



**Инструменты профилирования в
геймдеве и кейсы использования**

Обо мне



Lead developer, Playrix, Homescapes - tech team

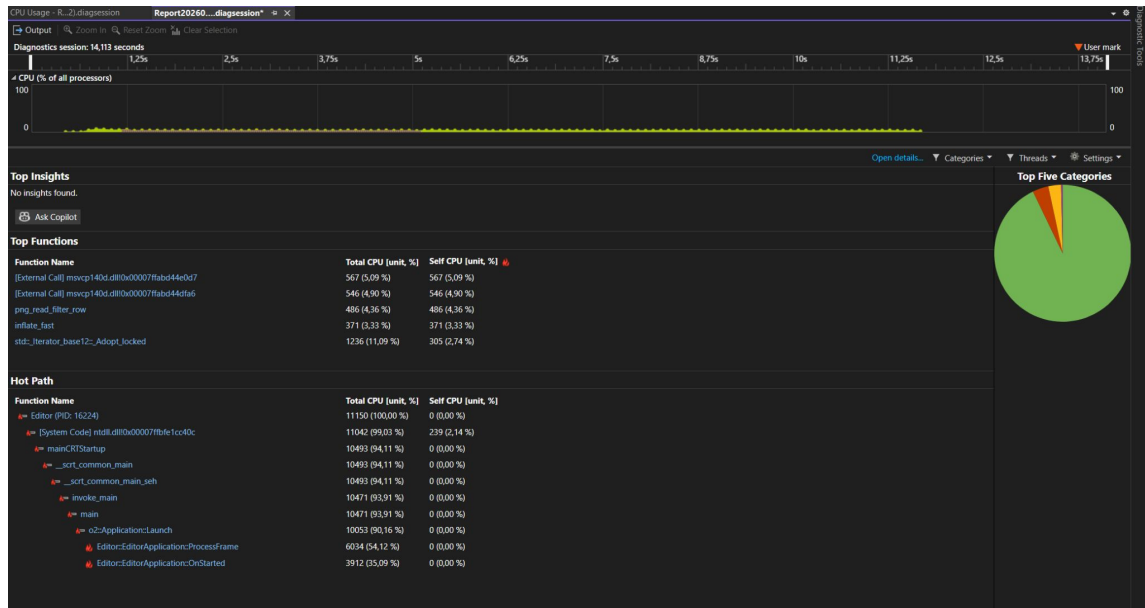
- 14 лет в C++ и геймдеве
- 10+ успешных проектов в продакшене
- 6 лет в playrix:
 - Работал на движке ([Движок VSO: Под капотом нашего редактора](#))
 - Работал на Township
 - Работаю на Homescapes
- Пишу игровой движок (<https://github.com/o2-engine/o2>)
- Пишу статьи на хабре (<https://habr.com/ru/users/anz/articles/>)

Какой профайлер выбрать

- Посмотреть “обзорно” - сэмплирующие
 - xcode instruments, MSVS
- Посмотреть “детально” - инструментрующие
 - tracy, optick, внутренние движковые, кастомные
- Статические - PVS
 - Рекомендации микрооптимизаций
- GPU
 - RenderDoc, xcode, Nsight
- Заглянуть в “железо”
 - vendor-specific, intel vtune

MSVS

- Сэмплирующий
- Встроен в IDE
- Очень тяжелый
- Подходит для базового анализа



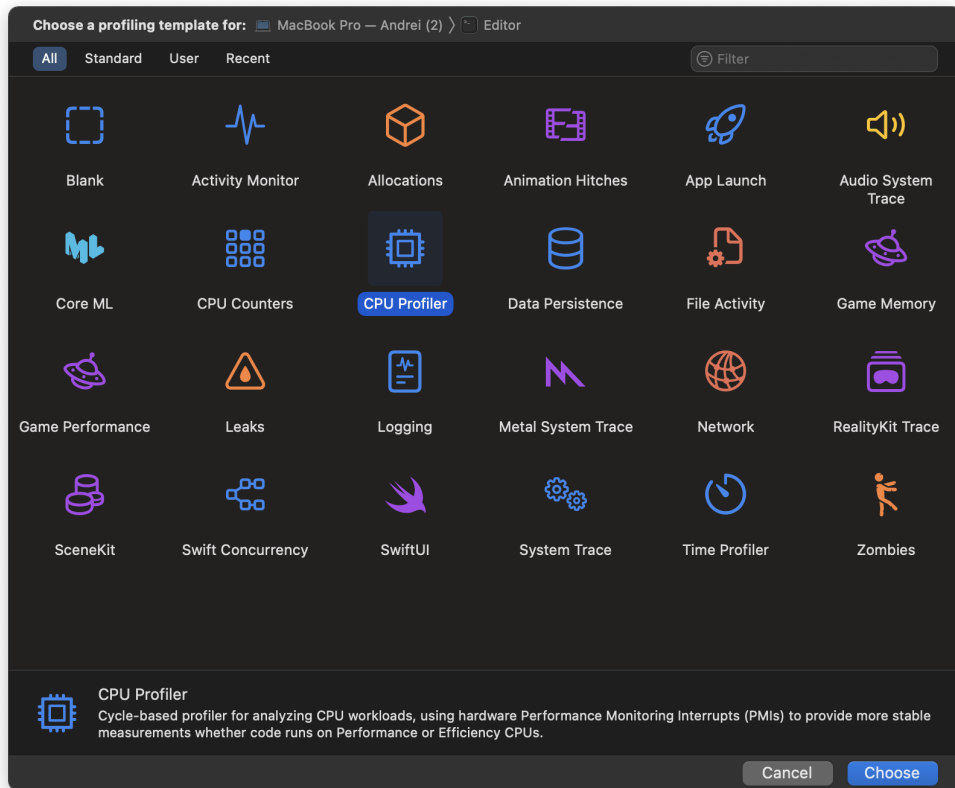
MSVS

- Профайлинг скриптового движка



xCode instruments

- Топ для работы на ios/mac
- Содержит в себе массу инструментария:
 - Анализ общей нагрузки
 - CPU
 - GPU, только metal
 - Память, утечки
 - ...
- Быстрый, в отличие от MSVS
- Удобный



xCode instruments - time profiling

- Сэмплирующий
- Изучение загрузки приложения
- Найден боттлнек - работа с файлами со стороны ОС

