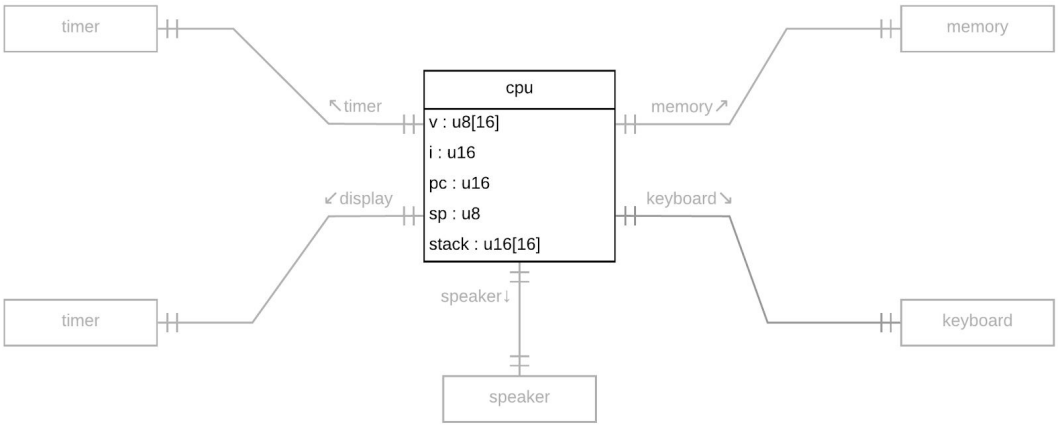

Figure 2 CPU and related classes

The *cpu* of the Chip8 is the heart of the machine and executes programs in memory instruction-by-instruction. It executes instructions at approximately 500-540 Hz.

Attribute	Description
v	The 16 general purpose 8-bit registers. Typically referenced as V0 - VF. VF is a special flag register used by some instructions.
i	A 16-bit register used to store memory addresses. Since address space only goes from 0x000 - 0xFFF, only the lower 12 bits of the register are typically used.
pc	The program counter, stores the address of the next instruction to execute. Updated after each instruction execution.
sp	8-bit register that stores the index of the top of the stack.
stack	Array of 16 16-bit address values, used to track return addresses for nested subroutine calls.

2.1.1.1. Instruction Set

All Chip8 instructions are exactly 2 bytes (16-bits) long. The first byte of each instruction should be located at an even numbered address. Chip8 programs are stored in big endian format.

The following designations are used to describe specific portions of instructions

Designation	Definition
nnn	The lowest 12 bits of an instruction.
n	The lowest 4 bits (nibble) of an instruction.
x	The lower 4 bits of the high byte of an instruction.
y	The upper 4 bits of the low byte of an instruction.
kk	The lowest 8 bits of an instruction.

The following is the complete description of the Chip8 instruction set:

Hex	Instruction	Definition
0*nnn*	SYS *nnn*	Jump to the routine at address *nnn*. pc := *nnn*
00E0	CLS	Clear screen.
00EE	RET	Return from subroutine. pc := top(stack) sp := sp - 1
1*nnn*	JP *nnn*	Unconditional jump to address *nnn*. pc := *nnn*
2*nnn*	CALL *nnn*	Call subroutine at *nnn*. push(stack, pc) sp := sp + 1 pc := *nnn*
3*xkk*	SE V*x*, *kk*	Skip the next instruction if register v*x* = *kk*.
4*xkk*	SNE V*x*, *kk*	Skip the next instruction if register v*x* ≠ *kk*.
5*xy*0	SE V*x*, V*y*	Skip the next instruction if register v*x* = v*y*.
6*xkk*	LD V*x*, *kk*	Set register v*x* := *kk*.
7*xkk*	ADD V*x*, *kk*	Set register v*x* := v*x* + *kk*.
8*xy*0	LD V*x*, V*y*	Set register v*x* := v*y*.
8*xy*1	OR V*x*, V*y*	Set register v*x* := v*x* ⋁ v*y*.
8*xy*2	AND V*x*, V*y*	Set register v*x* := v*x* ^ v*y*.

