redhill
GAMES

30 TIPS AND TRICKS IN 30 MINUTES

OTTO KIVLING, LEAD PROGRAMMER, REDHILL GAMES

Who am I

Currently Lead Programmer at Redhill Games

Worked at Starbreeze Studios, Guerrilla Games, Remedy Entertainment

Also have a background in low level and embedded programming

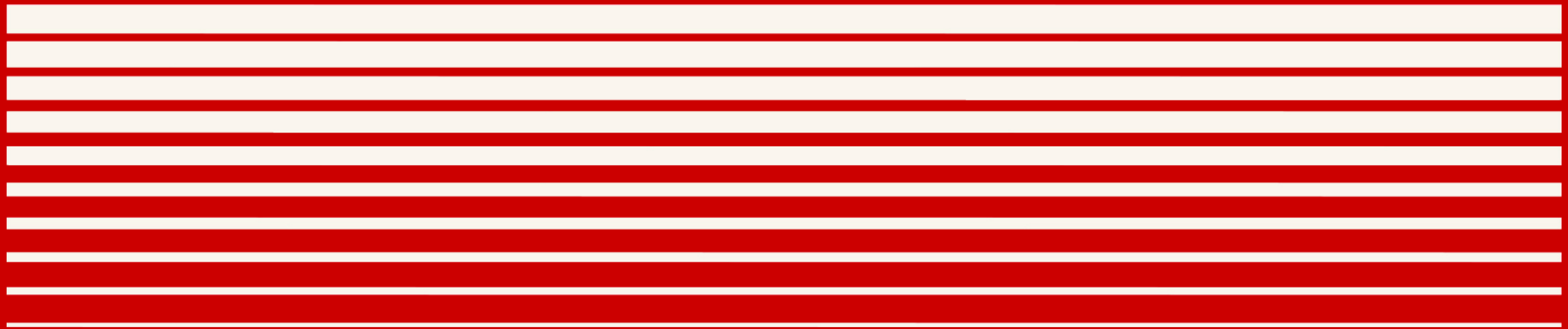
redhill GAMES

Agenda

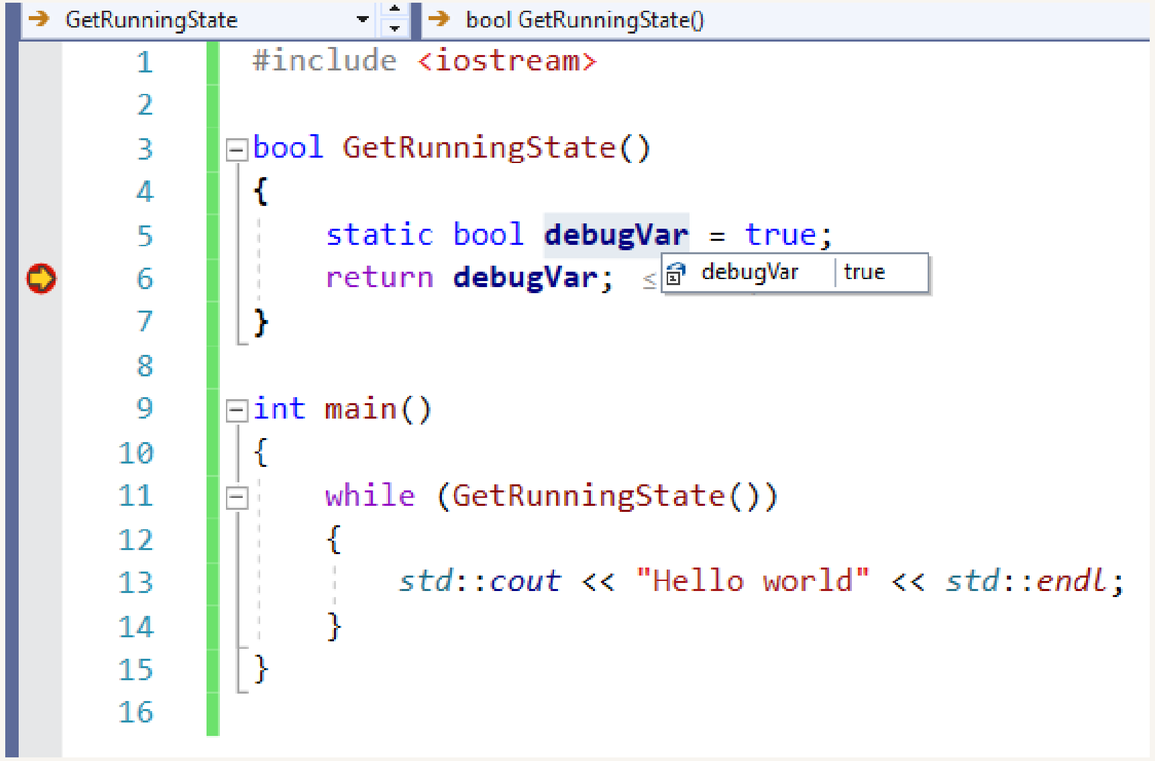
- Debugging
- Performance and memory
- Compiling
- Generic

redhill GAMES

DEBUGGING TRICKS



Static variables when debugging



- Use for debugging or testing
  - Shortcut through logic
- Examples
  - Simulate message received
  - Go straight to high score
  - Button/key presses

redhill GAMES

Recognize magic debug values

- Debugging/crashes
  - You'll recognize if it's heap or stack
    - `0xCC` Stack (Windows)
    - `0xCD` Heap (Windows)
- Different patterns on different platforms
- More extensive list
  - https://en.wikipedia.org/wiki/Magic_number_(programming)#Debug_values

Value	Description
CCCCCCCC	Microsoft's C++ debugging runtime library and many DOS environments to mark uninitialized stack memory.
CDCDCDCD	Microsoft's C/C++ debug malloc() function to mark uninitialized heap memory, usually returned from HeapAlloc()
FDFDFDFD	Microsoft's C/C++ debug malloc() function to mark "no man's land" guard bytes before and after allocated heap
ABABABAB	Microsoft's debug HeapAlloc() to mark "no man's land" guard bytes after allocated heap memory.
BAADF00D	Microsoft's debug HeapAlloc() to mark uninitialized allocated heap memory
DDDDDDDD	Microsoft's C/C++ debug free() function to mark freed heap memory
FEEEFEEE	Microsoft's debug HeapFree() to mark freed heap memory.

Uninitialized variables release compiled

- Global scope gets initialized
  - `globalVariable == 0`
- Local (stack) doesn't get initialized
  - Becomes random when release compiled

```
int globalVariable;
int main()
{
    int variable;
    std::cout << "Hello globalVariable: " << globalVariable << "\n";
    std::cout << "Hello variable: " << variable << "\n";

    return 0;
}
```

```
C:\Users\otto\Documents\Visual Studio 2019\ConsoleApplication1\ConsoleApplication1\Release\ConsoleApplication1.exe
Hello globalVariable: 0
Hello variable: 2004933264
```

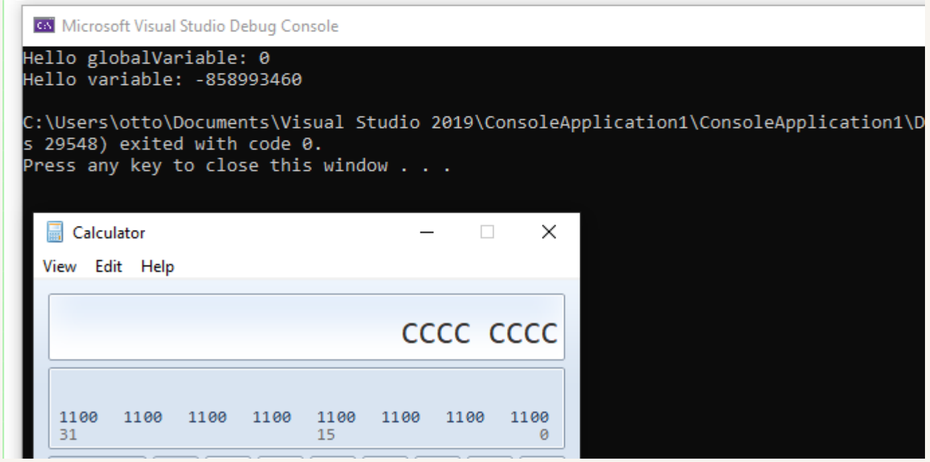

redhill GAMES

Unitialized variables debug compiled

- Remember the memory guardians?
- `variable == -858993460`
  - Which is `0xCCCC CCCC`
- You are looking at uninitialized stack memory