Weight	Self Weight	Symbol Name
5.02 s 100.0%	0 s	g (650)
634.00 ms 12.6%	634.00 ms	stat libsystem_kernel.dylib
595.00 ms 11.8%	0 s	> boost::filesystem::detail::status(boost::filesystem::path const&, boost::system::error_code*) g
22.00 ms 0.4%	0 s	> _validateVolume CacheDelete
4.00 ms 0.0%	0 s	> boost::filesystem::detail::last_write_time(boost::filesystem::path const&, boost::system::error_code*) g
3.00 ms 0.0%	0 s	> boost::filesystem::is_regular_file(boost::filesystem::path const&, boost::system::error_code&) [inlined] g
2.00 ms 0.0%	0 s	> _NSFileExists Foundation
1.00 ms 0.0%	0 s	> -[NSURL(NSURL) initWithPath:] Foundation
1.00 ms 0.0%	0 s	> _CFBundleCouldBeBundle CoreFoundation
1.00 ms 0.0%	0 s	> boost::filesystem::directory_entry::get_status(boost::system::error_code*) const g
1.00 ms 0.0%	0 s	> swift_copyAuxiliaryExecutablePath libswiftCore.dylib
1.00 ms 0.0%	0 s	> glpCacheOpen libGLProgrammability.dylib
1.00 ms 0.0%	0 s	> stat libsystem_kernel.dylib
1.00 ms 0.0%	0 s	> MTLCompilerFSCache::openSync() Metal
1.00 ms 0.0%	0 s	> _NSFileExistsAtPath Foundation
286.00 ms 5.6%	286.00 ms	> mach_msg2_trap libsystem_kernel.dylib
245.00 ms 4.8%	245.00 ms	> swtch_pri libsystem_kernel.dylib
245.00 ms 4.8%	0 s	> cthread_yield libsystem_pthread.dylib
245.00 ms 4.8%	0 s	> std::__1::__libcxx_thread_yield[abi:un170006]() [inlined] g
245.00 ms 4.8%	0 s	> GardenField::CreateGardenObjectsCoro(Core::Coro::ExecutionContext const&) (.resume) g
150.00 ms 2.9%	150.00 ms	> _platform_memmove libsystem_platform.dylib
148.00 ms 2.9%	148.00 ms	> __read_nocancel libsystem_kernel.dylib
127.00 ms 2.5%	127.00 ms	> __open_nocancel libsystem_kernel.dylib
93.00 ms 1.8%	93.00 ms	> lstat libsystem_kernel.dylib

xCode instruments - memory

- Сравнение снимков
- Ищем утечки памяти

The screenshot displays the Xcode Instruments Memory tool. The left pane shows a list of memory snapshots with columns for Snapshot, Timestamp, Growth, and # Persistent. The right pane shows a call stack for the selected snapshot.

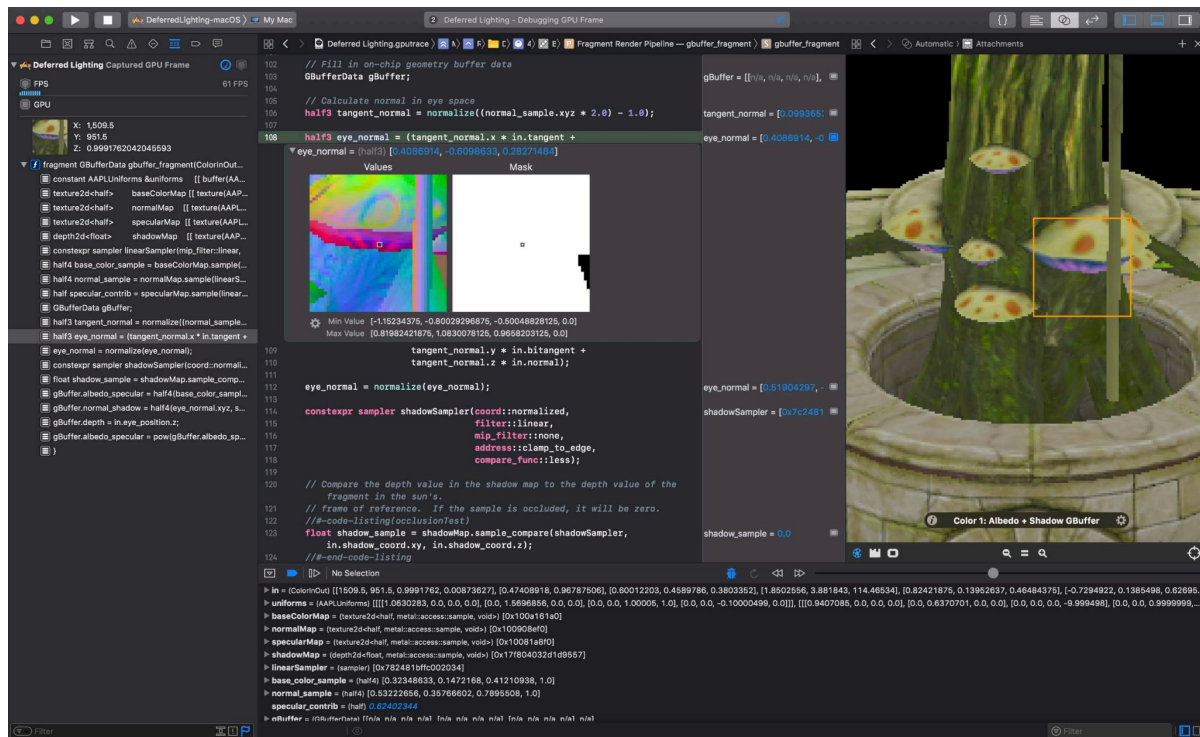
Snapshot	Timestamp	Growth	# Persistent
> Generation E	01:09.016.756	319,21 MiB	1.401.769
> Generation F	01:09.034.673	39,17 MiB	415
> VM: IOAccelerator		38,53 MiB	11
0x15c4d4000	01:09.020.338	16,00 MiB	
0x15b4d4000	01:09.020.284	16,00 MiB	
0x158000000	01:09.020.640	2,03 MiB	
0x1568b0000	01:09.021.544	1,83 MiB	
0x156da0000	01:09.019.392	1,31 MiB	
0x154408000	01:09.019.122	480,00 KiB	
0x154480000	01:09.019.894	256,00 KiB	
0x155000000	01:09.020.100	256,00 KiB	
0x155200000	01:09.020.496	256,00 KiB	
0x1538a0000	01:09.020.390	128,00 KiB	
0x13df30000	01:09.021.421	16,00 KiB	
> GFX Texture Level		576,00 KiB	3
> < non-object >		52,45 KiB	278
> AGX:10FamilyTexture		5,84 KiB	11
> AGX::G9::Texture		4,84 KiB	11
> GLDTextureRec		2,23 KiB	11
> MTLTextureDescriptorInternal		2,23 KiB	11
> IOGPUResource		1,38 KiB	11
> MTLSamplerDescriptorInternal		880 Bytes	11
> GLDSamplerRec		704 Bytes	11
> GLRTextureResource		704 Bytes	11
> GLRTextureViewResource		480 Bytes	10
> CFString (immutable)		272 Bytes	4
> NSURLRequestInternal		240 Bytes	1
> CFHTTPMessage		240 Bytes	1

The right pane shows a call stack for the selected snapshot. The stack includes the following frames (from top to bottom):

