Psycle Developer Guide

C-Version, Feb 2021 (unfinished)

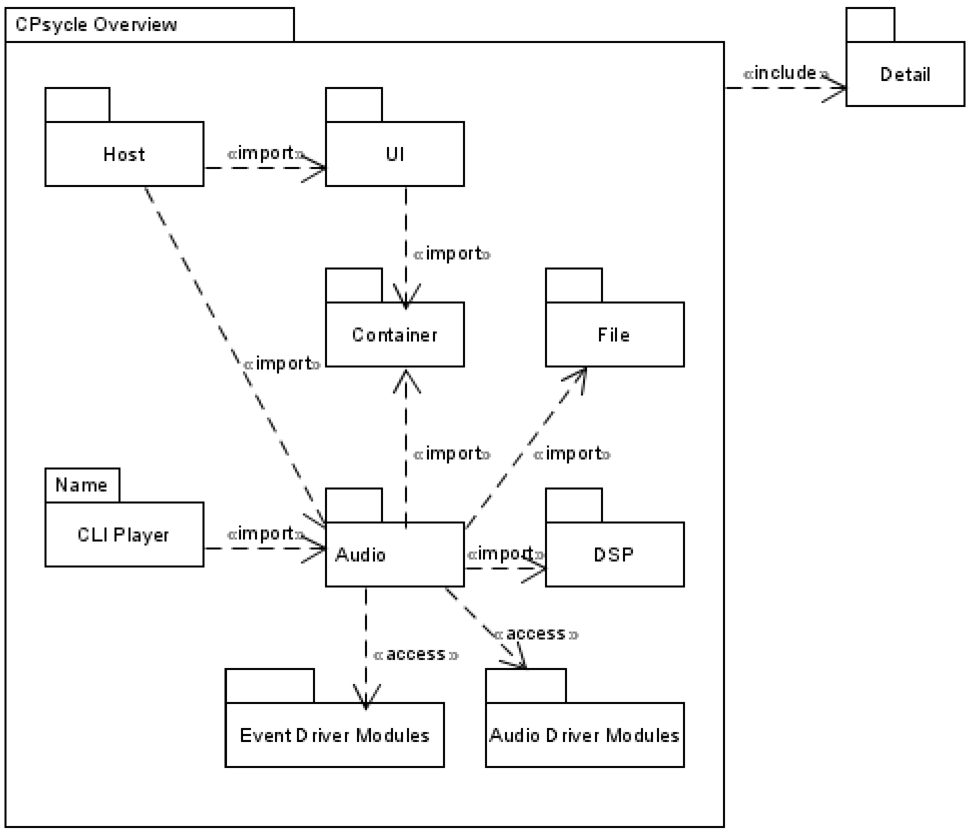
Structure of Psycle

Psycle is divided in two subsystems,

- the audio engine and
- the ui

The helpers dsp, container and file assist these subsystems.

The audio and event drivers will be loaded by the audio engine at runtime.



The Host

The host is the application the user starts. The program begins in psycle.c, first, initializing the ui, creating the mainframe and starting the ui event loop and finally leaves the program, if a close event occurs.

Depending on the platform the main entry point differs. Windows uses WinMain as entry point, other toolkits will use int main(int argc, char** argv)

DIVERSALIS, a collection of platform specific defines, is used to detect the platform and needed main entry. To leave the platform dependend startup, psycle_run is called as platform independed main entry point.

psycle_run:

1. The app is added to the enviremont path, that the scilexer module can be found.

Scilexer is an open source editor component, psycle uses since version 1.12.

2. psy_ui_app_init(&app, psy_ui_DARKTHEME, instance);

The ui is initialized with a darktheme. (psy_ui_LIGHTHEME starts a light theme).

3. mainframe = (MainFrame*)malloc(sizeof(MainFrame));

Mainframe is found in mainframe.c and is the composite of all views psycle will present to the user. Workspace holds the project song and configurations.

4. psy_ui_component_showstate(&mainframe->component, SW_MAXIMIZE);

Psycle is displayed.

5. Now the app main loop is started:

err = psy_ui_app_run(&app);

The main loop waits for an event (mouse, keyevent..) and triggers callbacks to the host. In case of a close event the main loop will be terminated.