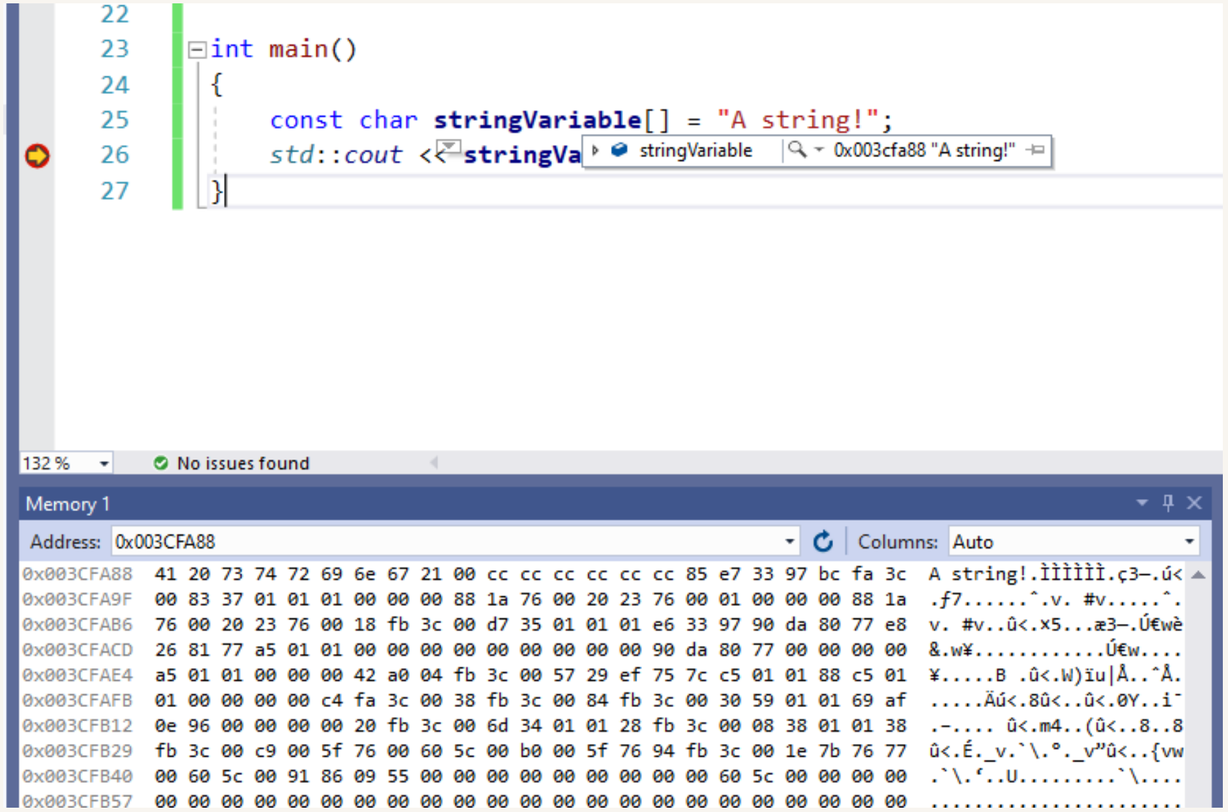
```
int globalVariable;
int main()
{
    int variable;
    std::cout << "Hello globalVariable: " << globalVariable << "\n";
    std::cout << "Hello variable: " << variable << "\n";

    return 0;
}
```



redhill GAMES

Use the memory viewer in VStudio

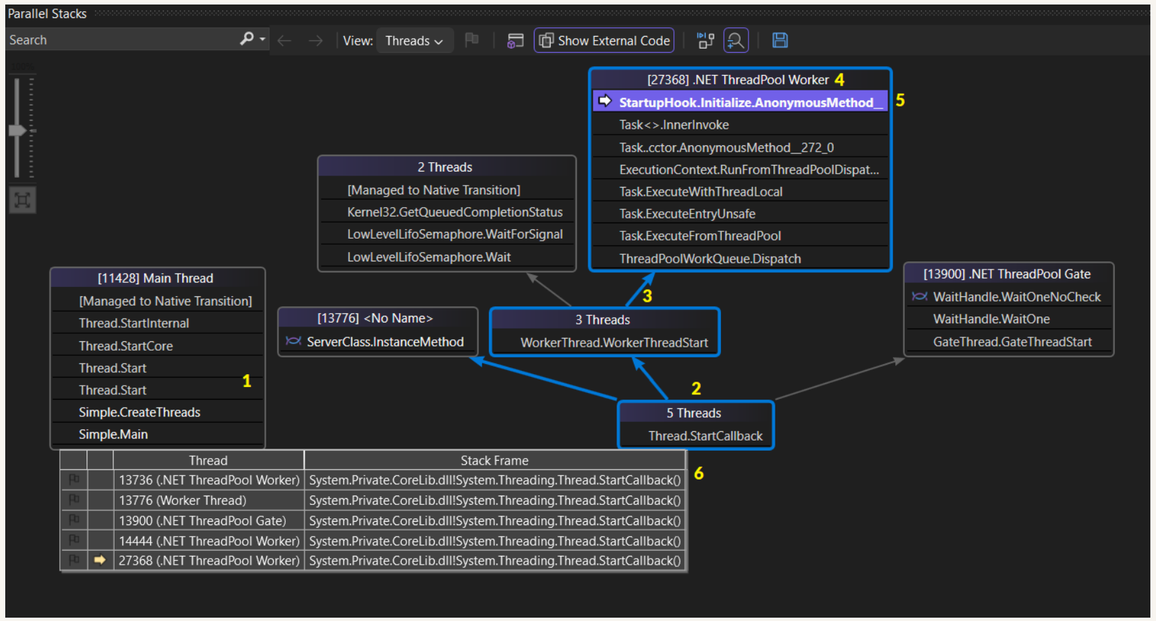


- Can give hints as to what's causing a memory stomp, overflow, etc
- Example below is read from heap(0xCD) to stack (0xCC)

```
0:000> dd esp 00000084`67b9fd30
00000084`67b96cc0  67b96d00 00000084 89194aa8 00007ff7
00000084`67b96cd0  cdcdcdcd cdcdcdcd cccccccc cccccccc
00000084`67b96ce0  cccccccc cccccccc cccccccc cccccccc
00000084`67b96cf0  cccccccc cccccccc cccccccc cccccccc
00000084`67b96d00  67b96d40 00000084 89194978 00007ff7
00000084`67b96d10  cdcdcdcd cdcdcdcd cccccccc cccccccc
00000084`67b96d20  cccccccc cccccccc cccccccc cccccccc
00000084`67b96d30  cccccccc cccccccc cccccccc cccccccc
00000084`67b96d40  67b96e10 00000084 891a942f 00007ff7
00000084`67b96d50  cdcdcdcd cdcdcdcd 67b96dd8 00000084
```