- GLDFramebufferRec::load()
- glLoadFramebuffer
- gleCheckFramebufferStatus
- glCheckFramebufferStatusEXT_Exec
- Core::Render::RenderDeviceGLES2Impl::CheckFramebufferStatus()
- Core::Render::RenderDeviceGLES2::UploadInternal(Core::Render::Target*)
- void Core::Render::RenderQueue::RunInRenderThreadAndWait<Core::Render::Target*>(void (*)(void*), Co...
- Core::Render::RenderDeviceInterface::Upload(Core::Render::Target*)
- Core::ResourceLoader::EndLoad(Core::Resource*, Core::ResourceLoader::AsyncLoader*, bool, char const*)
- Core::ResourceLoader::Load(Core::Resource*, Core::ResourceLoadMode)::\$_46::operator()(bool, char co...
- decltype(std::declval<Core::ResourceLoader::Load(Core::Resource*, Core::ResourceLoadMode)::\$_46>(...
- void std::_1::__invoke_void_return_wrapper<void, true>::__call<Core::ResourceLoader::Load(Core::Reso...
- std::_1::__function::__alloc_func<Core::ResourceLoader::Load(Core::Resource*, Core::ResourceLoadMod...
- std::_1::__function::__func<Core::ResourceLoader::Load(Core::Resource*, Core::ResourceLoadMode)::\$_...
- std::_1::__function::__value_func<void (bool, char const*)>::operator()([abi:v160006])(bool&&, char const...
- Core::AsyncWorkingQueue::AsyncHandlerItemImpl<void, bool>::Complete()
- Core::AsyncWorkingQueue::ExecuteReady(unsigned long, std::_1::chrono::duration<long long, std::_1::r...
- Core::ApplicationBase::UpdateAsyncWorkingQueue()
- Core::ApplicationBase::MainLoopContent(Core::MainLoopMode)
- [EAGLView drawView:]

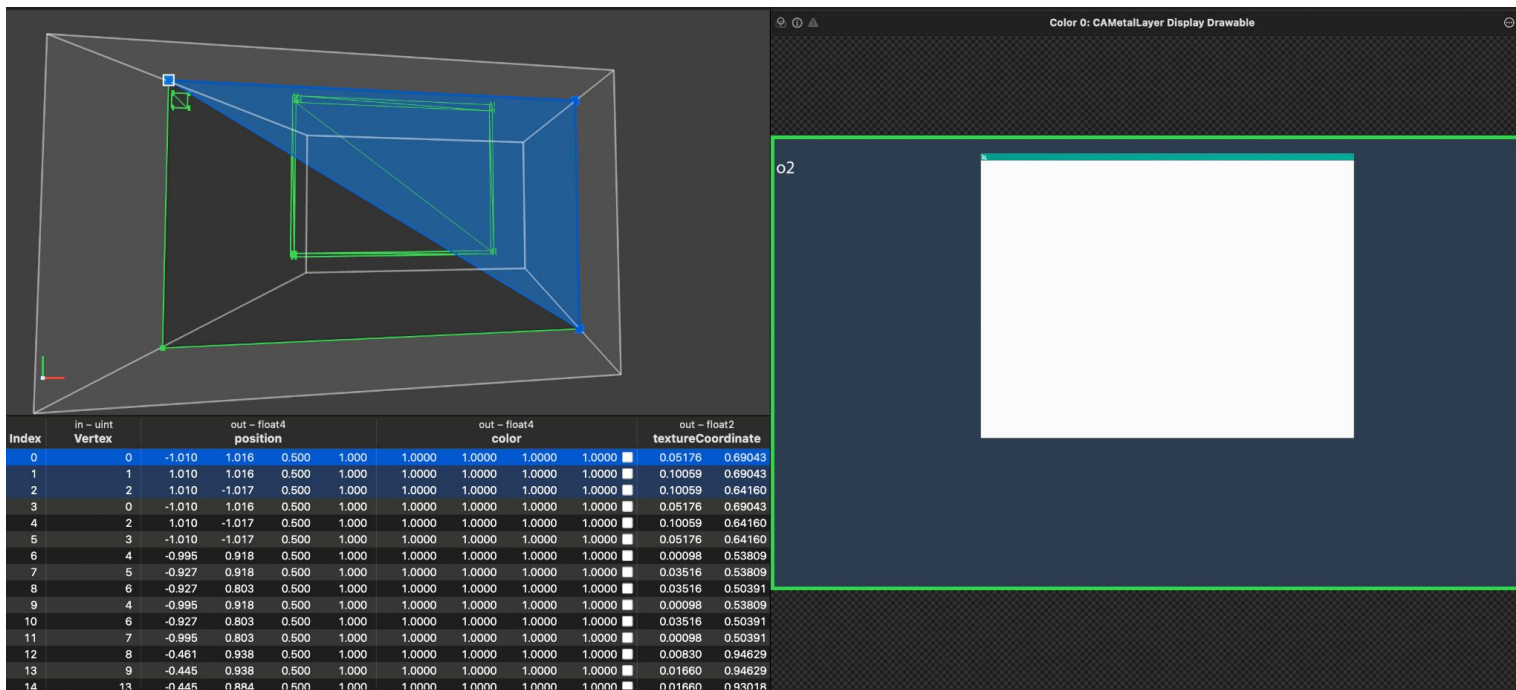
xCode instruments - gpu

- Лучший профайлер что я видел
- Заточен под ios/mac и metal



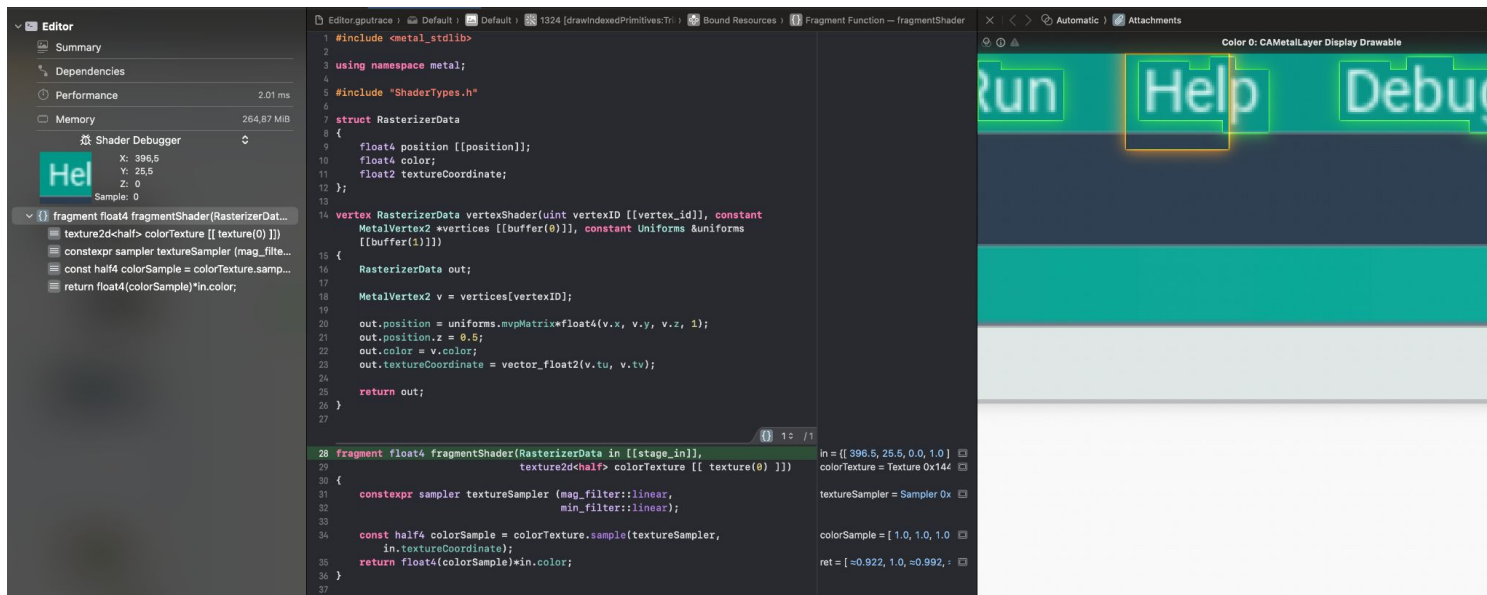
xCode instruments - gpu

- Проблема первого треугольника - где он?