6. The heap storage of the mainframe is cleaned up and the env path is restored

7. The app returns the status to the operation system and the process is terminated.

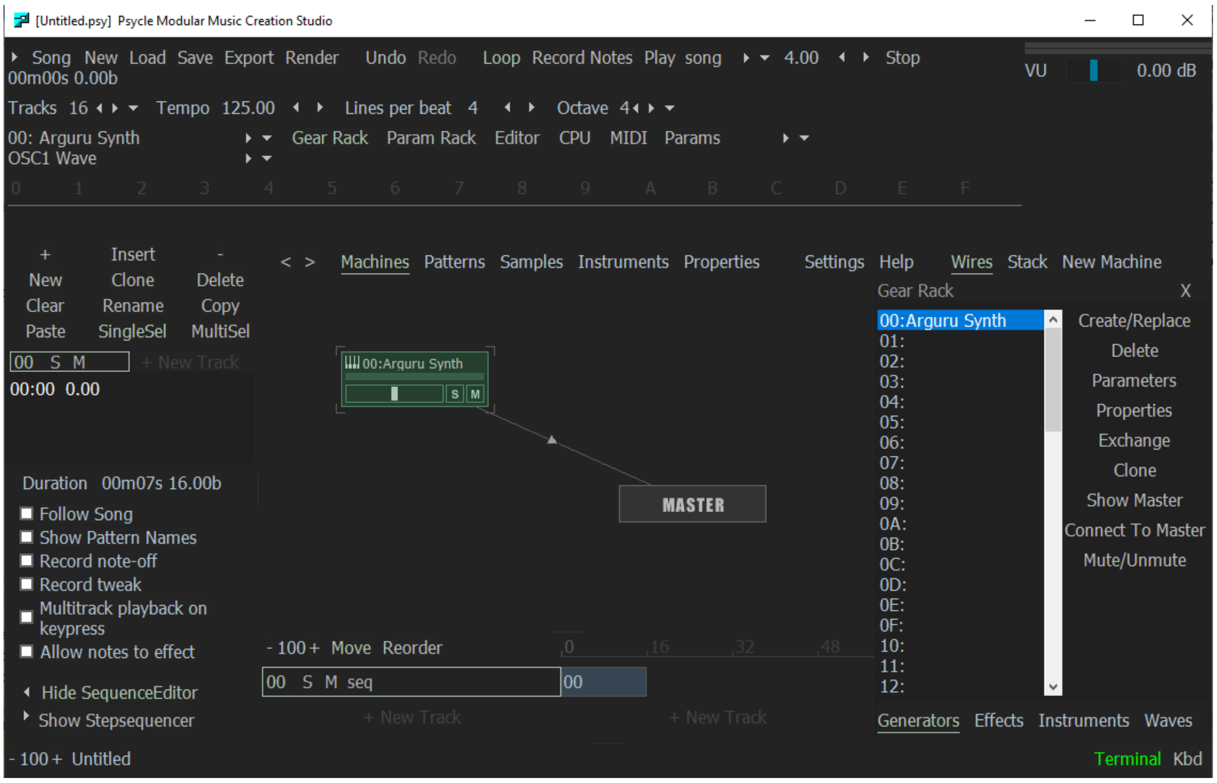
The Mainframe

The mainframe consists oft two basic systems:

The ui views and the workspace.

1. Machines, Patterns, Samples, Instruments, Help, SequencerView, and more …
2. The workspace contains the current song, owns the audio players and the configuration files.

Psycle is mainview orientated decorated with sidebars. The mainview structure is realized with a psy_ui_Notebook, that allows to switch components. Tabbar controls the notebook and offers the user tabs to switch it.



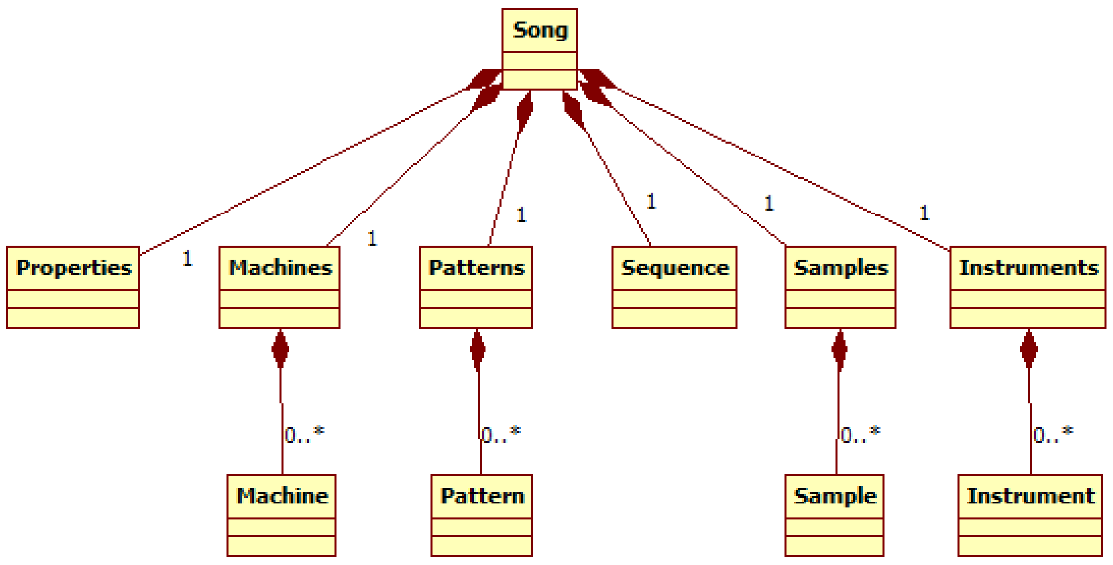
Workspace

The workspace connects the player with the psycle host ui and configures both

```
psy_audio_MachineCallback
       ^
     Workspace
         <>---- PsycleConfig;              host
         <>---- ViewHistory
         <>---- psy_audio_Player;          audio imports
         <>---- psy_audio_MachineFactory
         <>---- psy_audio_PluginCatcher
         <>---- psy_audio_Song
```

The Song

psy_audio_Song hold everything comprising a "tracker module", this include patterns, pattern sequence, machines and their initial parameters and coordinates, wavetables, ...



