General Rules

Slow code is a Bug!

- Performance and functionality should be considered equal
- Good architecture is better than late optimizations

Distributed and loose coupling are better.

- Otherwise the component system is a mess
- Naturally distributable systems are easier for multiple developers

Know your Datastructures!

- The stack is your friend :)
- Garbage is not your friend
- Check how much information you have about your data

"... The answer to that is that if you need more than 3 levels of indentation, you're screwed anyway, and should fix your program." (Linux Kernel Guidelines)

Unity Package Manager (UPM)

Create Package

- Create folder and package.json with naming conventions
- Setup inner folder structure
- View version and naming schemes

Add Package Content

- Setup Assembly Definition .asmdef Files
- Create Samples and Documentation
- Add Plugins

Publish and Edit Manifest

- Looking into manifest and possible workflows for package development
- Overview of registries and Git support

Additional Information

Packages: docs.unity3d.com/Manual/Packages.html

Creating Packages: docs.unity3d.com/Manual/CustomPackages.html

UPM Ecosystem: youtube.com/watch?v=22HIEQTyozQ

Package Samples: forum.unity.com/threads/samples-in-packages-manual-setup.623080/

Package Versioning

Major.Minor.Patch(-stage)

Preview

- Until version 1.0.0 the package stays in preview
- Sometimes multiple preview stages (-preview1, ...)
- After preview complition the package is verfied

Verification

- The new package system allows Unit and Integration Tests
- Passing all tests and feature complete

Production Ready

- Breaking Changes?
= Increase major
- Feature added, changed?
= Increase minor
- Fix or Optimized?
= Increase patch

General Advice

Stop using Tags!

Please no Camera.main or similiar!!!

Coroutines are evil!

Stop making everything a component!

Check a event driven architecture!

Keep the Update method slim!

Keep loose coupling in mind!

Problem: Too much Update Calls / Update Calls too slow

Problem

- Update calls are limited to 1000 methods
- Unity looks into each script and adds the Update method to a list
- Calling native C++ code in hot-paths has a high cost

Solution

- Decativate idling Update methods with `base.enable = false;`
- Create own Update loop
- Use event driven architecture

Additional Information

Event Function Order: docs.unity3d.com/Manual/ExecutionOrder.html

1K Update Calls: blogs.unity3d.com/2015/12/23/1k-update-calls/

Problem: UI is slow / UI Rendering takes too long

Problem

- Internally a Canvas generates a mesh with the principle
  "1 UI element changed - change 1 element, dirty the whole canvas"

- UI Layout and Scroll is expensive

Solution

- Use the new UI System ^^

- Create seperate Canvas objects for each logical widget group

- Deactivate Pixel Perfect and clamp the Scroll

- Check for missing fields (see: Graphic Raycaster Source `eventCamera`)

Additional Information

uGUI Source Code: bitbucket.org/Unity-Technologies/ui/src/2019.1/

C# Performance: How does a CPU process data?

SISD

- Single instrution operates on single data element

SIMD

- Single instruction operates on multiple data elements

MISD

- Multiple instructions operate on single data element

MIMD

- Multiple instructions operate on multiple data elements

Additional Information

Mike Flynn: Very High-Speed Computing, 1966

C# Performance: SIMD

System.Numerics.Vector

- Built-in SIMD support for C#
- Does not work well with Unity

System.Runtime.Intrinsics

- Only available for .Net Core 3.0
- Does not work Unity
- Best library for Intrinsics in C#

Unity.Mathematics

- Works together with Burst Compiler to output optimized SIMD code
- Will be the new standard after Mathf
- Similar to shader programming

Additional Information

.Net Core 3 SIMD: link.medium.com/84G4kcrnA1

Unity Mathematics Source: github.com/Unity-Technologies/Unity.Mathematics

C# Performance: MIMD

System.Threading and
System.Collections.Concurrent

- Typical theading approach
- Does not work well with Unity
(because of the main thread limitation)

System.Threading.Tasks

- Easier to write threaded code (like Linq)
- Shares the limitation with Unity

UnityEngine.Jobs

- Limited to Unity
- Requires native containers
- Allows custom scheduling

Additional Information

Unity Job System: docs.unity3d.com/2018.3/Documentation/Manual/JobSystem.html

Unity Job Samples: github.com/stella3d/job-system-cookbook/

C# Performance: Unsafe (Code like in C)

Use more structs!

- Stored on the stack (value types)
- Small structs are a lot faster
- Large structs can be passed via reference
- High density when aligned correctly

Store temporary things on the stack!

- see stackalloc or Span<T> in .Net Core 2.1
- Garbageless arrays

Do cool things with pointers

- Allows pointer arithmetic
- Directly interact with low level components

Additional Information

Memory Layout: kalapos.net/Blog/ShowPost/DotNetConceptOfTheWeek13_DotNetMemoryLayout

Unsafe Specification: docs.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/unsafe-code

Efficent Code: docs.microsoft.com/en-us/dotnet/csharp/write-safe-efficient-code

It's Showtime!

C# Future

youtube.com/watch?v=ZlO1utbB2GQ