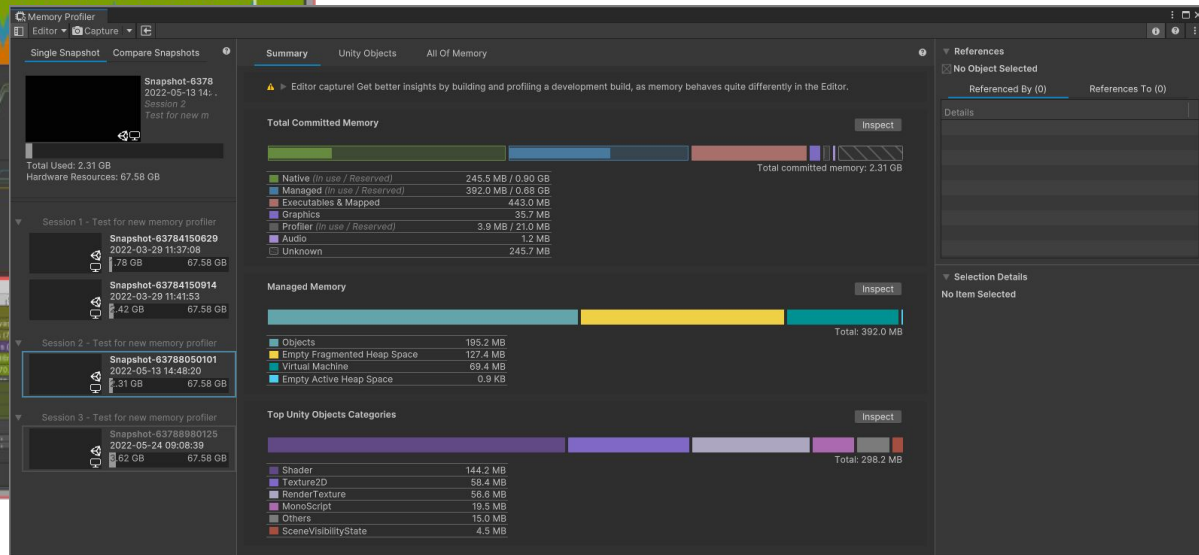
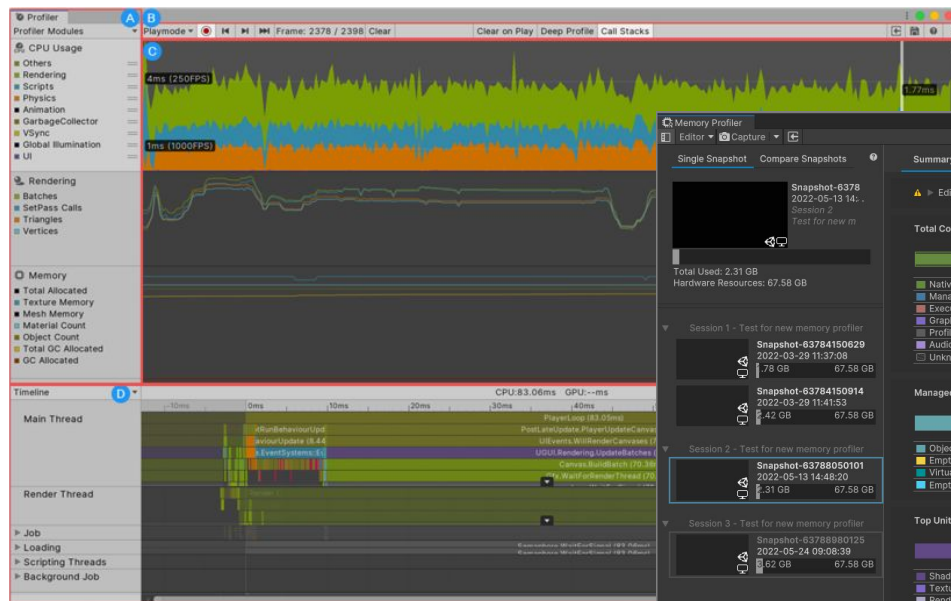
xCode instruments - gpu

- Отладка шейдеров. Можно отладить конкретный пиксель



Unity3D profilers

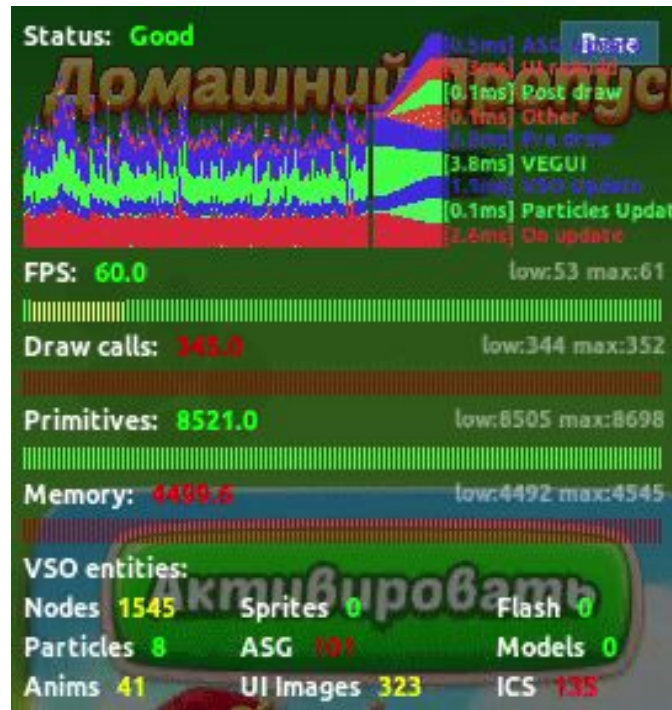
- Заточены под Unity3D
- Профайлинг CPU, GPU, Memory