8*xy*3	XOR V*x*, V*y*	Set register v*x* := v*x* ˇ v*y*.
8*xy*4	ADD V*x*, V*y*	Set register v*x* := v*x* + v*y*. Set register v*f* := 1 if result overflows 8-bits and is > 255, otherwise set register v*f* := 0.
8*xy*5	SUB V*x*, V*y*	Set register v*x* := v*x* - v*y*. Set register v*f* := 1 if v*x* > v*y*, otherwise set register v*f* := 0.
8*xy*6	SHR V*x*	Set register v*x* := v*x* >> 1. Set register v*f* := 1 if the least significant bit of v*x* is 1, otherwise set v*f* := 0.
8*xy*7	SUBN V*x*, V*y*	Set register v*x* := v*y* - v*x*. Set register v*f* := 1 if v*y* > v*x*, otherwise set register v*f* := 0.
8*xy*E	SHL V*x*	Set register v*x* := v*x* << 1. Set register v*f* := 1 if the most significant bit of v*x* is 1.
9*xy*0	SNE V*x*, V*y*	Skip the next instruction if v*x* ≠ v*y*.
A*nnn*	LD I, *nnn*	Set register i := *nnn*.
B*nnn*	JP V0, *nnn*	Jump to address *nnn* + v0. pc := *nnn* + v0
C*xkk*	RND V*x*, *kk*	Set v*x* := rand8() ^ *kk*. Where rand8() returns a random 8-bit value.
D*xyn*	DRW V*x*, V*y*, *n*	Load *n*-byte sprite from address *i*. Display on screen at location (v*x*, v*y*). Sprites are XORed onto the frame buffer. VF is set to 1 if a pixel goes from a 1 to a 0, otherwise it is set to 0. Drawing wraps around if it goes past the edge of the screen.
E*x*9E	SKIP V*x*	Skip the next instruction if the key with value v*x* is pressed.

E*x*A1	SKNP V*x*	Skip the next instruction if the key with value v*x* is not pressed.
F*x*07	SET V*x*, DT	Set register v*x* := value of delay timer.
F*x*0A	LD V*x*, K	Wait for a key press, store the value in v*x*. All execution stops until a key is pressed.
F*x*15	LD DT, V*x*	Set delay timer := v*x*.
F*x*18	LD ST, V*x*	Set sound timer := v*x*.
F*x*1E	ADD I, V*x*	Set register i := i + v*x*.
F*x*29	LD F, V*x*	Set i := address of sprite for digit v*x*.
F*x*33	LD B, V*x*	Store the BCD representation of V*x* at addresses i, i + 1, and i + 2.
F*x*55	LD [I], V*x*	Store registers v0 through v*x* in memory starting at location i.
F*x*65	LD V*x*, [I]	Read values starting at memory location i into register v0 through v*x*.

The Super-Chip8 adds the following additional opcodes:

Hex	**Instruction**	**Definition**
00C*n*	SCD *n*	Scroll the display down *n* lines.
00FB	SCR	Scroll the display 4 pixels right.
00FC	SCL	Scroll the display 4 pixels left.
00FD	EXIT	Exit CHIP interpreter.
00FE	LOW	Disable extended screen mode from 64x128 to 32x64 pixels.
00FF	HIGH	Enable extended screen mode from 32x64 to 64x128 pixels.
D*xy*0	DRW V*x*, V*y*, 0	Show at 16x16 sprite starting from address i at (V*x,* V*y*).
F*x*30	LD F10, V*x*	Set i := address of 10 byte sprite for digit v*x*.
F*x*75	LD R, V*x*	Store registers v0 through v*x* in RPL user flags. $x \leq 7$.
F*x*85	LD V*x*, R	Load registers v0 through v*x* from RPL user flags. $x \leq 7$.