Song hold everything comprising a "tracker module", this include patterns, sequence, machines, samples(wavetables), and instruments

The Player

The player controls the audio drivers. It loads a driver and has a work callback. The audio driver will call them if the soundcard buffer needs to be filled. Usually the player work callback will be in the audio driver thread (different than the ui thread).

Work callback:

psy_dsp_amp_t* psy_audio_player_work(psy_audio_Player* self, int* numsamples, int* hostisplaying)

The Player first fills its own buffer defined in player:

static psy_dsp_amp_t bufferdriver[MAX_SAMPLES_WORKFN];

with the numsamples the driver asked for. The buffer will be interleaved (mix of the channels in one stream.) Work returns a pointer to this interleaved buffer and the audio driver can use (copy) the needed frames. To fill this buffer the player has a reference to a song. With help oft the song, the player has a helper, psy_audio_Sequencer, that processes the patterns oft the song. The player will then work the machines with the current events, the sequencer has collected. psy_audio_Machines compute the machine path determined by psy_audio_Connections, the wires of the Machines, from their leafs to the Master.

Player will not work all samples at once, but splits the fill amount into psy_audio_MAX_STREAM_SIZE chunks (256). This is done for psycle plugins to maintain compatibility. (e.g. the sampler ps1 slides used this amount as tick). To maintain compatibility the amount is further splitted if a new trackerline event occurs to notify the machines of this change. This is done because psycle was a pure tracker and worked this way:

1. Work machines till all frames of a line are worked
2. Notify machine line has ended
3. Execute global events (e.g. bpm change)
4. Notify all machines a newline has started
5. Execute line events of the pattern (send a seqtick to the machines).The plugins will setup their voices to work the events
6. Repeat step 1 until all numsamples are worked and return buffer to the audiodriver

The new version is sequencer orientated. This has two mayor differences.

1. The timestamp of an event isn‛t the line number, but a position in beats.
2. The player playposition is in beats, not in line numbers, frames or ticks

Events can occur at any time, not only at line start.

The sequencer works this way:

1. Calculate amount of frames of the interval to beats using the current bpm tempo.
2. Update linetickcount (in beats) and split current amount to notify linecount and send events that are on linestart. (for compatibility)
3. Search for all pattern events in this time interval
4. Timestamp all events of this interval with an offset in frames relative to the splitted interval start.
5. Work machine with this event list
6. Increase playposistion with interval beat time.
7. Repeat step 1 until all numsamples are worked and return buffer to the audiodriver

Vst plugins and psycle effects work without problems, because they dont rely on ticks. Problemematic are the samplers psycle uses. Sampler PS1 was rewritten, XMSampler isn't tested enough. For this machines the linetickcount (in beat unit) splits the work amount again to ensure the samplers get their events at linestart. For events occuring inside, the samplers will have problems with the effect processing. On the other side, this newline split can make problems at some vsts that usually are getting equal process intervals. This problem have all programs that maintain tracker functionality. A vst plugin should handle every amount of samples correct.

Retrigger/Delay/etc..

Since psycle used lines/ticks instead of beat timing and events occured only at line start, positioning in one line was done using a tick offset and a global retrigger delay array in the player. Each plugin api had an own part to handle this retrigger delay code. The sequencer insted delays events changing the beat position and new retrigger events are cloned and put at the later beat positions. Smooth tweaks (tws) are generated the same way generating 64 new tws events matching the line.

Ticks occur in the samplers, too. The samplers alter the effects each tick (standard is 24 ticks per beat). The samplers used the global player retrigger array to time this ticks. This was replaced by a new class TickTimer, that is resetted at each newline event. Each sampler has an own instance of a TickTimer. This removes the need to work with the player but adds redundancy counting the ticks. The Extraticks per beat were introduced to handle bpm timings better. To reimplement this, the sequencer lpb speed will be altered to match this extra ticks. Normally this isn't needed anymore because the play time can be changed continious by the lpb speed in the sequencer.

Accuracy

Since the tracker engine counted frames and the sequencer adds up real values, rounding errors occur. This means, the tick timer in the sampler may differ by one or more samples over the timer or the seqtick at line start may be some samples offseted. This is corrected by the ticktimer class resetting the counter at each newline notify.