Кастомщина - playrix perfmon

- Быстрый анализ через цветовую дифференциацию
- Специфичные метрики движка
- ImGui

https://github.com/zenkovich/imgui_perfmon

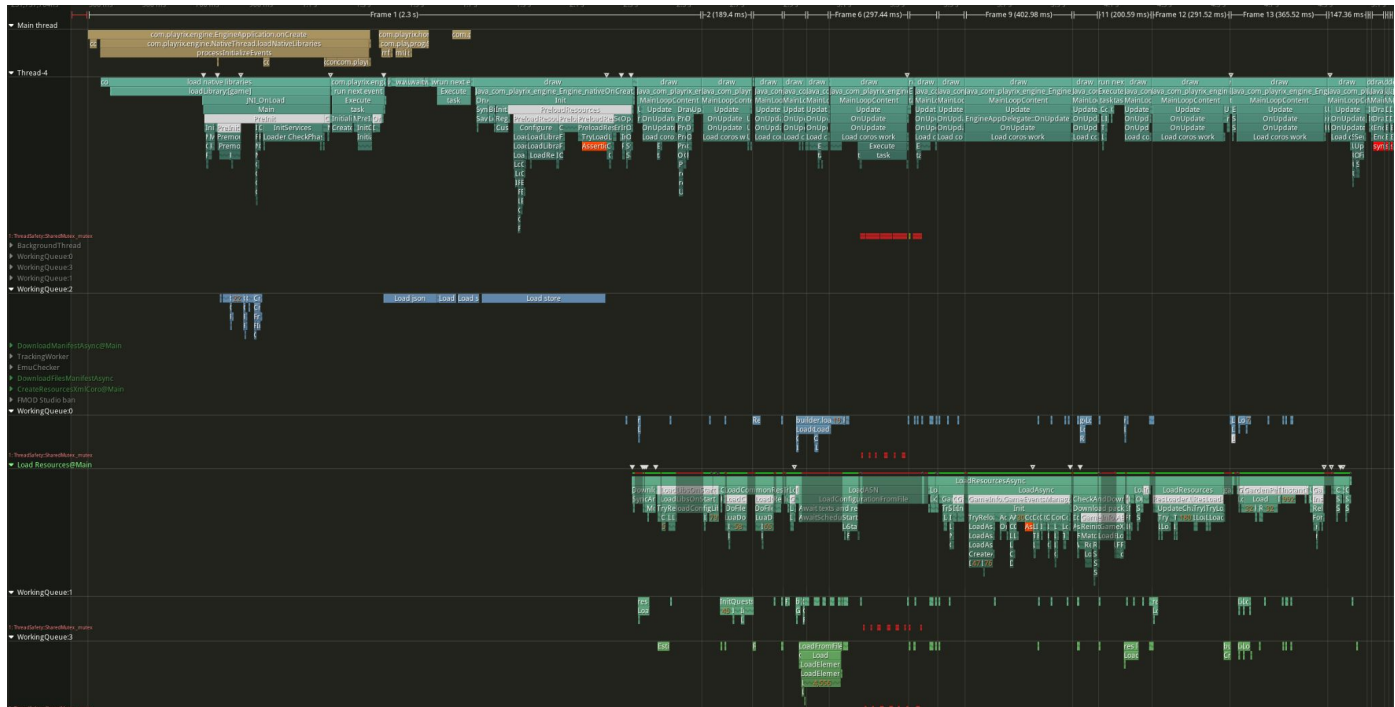


Кастомщина - логи!!1

- Супер-просто покрыть код замерами
 - `std::chrono::system_clock::now()`
- Профайлить там, где ничего другого нет
 - Платы на linux для гемблинга
- Выводить в лог дампы
 - Легко смотреть и сравнивать