redhill GAMES

Parallel stacks in Visual Studio



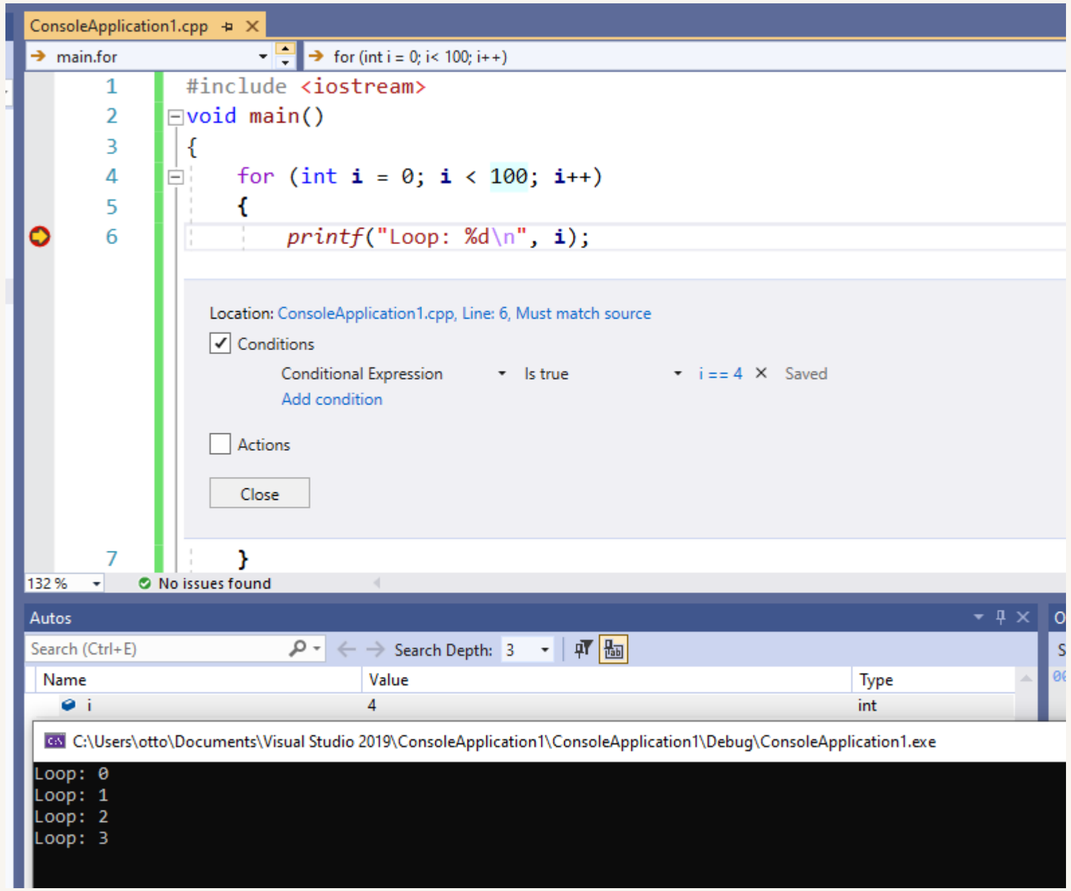
- See what other threads are doing during a breakpoint
  - Can be useful to see what else is getting processed
- Can be a timesaver when debugging deadlocks
  - Could very well see cause of deadlock

redhill GAMES

Conditional breakpoints

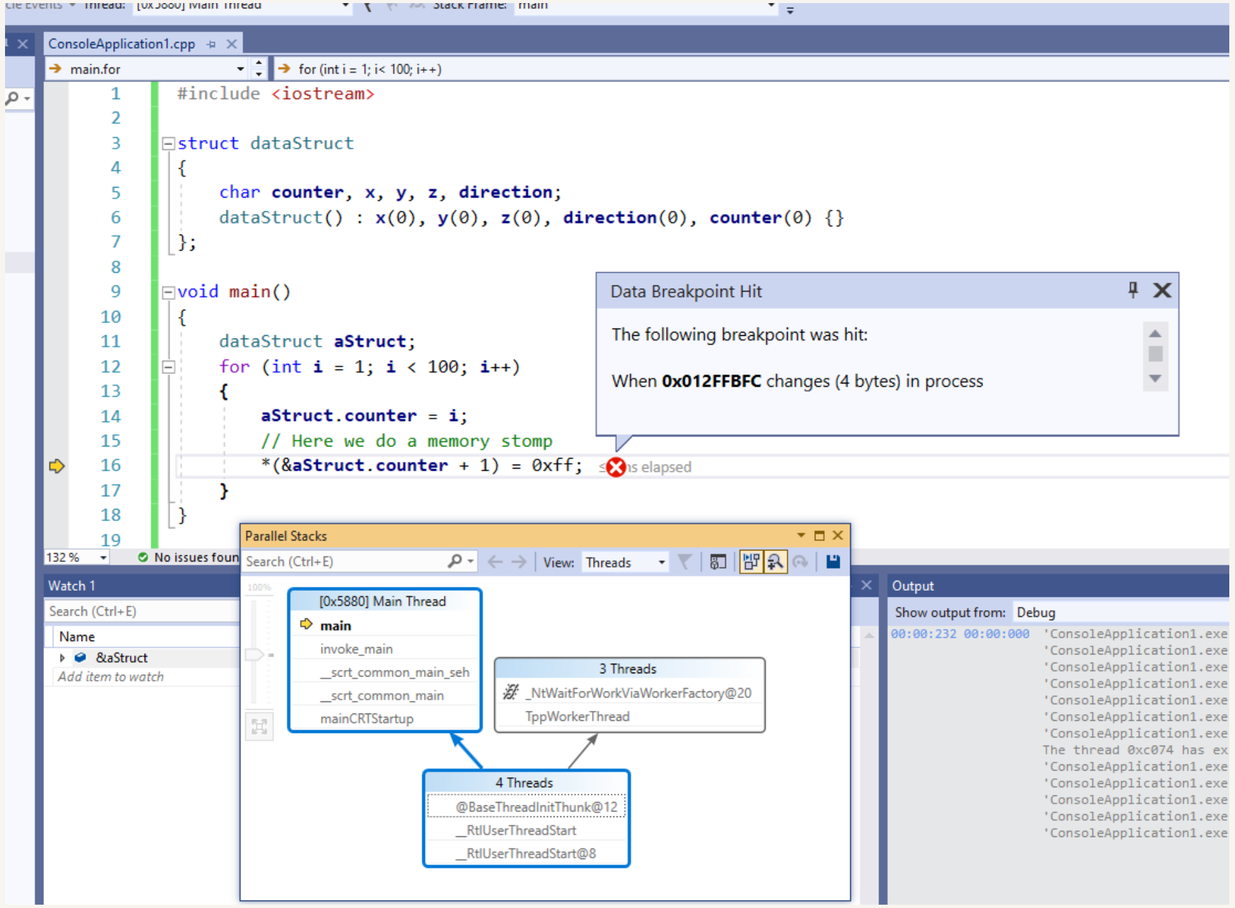
- Stop when a condition is met
  - A file is open
  - A socket is open
  - Iteration reached a certain value
- Poor-man conditional breakpoint
  - Use if-statement and DebugBreak()

```
#include "debugapi.h"
...
if(i == 5) DebugBreak();
```