R-1 ChipAte supports both Chip8 and SuperChip8 instruction sets.

2.1.2. Memory

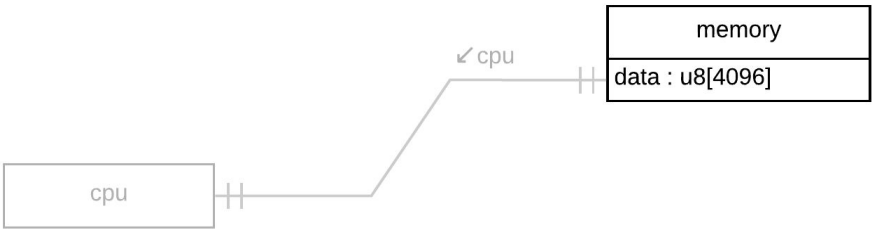

Figure 3 Memory and related classes

Attribute	Description
data	4,096 bytes of addressable memory.

The Chip8 contains 4,096 bytes of addressable memory from address 0x000 (0) to address 0xFFF (4,095).

R-2 Chip8 programs are loaded into address 0x200 (512)

2.2. Timer, Speaker, and Keyboard

2.2.1. Timer

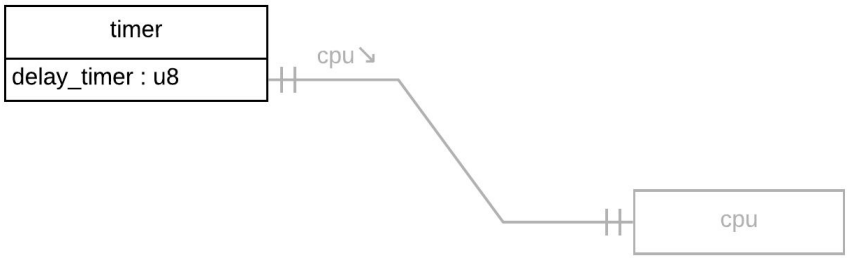

Figure 4 Timer and related classes

Attribute	Description
delay_timer	8-bit delay timer, counts down to 0 at a rate of 60Hz when > 0.

The Chip8 contains a delay timer. The delay timer has a single register, dt. When dt is > 0, it counts down to zero by subtracting one at a rate of 60Hz.

2.2.2. Speaker

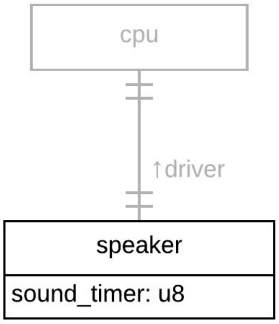

Figure 5 Speaker and related classes

Attribute	Description
sound_timer	8-bit sound timer, counts down to 0 at a rate of 60Hz when > 0.

The Chip8 also contains a speaker that produces a single tone.

R-3 ChipAte will play a middle C tone (262Hz) when the speaker is on.

The speaker contains its own sound timer. The sound timer has a single register `st`. When `st` is > 0, it counts down to zero by subtracting one at a rate of 60Hz.

The speaker plays the same tone for as long as the sound timer is non-zero.

2.2.3. Keyboard

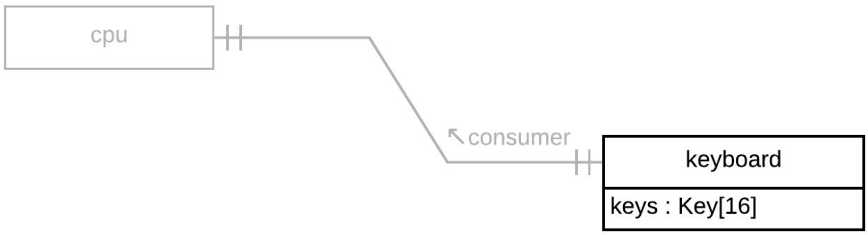

Figure 6 Keyboard and related classes

Attribute	Description
keys	16 available keys on the Chip8 keyboard. This is a key map indicating whether or not a given key is pressed.