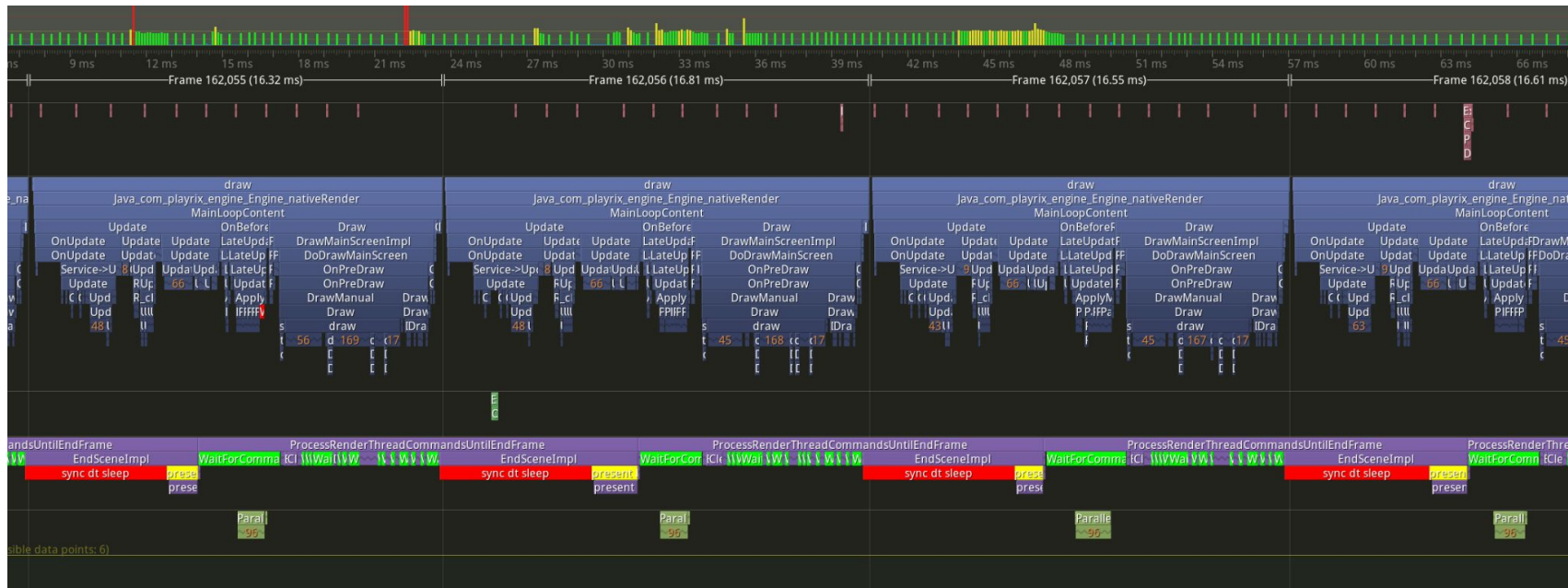
Трасу - профайлинг загрузки игры

Флоу загрузки нашей игры через tracy



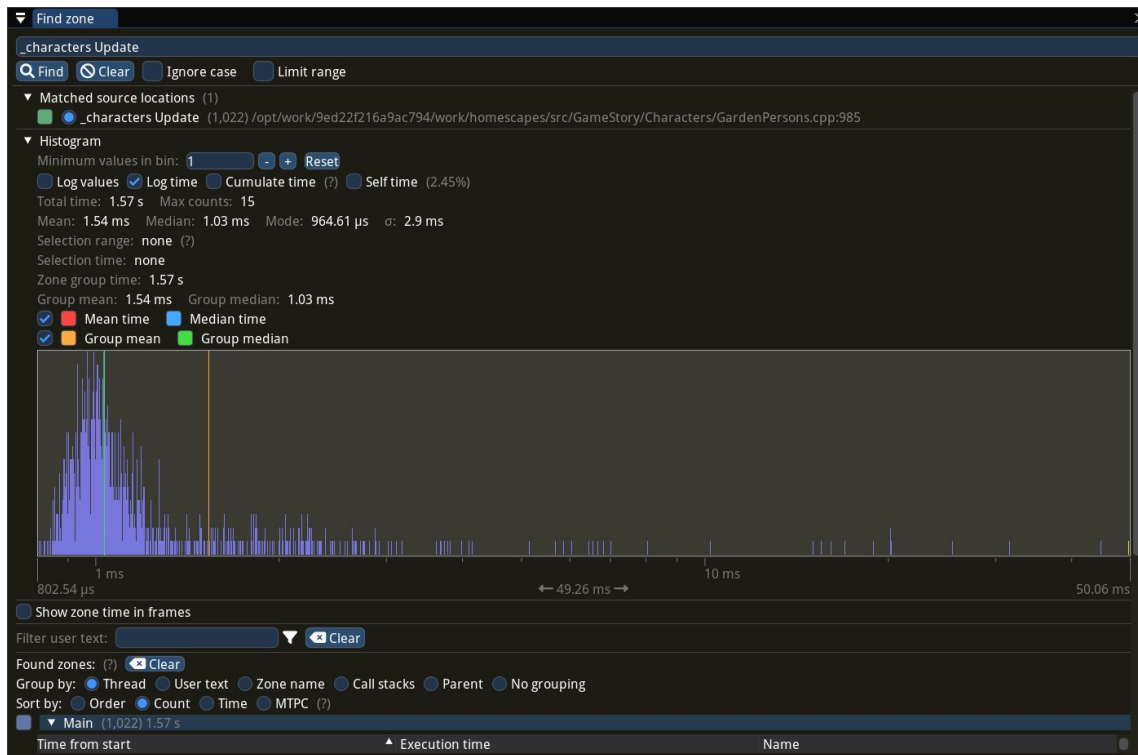
Tracy - профайлинг FPS / CPU

Апдейт кадра в детализации



Tracy - профайлинг FPS / CPU

Статистика в разрезе N кадров



The screenshot displays the Visual Studio Task Scheduler, showing the execution of various threads and tasks. Red arrows are used to highlight specific components:

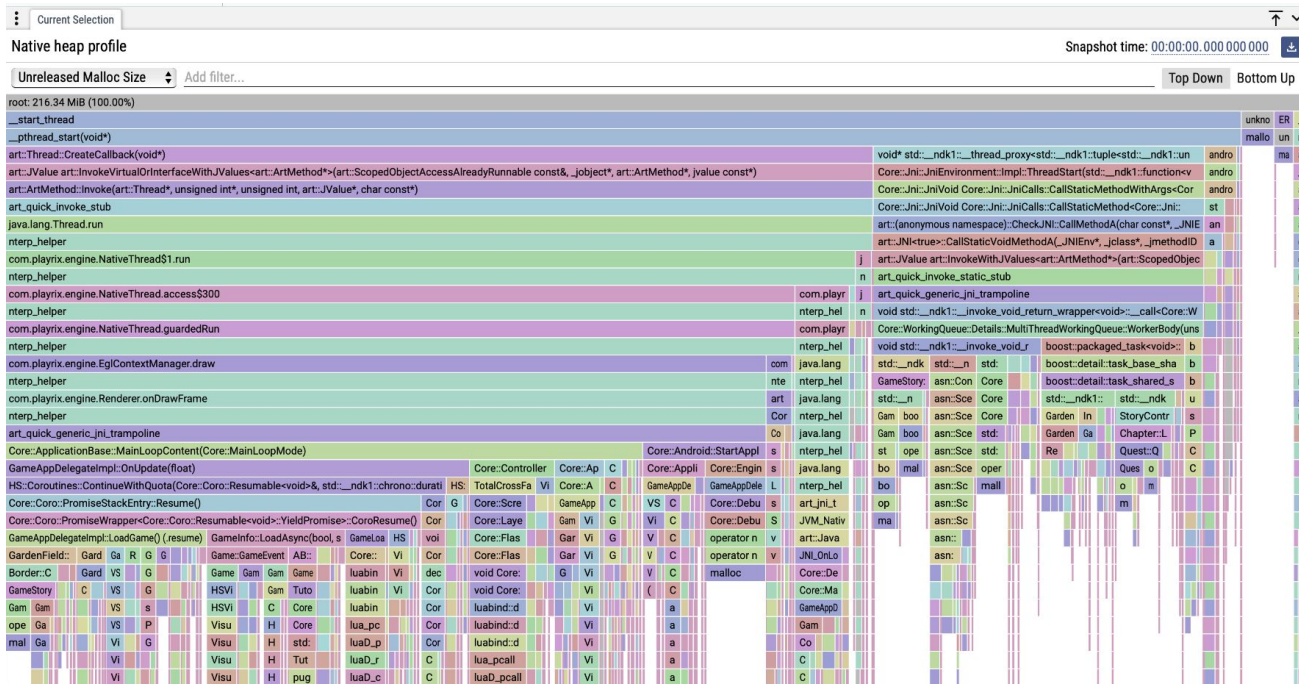
- threads:** Points to the 'WorkingQueue:0' thread, which is currently in a 'locked mutex' state.
- locked mutex:** Points to the 'locked mutex' label, indicating a synchronization point where the thread is waiting for the mutex to be released.
- WorkingQueue:0:** Points to the 'WorkingQueue:0' thread, which is currently in a 'locked mutex' state.

Other visible threads and tasks include:

- OnUpdate Load coros work:** The main thread, currently in a 'locked mutex' state.
- BackgroundThread:** A thread that is currently in a 'locked mutex' state.
- FMOD Studio ban:** A thread that is currently in a 'locked mutex' state.
- WorkingQueue:3:** A thread that is currently in a 'locked mutex' state.
- WorkingQueue:1:** A thread that is currently in a 'locked mutex' state.
- WorkingQueue:2:** A thread that is currently in a 'locked mutex' state.
- Load Resources@Main:** A thread that is currently in a 'locked mutex' state.
- Wait for text and resources StartAsn Start:** A thread that is currently in a 'locked mutex' state.