Data Breakpoints

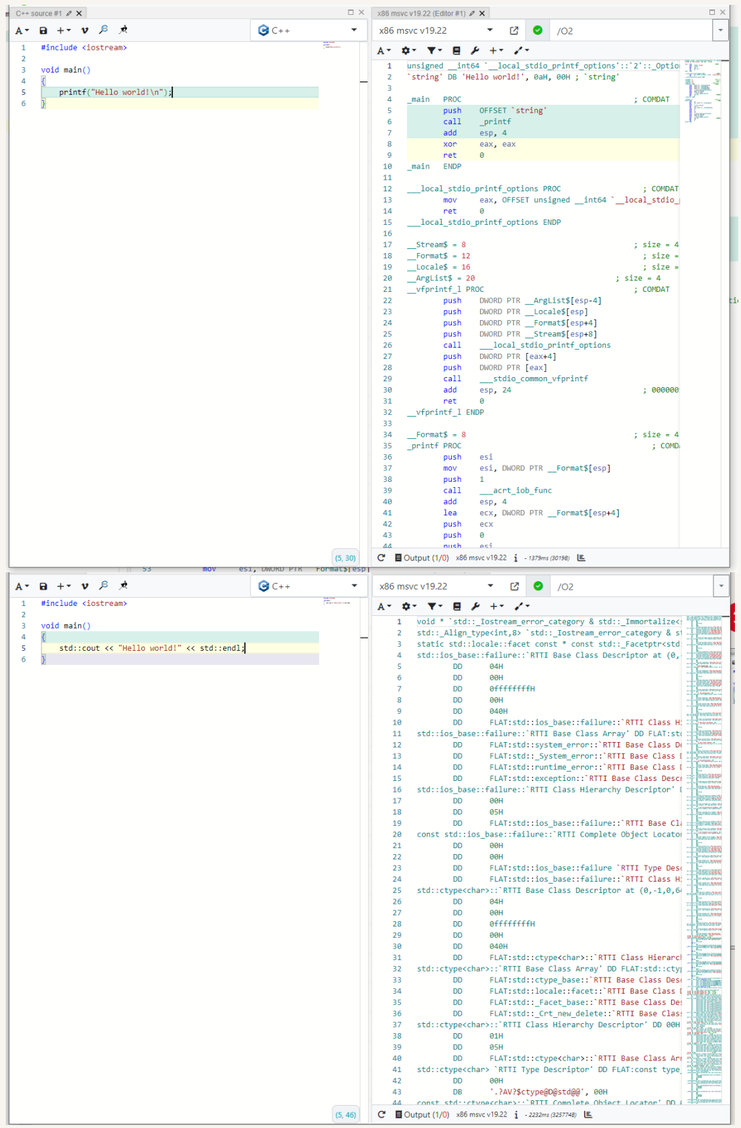
- Good for finding e.g memory stomps
- Must first hit any breakpoint before it can be set
- Will break when X bytes at memory location changes
  - Can use offset if watching a struct
- Memory address change between sessions



redhill GAMES

Check snippet in Compiler Explorer

- Put your snippet in, select compiler, check output
  - https://godbolt.org/
- You can also use disassembly view in VStudio
- Compiler errors on cross-platform code e.g Linux vs Windows
  - ...or verify that it is a compiler bugs
- Visualize bloat in functions
  - printf vs std::cout on length



#pragma optimize("", off)

Parameters of the optimize Pragma

Parameter(s)	Type of optimization
`g`	Enable global optimizations. Deprecated. For more information, see /Og (Global optimizations).
`s` or `t`	Specify short or fast sequences of machine code.
`y`	Generate frame pointers on the program stack.

These parameters are the same letters used with the /O compiler options. For example, the following pragma is equivalent to the `/Os` compiler option:

C++ Copy

```
#pragma optimize( "s", on )
```

Using the `optimize` pragma with the empty string (`""`) is a special form of the directive:

When you use the `off` parameter, it turns all the optimizations, `g`, `s`, `t`, and `y`, off.

When you use the `on` parameter, it resets the optimizations to the ones that you specified using the /O compiler option.

C++ Copy

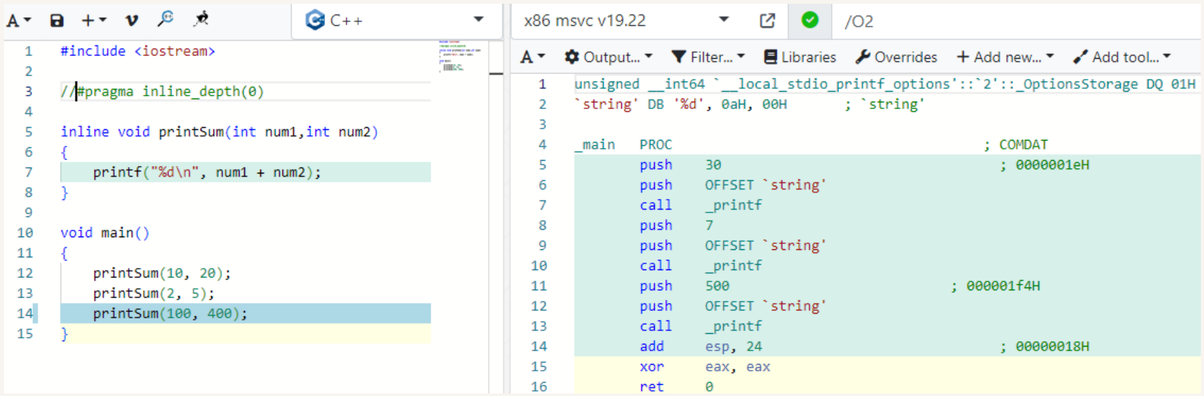
```
#pragma optimize( "", off )
/* unoptimized code section */
#pragma optimize( "", on )
```

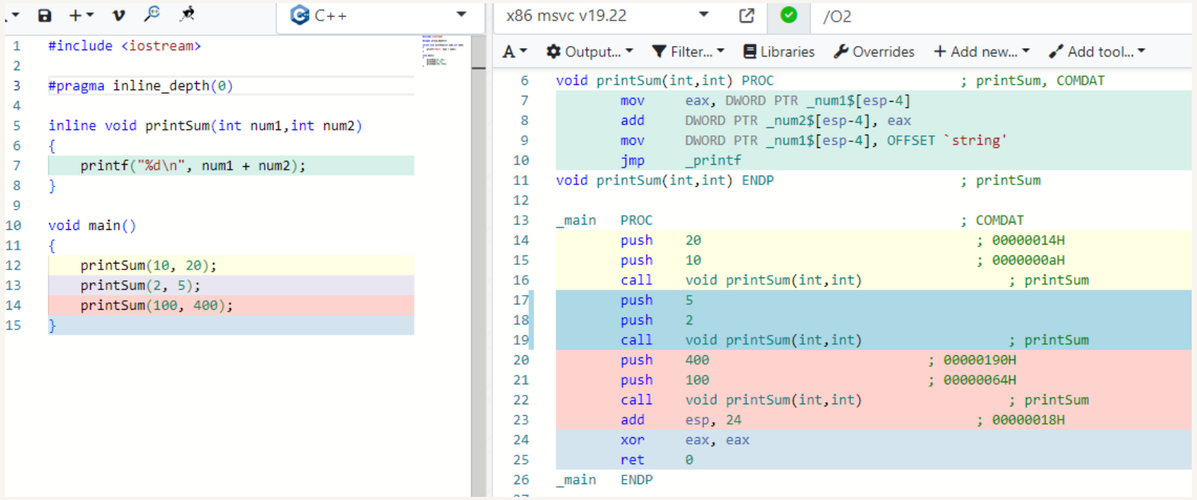
- Makes release/optimized code easier to debug
- #pragma are compiler specific
- Don't forget it in shipping code
  - Grep before shipping!
  - Could be checked for in your build system

redhill GAMES

#pragma inline_depth(0)

- Used when debugging optimized code
  - inlined code can be hard to debug
- Value of 0 inhibits inlining

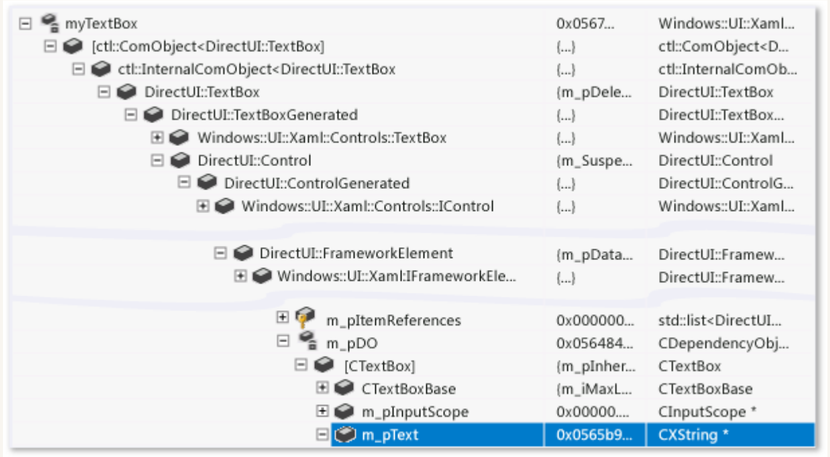


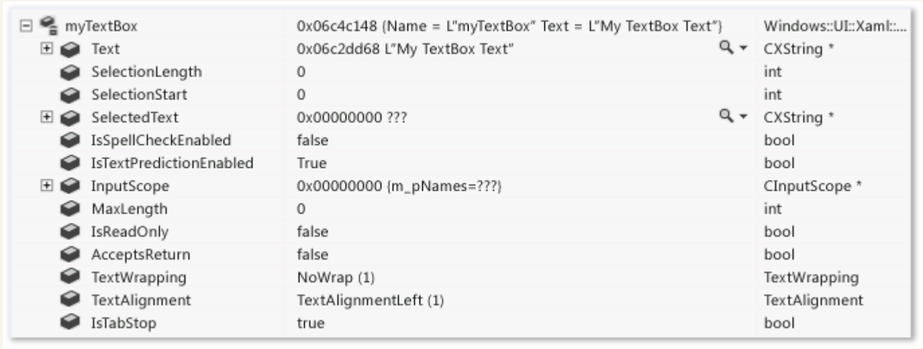


redhill GAMES

Use natvis in Visual Studio

- NATive type VISualization
- Helps with debugging
- Premade exists for major engines
- Make your own for proprietary engine