The keyboard on the Chip8 had the following layout associating keys with hexadecimal values:

1	2	3	C
4	5	6	D
7	8	9	E
A	0	B	F

R-4 ChipAte should map the keys onto the keyboard in the same layout as the original, as shown below:

1	2	3	4
Q	W	E	R
A	S	D	F
Z	X	C	V

2.3. Display

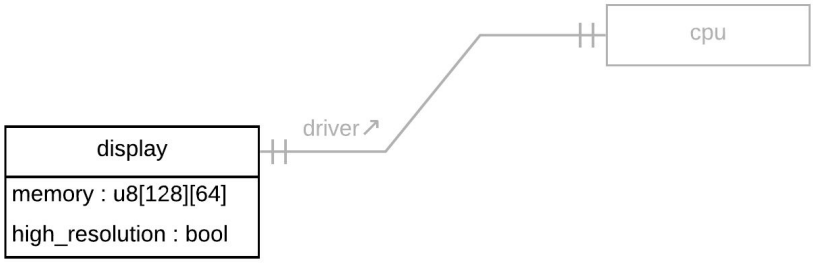

Figure 7 Display and related classes

Attribute	Description
memory	128x64 pixel frame buffer used for rendering video. In the default case only 64x32 pixels are available for usage. When high_resolution mode is active, the whole frame buffer is available for usage.
high_resolution	Indicates whether or not the display is operating in high resolution (128x64 pixel) mode.

R-5 The display frame buffer initially contains all 0 values.

R-6 The display is initially in low resolution mode.

2.3.1. Digit Sprites

The Chip8 virtual machine contains a set of font sprites representing the hexadecimal digits 0 - F.

R-7 The digit sprites must be located somewhere in the address space 0x000 - 0x1FF. They are defined in the following table:

Digit	Sprite Bytes	Pixel Representation
0	0xF0 0x90 0x90 0x90 0xF0	**** *..* *..* *..* ****
1	0x20 0x60 0x20 0x20 0x70	..*. .**. ..*. ..*. .***
2	0xF0 0x10 0xF0 0x80 0xF0	**** ...* **** *... ****
3	0xF0 0x10 0xF0 0x10 0xF0	**** ...* **** ...* ****
4	0x90 0x90 0xF0 0x10 0x10	*..* *..* **** ...* ...*
5	0xF0	****

	0x80 0xF0 0x10 0xF0	*... **** ...* ****
6	0xF0 0x80 0xF0 0x90 0xF0	**** *... **** *..* ****
7	0xF0 0x10 0x20 0x40 0x40	**** ...* ..*. .*.. .*..
8	0xF0 0x90 0xF0 0x90 0xF0	**** *..* **** *..* ****
9	0xF0 0x90 0xF0 0x10 0xF0	**** *..* **** ...* ****
A	0xF0 0x90 0xF0 0x90 0x90	**** *..* **** *..* *..*
B	0xE0 0x90 0xE0 0x90 0xE0	***. *..* ***. *..* ***.
C	0xF0 0x80 0x80 0x80 0xF0	**** *... *... *... ****
D	0xE0	***.

	0x90 0x90 0x90 0xE0	*..* *..* *..* ***.
E	0xF0 0x80 0xF0 0x80 0xF0	**** *... **** *... ****
F	0xF0 0x80 0xF0 0x80 0x80	**** *... **** *... *...

3. Open Questions

- ~~Should the sound and delay timers be coupled together or separately?~~

A. References

S-CHIP8 V1.1 Specification	SuperChip8 specification including new instructions and screen mode description.
CowGod's Chip8 Technical Reference	Very nice, concise, and thorough Chip8 reference documentation.
CPU Emulation Course At Cornell	Useful additional documentation on SuperChip8 opcodes here.