Perfetto - анализ памяти

- Обзор что загружено и откуда
- Утечки через сравнение дампов



Кастомный профайлер памяти o2

- Поиск утечек через механизм GC
- Где по стеку загружен объект

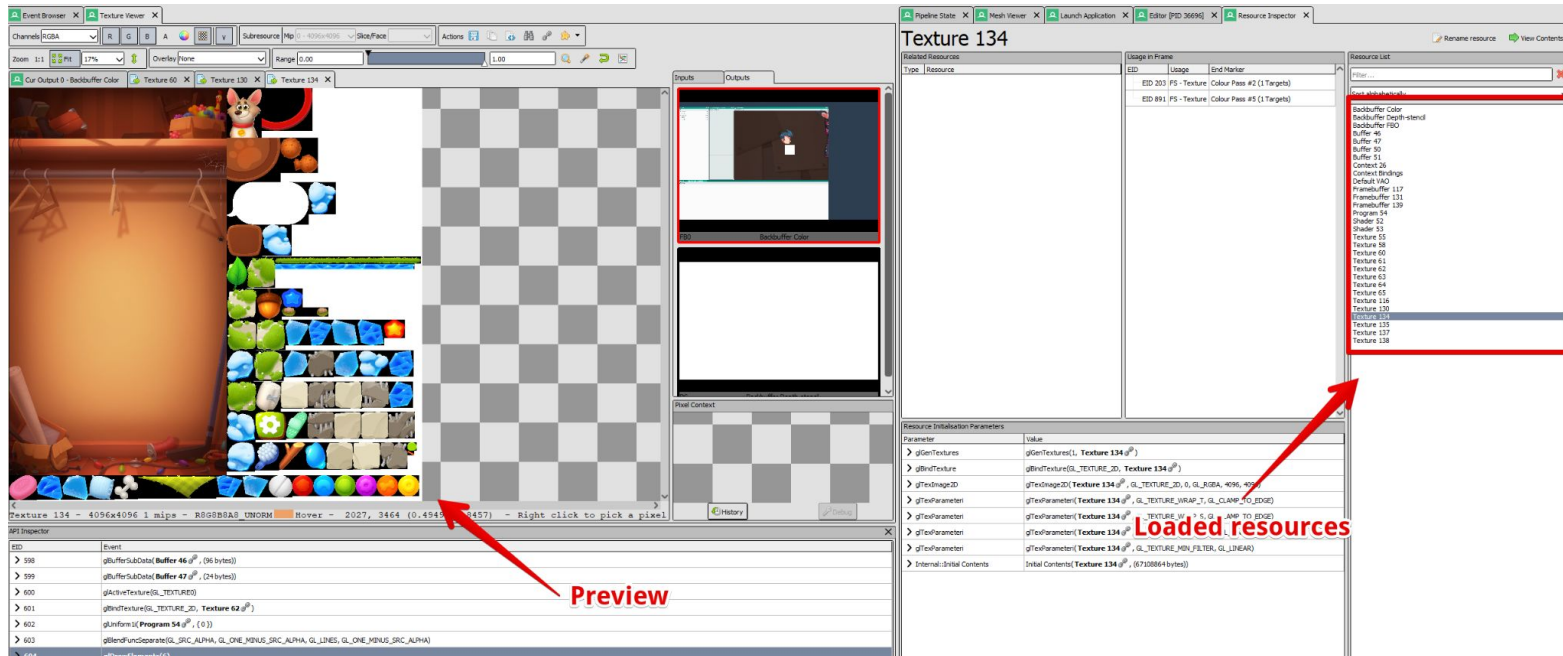
The screenshot displays the 'Memory analyzer' interface. The left pane shows a tree of memory allocations, including categories like 'o2::Debug', 'o2::FileSystem', 'o2::ActorRefResolver', and 'Editor::EditorApplication'. The right pane provides a detailed view of a selected object, 'o2::WidgetState mVisibleState', showing its stack trace and properties.

Object	Address	Size
o2::Debug	000001CD21654950	3.70 Kb
o2::FileSystem	000001CD19920FA0	256 b
o2::ActorRefResolver	000001CD21617780	880 b
Editor::EditorApplication	000001CD314FB890	38.05 Mb
o2::Assets	000001CD3128C5A0	10.81 Mb
o2::EventSystem	000001CD216BD3B0	40.08 Kb
o2::Input	000001CD216BC830	984 b
o2::LogStream mLog	000001CD314353F0	168 b
o2::PhysicsWorld	000001CD31B6D030	100.91 Kb
o2::ProjectConfig	000001CD317C13F0	144 b
o2::Render	000001CD328E76C0	85.59 Kb
o2::Scene	000001CD320B7AC0	114.71 Kb
o2::TaskManager	000001CD218DBA60	120 b
o2::Time	000001CD30B89BC0	64 b
o2::UIManager	000001CD31A01C70	33.23 Kb
o2::ScriptEngine	000001CD31A3AD40	256 b
o2::CursorAreaEventListenersLayer mMainListenersLayer	000001CD3257C650	120.68 Kb
o2::Vector<o2::Ref<o2::Action>>	000001CD60248BB0	29.16 Kb
o2::Sprite mBackground	000001CD329230B0	2.34 Kb
o2::Sprite mBackSign	000001CD32918C70	2.34 Kb
Editor::UIRoot	000001CD218DEE20	4.72 Kb
Editor::WindowsManager	000001CD3EC7F160	13.41 Mb
Editor::EditorConfig	000001CD32F03F00	3.53 Kb
Editor::ToolsPanel	000001CD45C48200	344.61 Kb
Editor::MenuPanel	000001CD45DF5830	444.30 Kb
Editor::Properties	000001CD41CE0B80	12.53 Mb
o2::SceneAsset	000001CD479E7EE0	520 b
Editor::PropertiesListDlg	000001CD427AFA00	2.06 Mb
Editor::MemoryAnalyzerWindow	000001CD3FAAD190	15.64 Mb
Possible leaks	0000000000000000	1.42 Mb
o2::Vector<o2::Ref<o2::WidgetLayer>>	000001CD32EF8820	344 b
o2::Vector<o2::Ref<o2::WidgetLayer>>	000001CD32EFBF20	344 b
o2::Vector<o2::Ref<o2::Actor>>	000001CD41DB1460	152 b
o2::Vector<o2::Ref<o2::WidgetLayer>>	000001CD419980A0	2.02 Kb
o2::Vector<o2::Ref<o2::WidgetState>>	000001CD420D5E70	6.24 Kb

The right pane shows the stack trace for the selected object 'o2::WidgetState mVisibleState' (Address: 000001CD47781C00, Size: 2.20 Kb). The stack trace includes frames from 'boost::stacktrace::basic_stacktrace' to 'RtlUserThreadStart in ntdll'. Below the stack trace, the 'Properties' section shows the object's name as 'visible' and its state as 'Off State Animation Speed' with a value of '1'. The 'Animation' section shows the object's meta-information as 'o2::DefaultAssetMeta-o2::AnimationAsset'.

RenderDoc - анализ загруженных ресурсов

- Проверка форматов сжатия
- Нет ли лишнего?



RenderDoc - анализ drawcall

- Поиск причин разрывов drawcall'ов
 - Смена шейдера, материала и тп

The screenshot displays the RenderDoc application interface, used for analyzing graphics driver events. The left pane shows a list of drawcalls, with a red vertical line and an arrow pointing to the 'Events queue' section. The right pane shows a table of drawcall data, including vertex positions and shader outputs, and a 3D visualization of a red triangle on a checkerboard background.