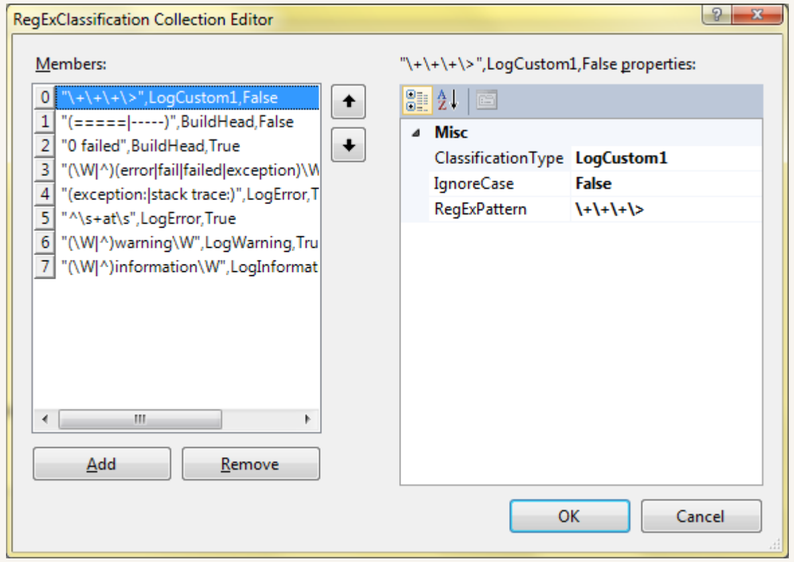


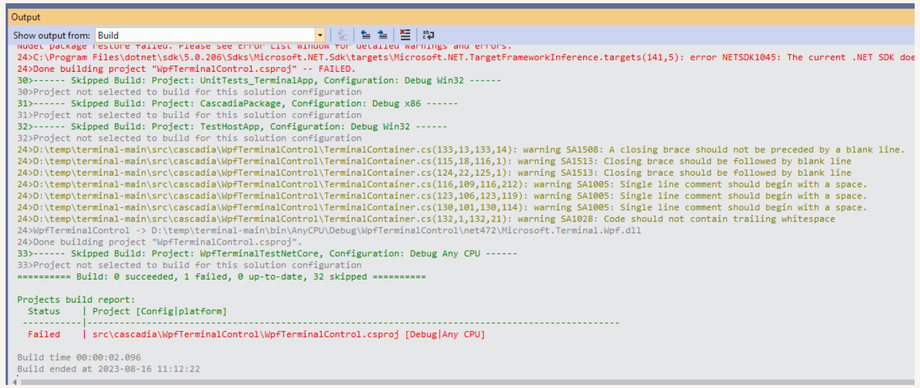


redhill GAMES

VSColorOutput for Visual Studio

- Makes sifting through text much easier
- Create your own regexp for highlight
- Compiler output
- Log output





redhill GAMES

Use Internal speakers

Syntax

C++ Copy

```
BOOL Beep(
  [in] DWORD dwFreq,
  [in] DWORD dwDuration
);
```

Parameters

`[in] dwFreq`

The frequency of the sound, in hertz. This parameter must be in the range 37 through 32,767 (0x25 through 0x7FFF).

`[in] dwDuration`

The duration of the sound, in milliseconds.

Return value

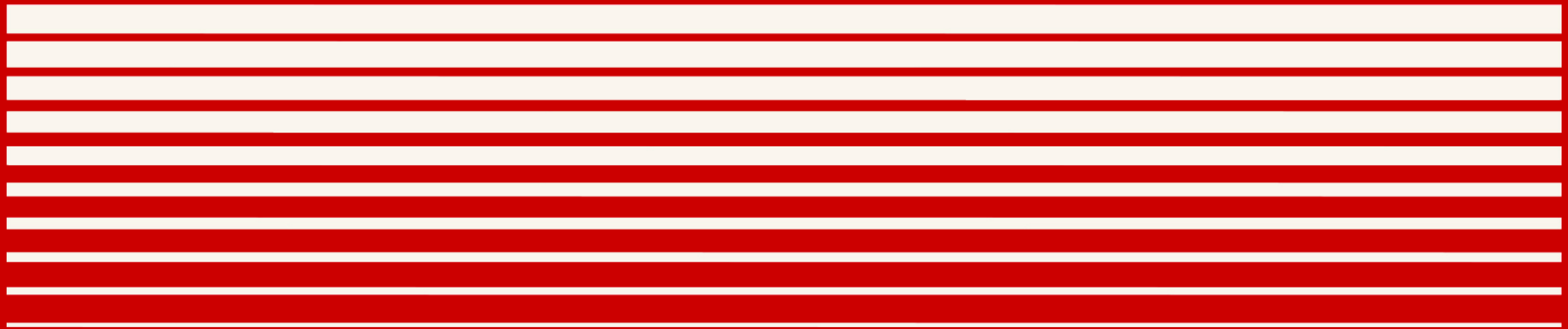
If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information, call GetLastError.

- Used to debug threading issues
  - Listen to the sequence of beeps and hear something out of sync
- One frequency per thread
  - Thread1: Beep(750, 50);
  - Thread2: Beep(850, 50);
  - …
- utilapiset.h
- https://learn.microsoft.com/en-us/windows/win32/api/utilapiset/nf-utilapiset-beep

redhill GAMES

PERFORMANCE AND MEMORY



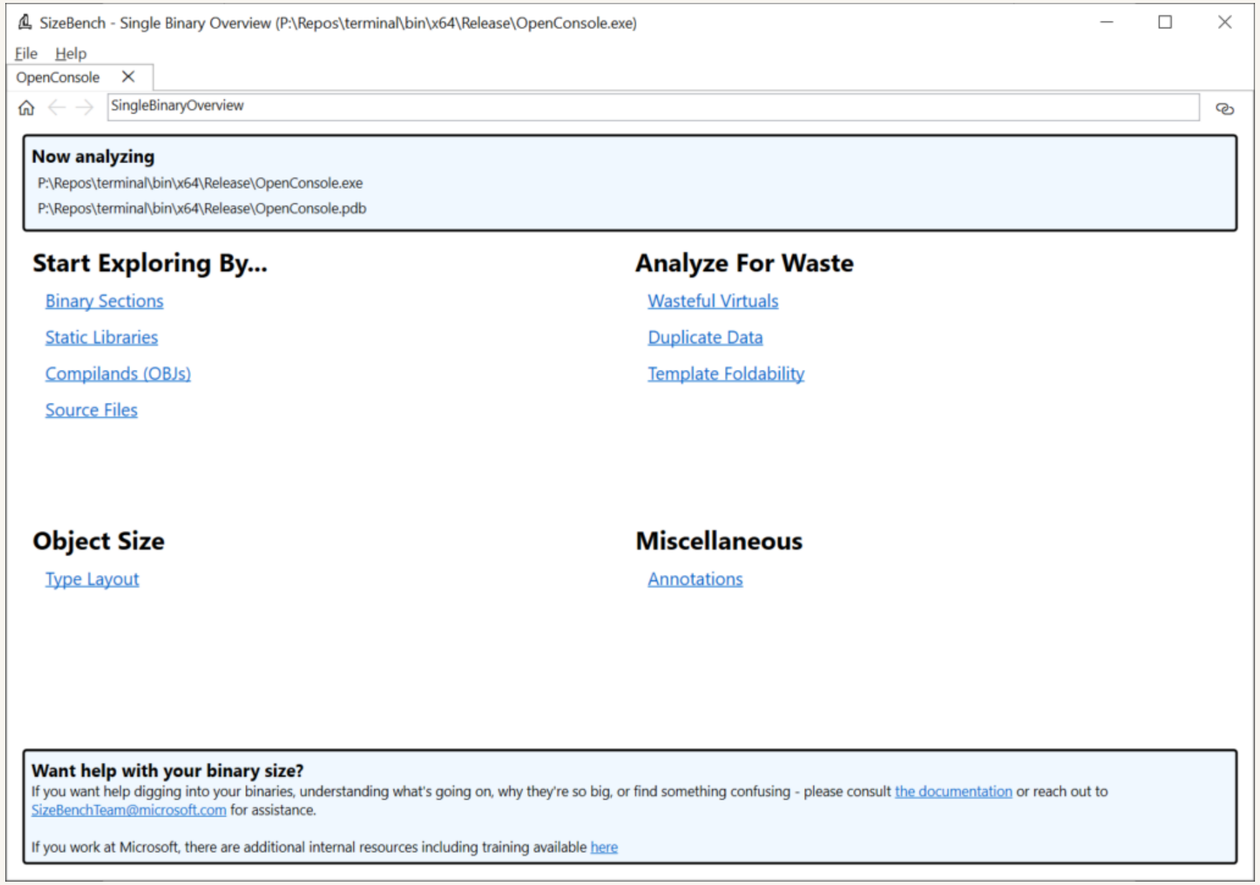
Measure before and after optimization

- Can’t be sure the optimization was an optimization
- Must have a baseline to compare against
  - Compilers and CPUs may behave unexpectantly

redhill GAMES

Optimize binary size for DLL & EXE

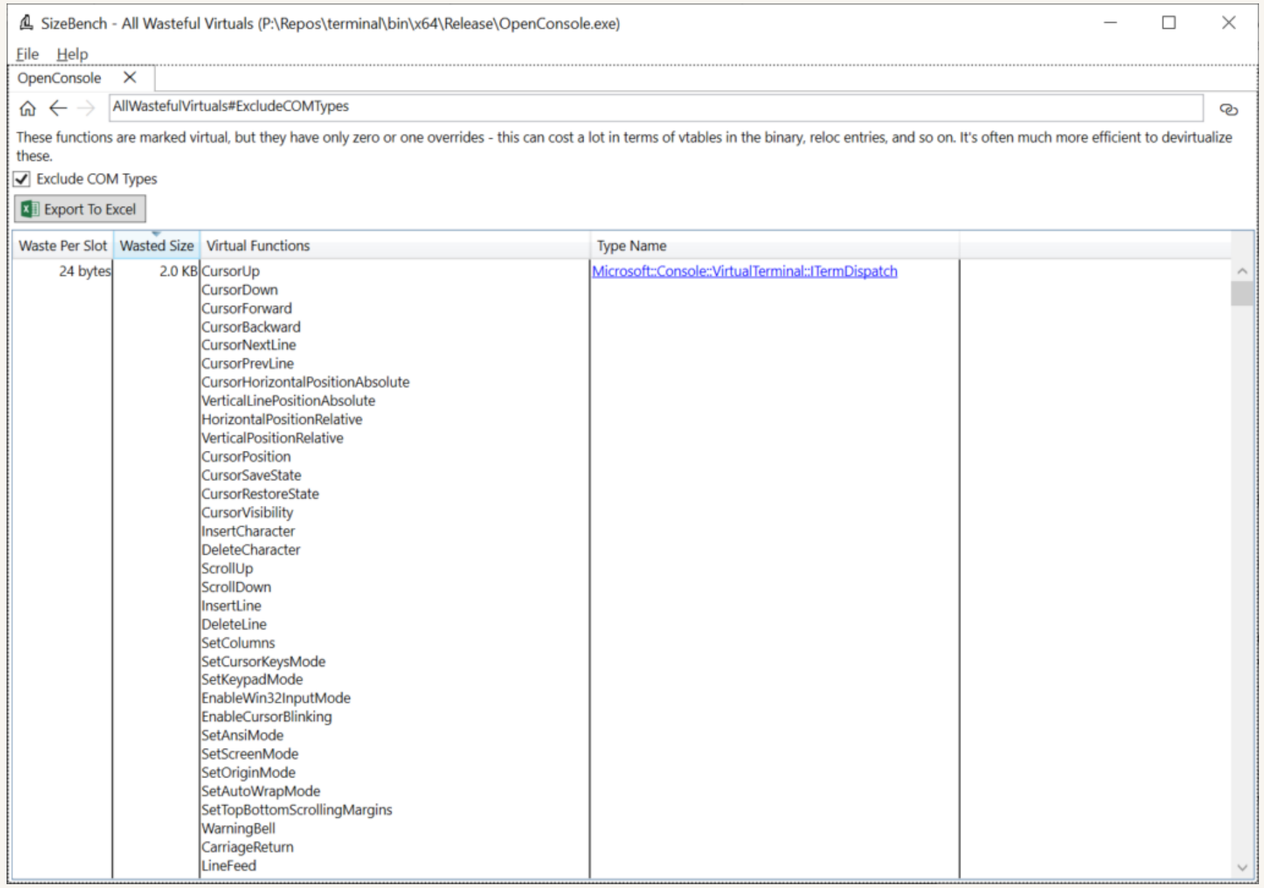
- Smaller binary size equals…
  - Smaller patches/updates
  - Faster loading
  - Less memory when loaded
- Sizebench best tool I've found
  - Check imports
  - Binary sections in executable
  - Template, virtual, padding waste
  - https://aka.ms/SizeBench



redhill GAMES

Wasteful virtuals

- Virtual functions create slots in vtable
- Having a large hierarchy can add up
- Wasteful
  - Virtuals with no overrides
  - Pure virtual with exactly one override
- Fix by removing virtual keyword



redhill GAMES

Padding waste

- Caused by alignment
  - Each field in a type must be aligned in some way
- Is sizeof(SimpleStruct) == 4?

```
struct SimpleStruct
{
    char  x; // 1 byte
    short y; // 2 byte
    char  z; // 1 byte
};
```

- sizeof(SimpleStruct) is 6, waste is 2 bytes

```
struct SimpleStruct
{
    char  x;   // offset: 0, alignment 1 byte
    [padding]  // offset: 1
    short y;   // offset: 2, alignment 2 bytes
    char  z;   // offset: 4
    [padding]  // offset: 5, alignment 1 byte
};
// sizeof(SimpleStruct) == 6
```