Events queue

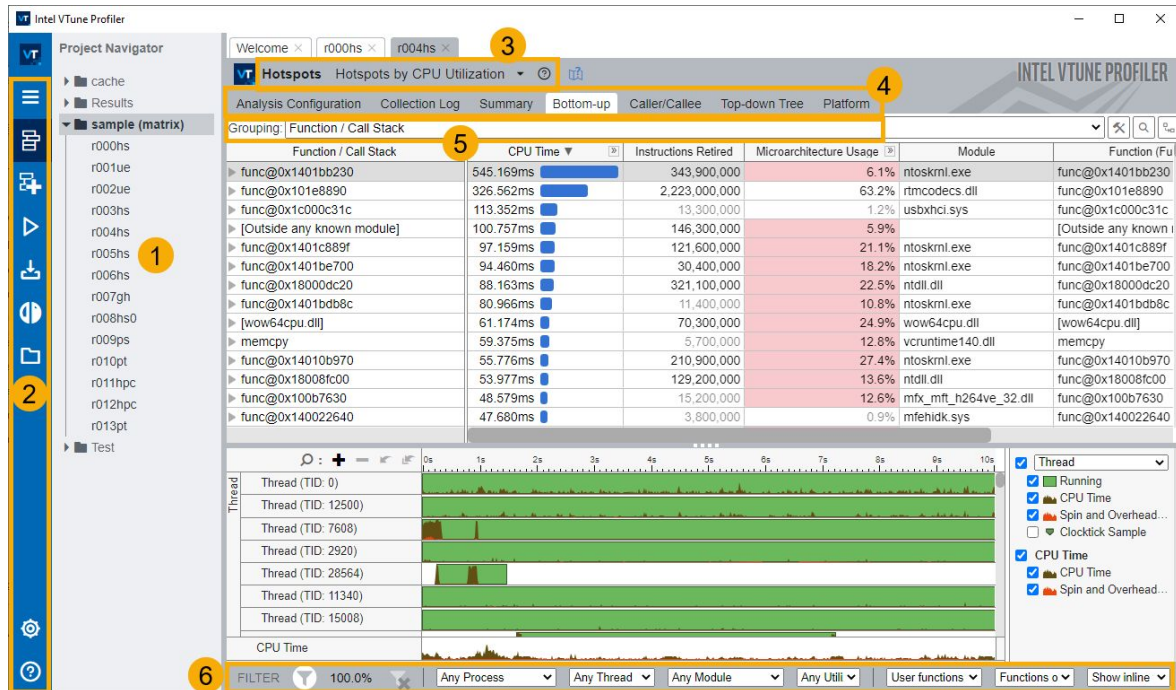
ID	Name
480	glDrawElements(0)
487	glDrawElements(0)
494	glDrawElements(0)
501	glDrawElements(0)
508	glDrawElements(0)
514	glClearColor(0.882353, 0.909804, 0.909804, 1.000000), Depth = <1.000000>, Stand = <0.00>
523	glDrawElements(0)
524-874	glDrawElements(0)
538	glDrawElements(0)
545	glDrawElements(0)
553	glDrawElements(0)
560	glDrawElements(0)
567	glDrawElements(0)
574	glDrawElements(0)
583	glDrawElements(0)
590	glDrawElements(0)
597	glDrawElements(0)
604	glDrawElements(0)
611	glDrawElements(0)
618	glDrawElements(0)
623	glDrawElements(0)
632	glDrawElements(0)
639	glDrawElements(0)
646	glDrawElements(0)
653	glDrawElements(0)
661	glDrawElements(0)
672	glDrawElements(0)
681	glDrawElements(0)
689	glDrawElements(0)
697	glDrawElements(0)
704	glDrawElements(0)
711	glDrawElements(0)

Events queue

ID	Event
619	glVertexAttribPointer(46, 3, GL_FLOAT, GL_FALSE, 12, 0)
620	glVertexAttribPointer(47, 3, GL_FLOAT, GL_FALSE, 12, 0)
621	glActiveTexture(GL_TEXTURE0)
622	glBindTexture(GL_TEXTURE0, Texture 58)
623	glUniform1i(Program 54, 0, 1)
624	glBlendFuncSeparate(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA, GL_ONE, GL_ONE_MINUS_SRC_ALPHA)
625	glDrawElements(0)

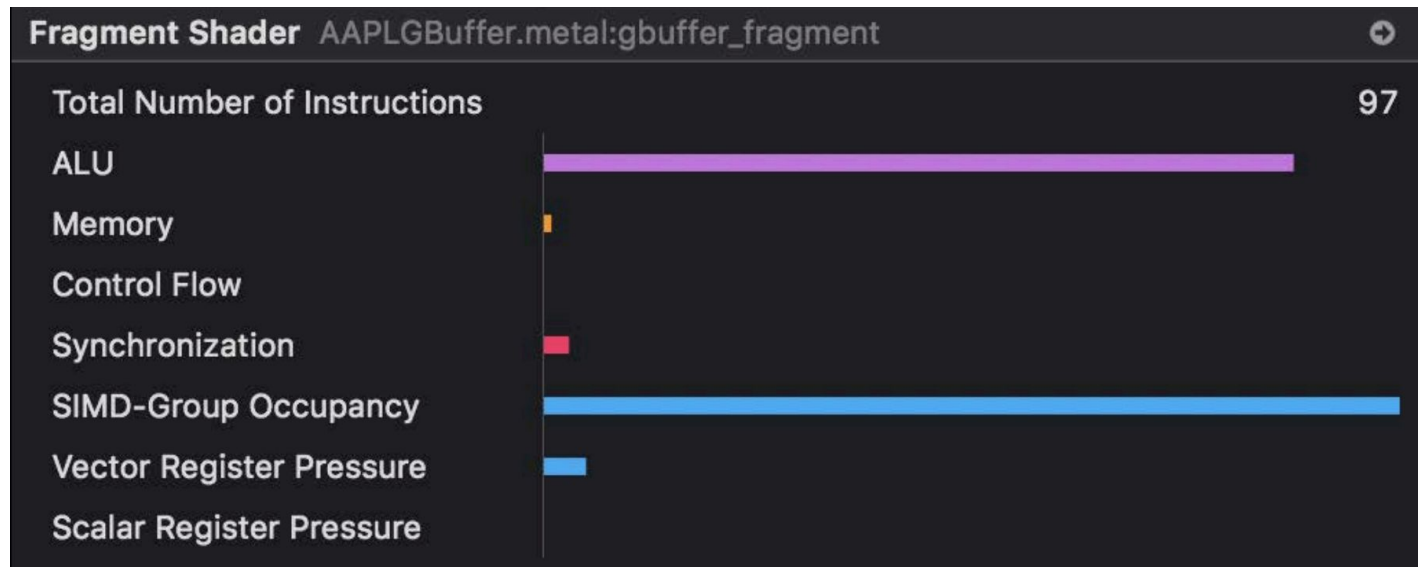
intel vtune

- Близко к железу
 - кеш-утилизация
 - branch prediction
- Когда нужно выжать максимум из железа



GPU vendor-specific

- Производители GPU выпускают свои профайлеры
- Показывают загруженность юнитов, бутлнеки
- Анализ буттлнеков на редких платах со специфичным GPU



PVS

- Держит код “в тонусе”
- Подсвечивает то, что можно упустить на ревью

```
if (chainType == Game::LevelChainType::STORY_CHAIN) {  
    chainId = "";
```

↑ [V815](#) Decreased performance. Consider replacing the expression 'chainId = ""' with 'chainId.clear()'.
}

```
void ResDownloader::DownloadRequiredMatch3LevelResources(int level, int offset, Game::LevelChainType chainType, std::string chainId, std::function<void()> onFinish)
```

↑ [V813](#) Decreased performance. The 'chainId' argument should probably be rendered as a constant reference.
{