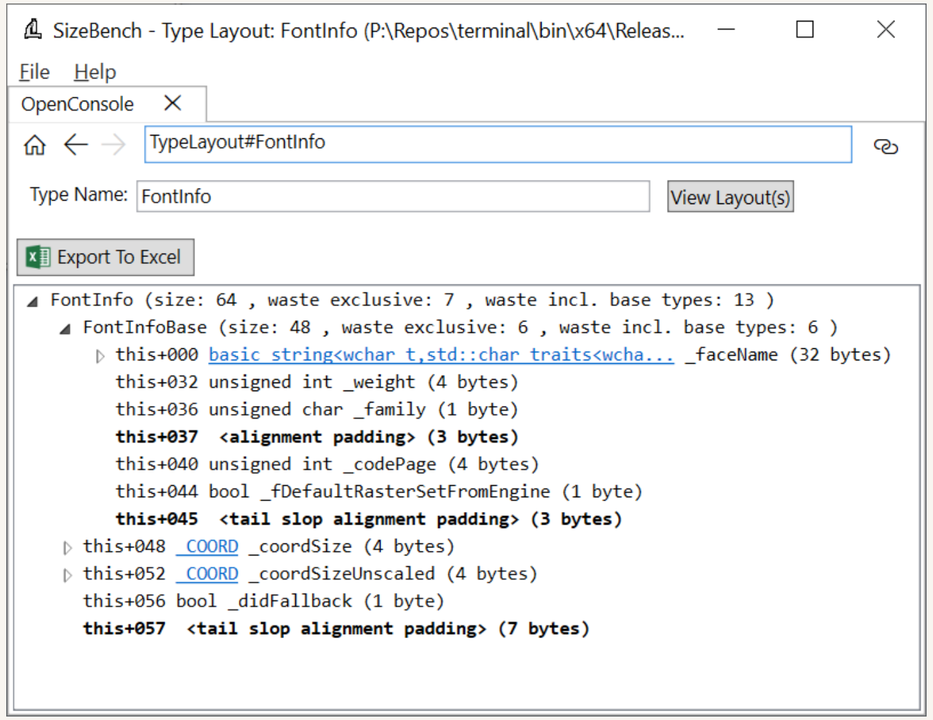
Padding waste cont'd

- Simple fix – restructure your structure

```
struct SimpleStruct
{
    short y;  // offset: 0
    char  x;  // offset: 2
    char  z;  // offset: 3
};
// Now sizeof(SimpleStruct) == 4
```

- Fix by rearranging data and using bitfields
  - Note that bitfields get padding if underlying types don't match

```
struct bitfieldStruct
{
    char x :           2; // offset: 0
    [padding] :       30; // offset: 2 bits
    unsigned int y : 30; // offset: 2 bytes
    [padding] :        2; // offset: 2 bytes + 30 bits
};
```



redhill GAMES

Unrolling loops when it makes sense

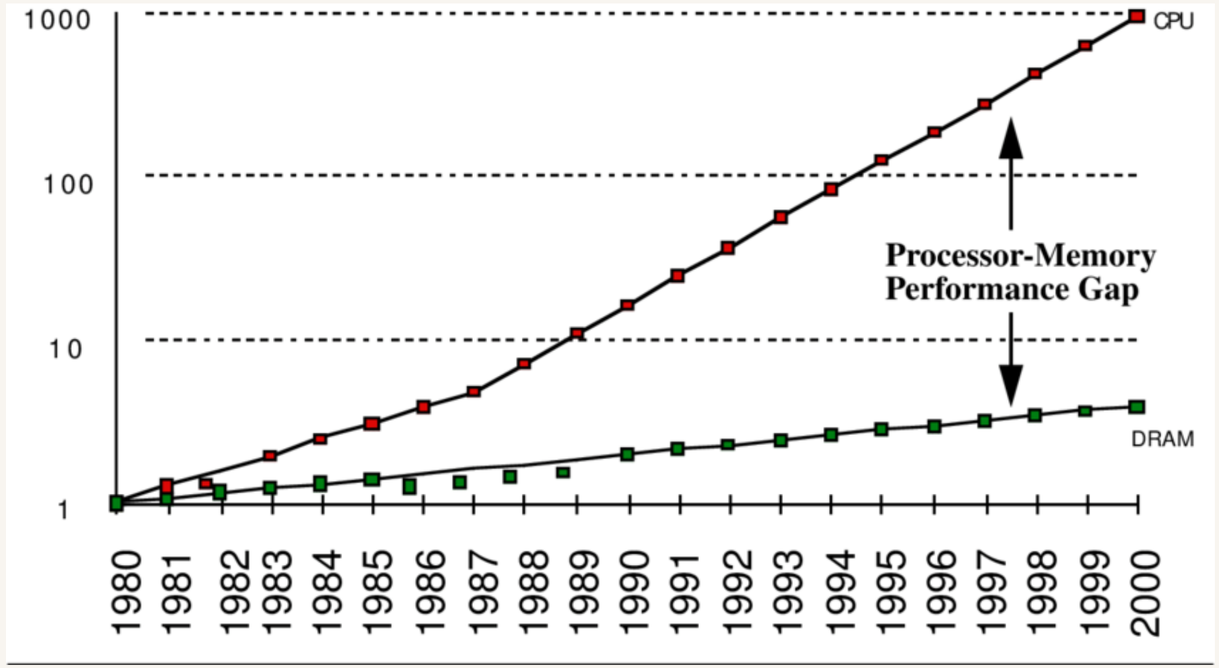
- It can give gains on stalls
  - ...but can also get worse
- Compilers will help unroll
  - Human needed for more complex cases
- Compiler+CPU won't behave as you expect
  - You must benchmark before and after

```
for (int i = 0; i < maxLoop - 3; i += 4)
{
    bigSum += data[i + 0];
    bigSum += data[i + 1];
    bigSum += data[i + 2];
    bigSum += data[i + 3];
}
```


redhill GAMES

Lookup tables can lose performance

- What is it?
  - Use pre-calculated values
    - F.e trigonometric values
  - Quite legacy/old-school
- CPU + memory gap
  - CPUs much faster than memory lookup



redhill GAMES

Check your compiler flags

- Set in the beginning of a project
  - ...then forgotten about
- Can give you “free” performance boost
- Check the documentation for your compiler/platform
- Do this late in a project when needing that extra gain

redhill GAMES

Evict log text from memory

- Take stock of e.g log windows
- Serialize to disk, prone amount of text
- Had a C# tool that kept all text
  - It did eat up 30GB+ memory when running for a long time

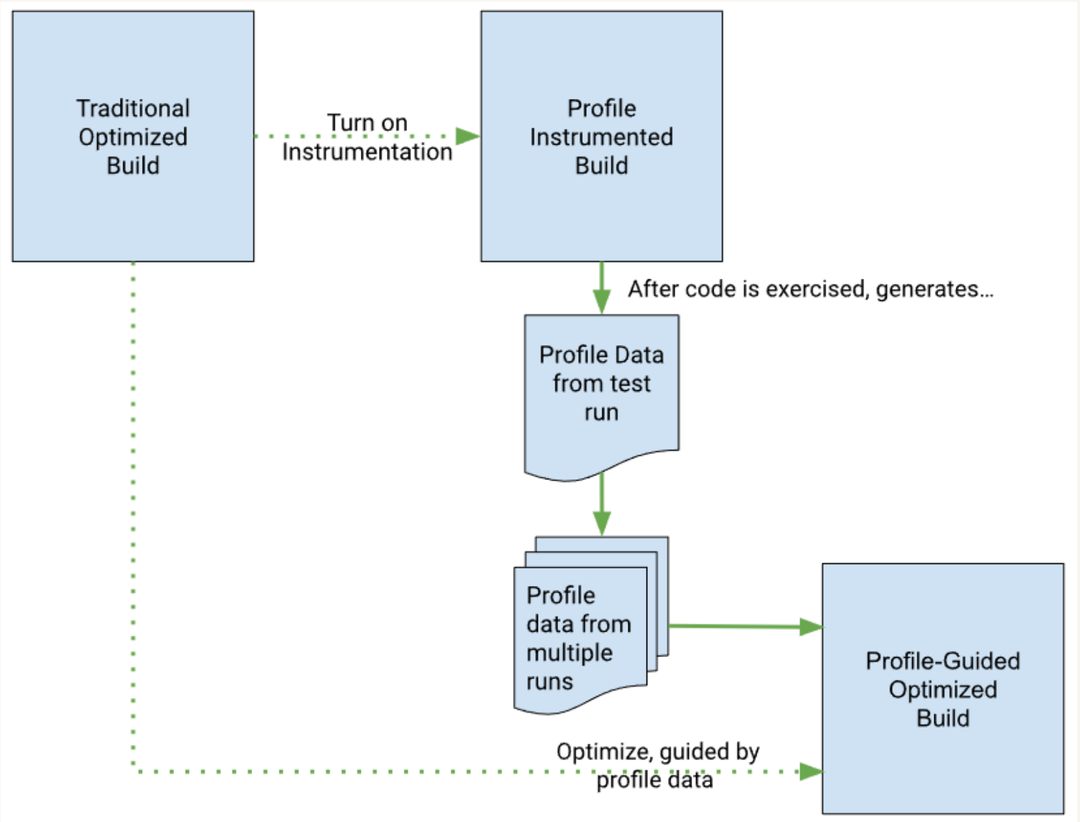
“Hidden” memory allocation for shipping

- Beginning of a project create a huge allocation
- Don’t touch it during production
- Last thing before cert – release allocation
  - Can really be a crunchsaver

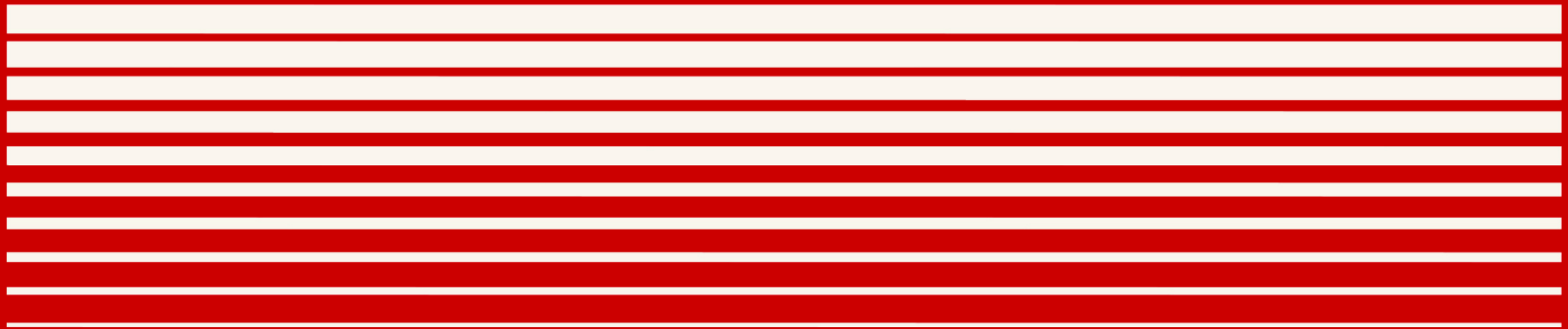
redhill GAMES

PGO – Profile Guided Optimization

- Better performance, but..
  - ...extra build step
- Gets real data from how the game runs
  - Generates file, feed back to linker
- Can yield 5% - 10% perf boost



COMPILING



Forward declare for faster compiles

- Reduce the number of files opened by #include
- Reducing number of the pre-processed files
  - as the header is not included
- Reducing recompilation impact…
  - …when the forward declared class is modified.

```
class ForwardDeclare;
....

class ForwardDeclare
{

};
```


redhill GAMES

Use unity build for faster compilation

- Combines several source files to one bigger
  - Fewer number of header files for parsing and compiling
- Easiest implementation is to #include cpp files
  - Other implementations can write and create new (huge) cpp files
- Potential issues
  - Memory can be an issue
  - Symbol clashes going from local to global scopes

When using unity builds, disable them occasionally

- If not checked regularly...
  - ...can yield quite weird compiling and linking errors
  - If compiling cleanly with Unity-build turned off, it wont break with it turned on

- Disable and start a compilation
  - In Unreal `<bUseUnityBuild>false</bUseUnityBuild>`
  - Issues are usually forgotten #includes / forward declares

redhill GAMES

Include-what-you-use

- Clang-based
- Faster compile / link
  - Remove redundant #includes
- Cleaner view of dependencies
  - An #include is a dependency

redhill GAMES

Use static asserts (compile time asserts)

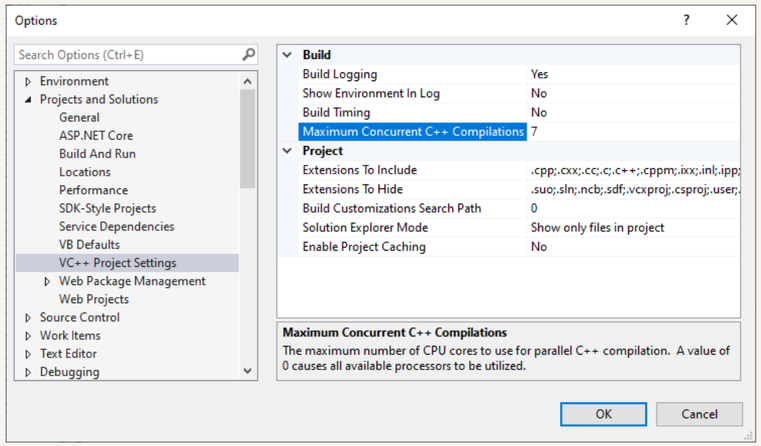
- Takes place after preprocessing
  - When triggered will break compilation
- Size checks for vectors, sizeof(<datatype>), etc.
- Error out on dependencies e.g a certain version of a library

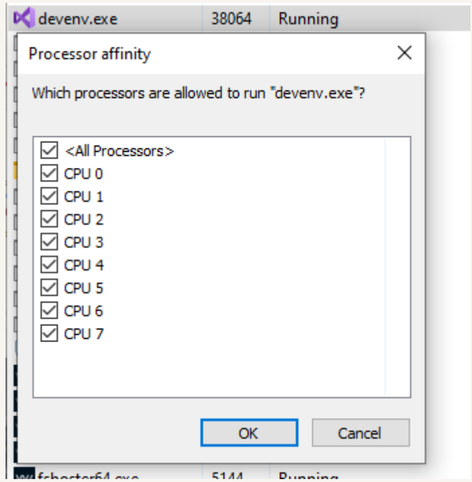
```
static_assert(  THIRD_PARTY_LIB_VERSION == 123,
                "This only works for version 123 of library");
```


redhill GAMES

Limit number of cores during compilation

- In Settings in Visual Studio
  - Num of cores – 1 works for me
- Setting affinity also works
  - Spawned processes inherit affinity
- VS2022 has better support
  - Have not tried it myself





redhill GAMES

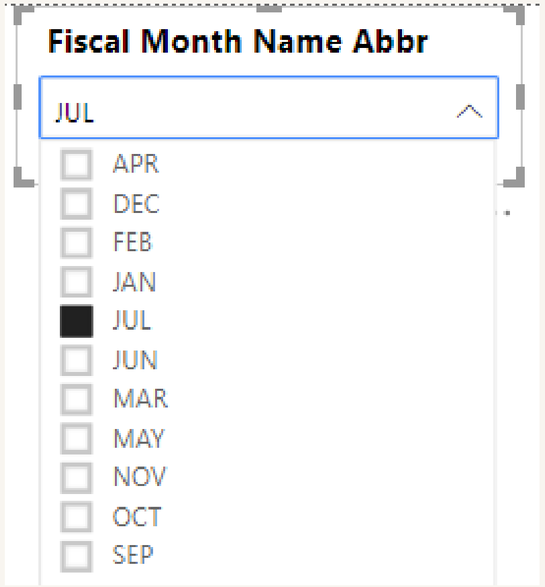
GENERIC TIPS

Be wary of copy/paste

- It’s easy to miss changing index
  - ```
    vector[0] = 0.0f;
    vector[1] = 0.0f;
    vector[1] = 0.0f;
    ```
- Check an extra time
  - If code reviewing, keep an eye on these types of errors
    ```

Sort data – especially when serializing

- Remove random nature of data
- Doing patches
  - Randomized order => big patch
- UI
  - Funky ordering
- Debugging data fields
  - Easier compare



redhill GAMES

Automation

- Use static code analyzers as build task and part of CI
  - After initial setup find new bugs
- Code-complexity charting
  - Cyclomatic complexity, maintainability index, coupling, …
- Submitted bugfix heat mapping in code
  - Find candidates for rewrite
- Chart/check compilation times
  - When reaching a threshold, start optimizing
  - Build Insights now integrated into Visual Studio 2022
    - https://aka.ms/BuildInsightsVS

redhill GAMES

redhill GAMES

THANK YOU FOR LISTENING!
ANY QUESTIONS?

OTTO KIVLING

OTTO@REDHILLGAMES.COM