

Create a benchmark mobile game!

Tobias Tost

Senior Programmer,
Blue Byte GmbH – A Ubisoft Studio



GAME DEVELOPERS CONFERENCE™ EUROPE
CONGRESS-CENTRUM OST KOELNMESSE · COLOGNE, GERMANY
AUGUST 3-4, 2015

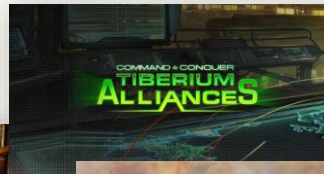


Who am I?

- Tobias Tost, MSc
- In the Games Industry since 2006
- Visualization, Sound, Gameplay, Tools



morpheme
with euphoria



- Joined Ubisoft Blue Byte in Düsseldorf 2013



Agenda

- 1. Project introduction**
2. Engine Selection
3. Content Creation
4. Architecture Overview
5. Game Logic – Game Simulation
6. Unity Editor Extensions
7. C# for Unity
8. Conclusion





Project outline

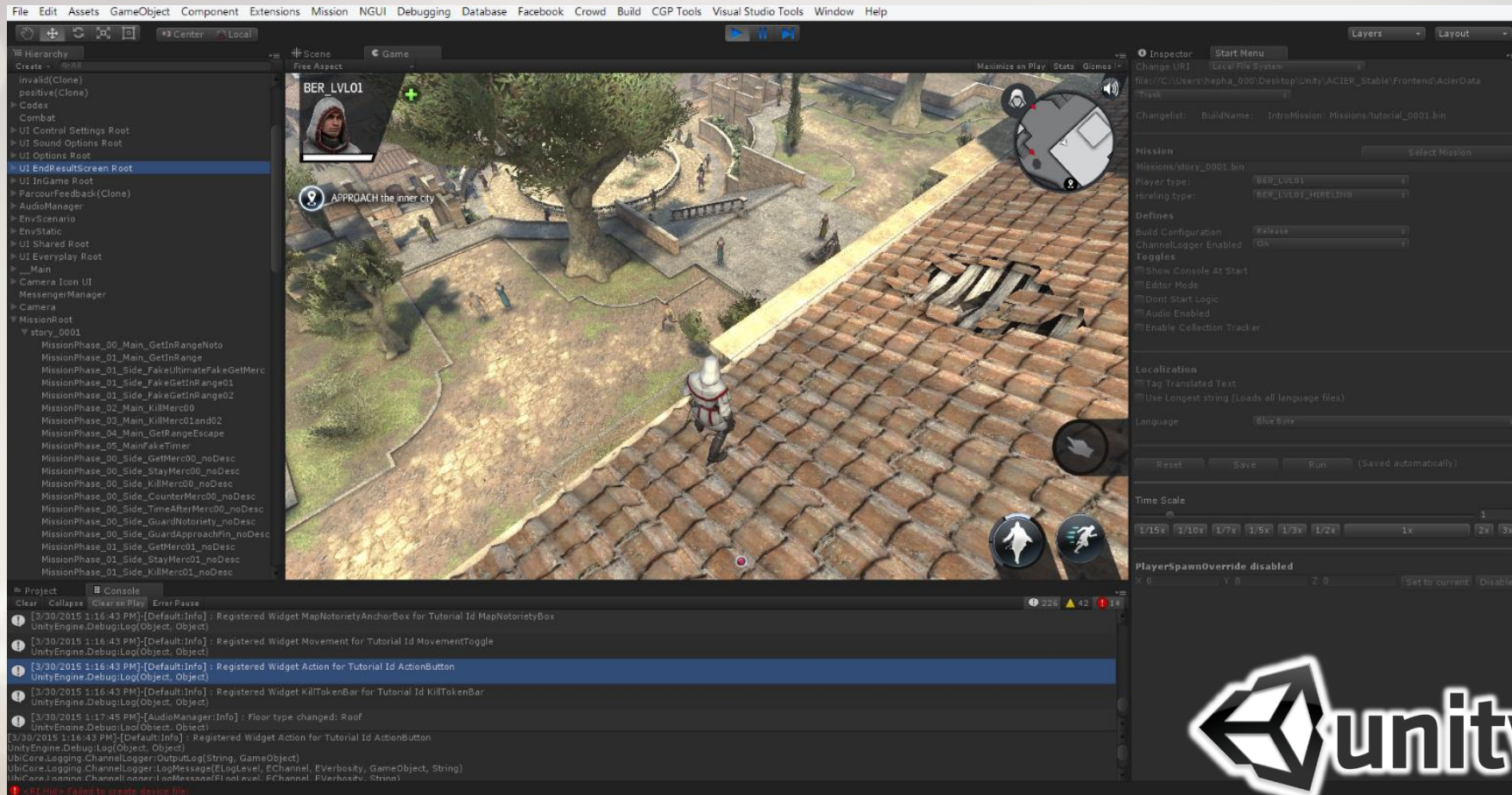
- **Main Goal: delivering high profile AC gameplay Free to Play on mobile platforms**
- In production since September 2013, restricted launch in Australia and New Zealand October 2014
- Considered flagship project of Blue Byte with a noteworthy commitment regarding human resources
- Working with studios in Pune (India), Chengdu (China) and Montreal (Canada)





Engine requirements

- Fast Prototyping support required:
 - Easy to learn environment
 - Support for animation driven gameplay
 - Editor Framework should be easy to understand
- Mobile-friendly
 - Assassins Creeds original Anvil Engine does not target mobile audiences
- Artist-driven with strong focus on extensible Tools
- Flexible licensing terms (e.g. iOS and Android only needed for some engineers)





Agenda

1. Project introduction
2. Engine Selection
- 3. Content Creation**
4. Architecture Overview
5. Game Logic – Game Simulation
6. Unity Editor Extensions
7. C# for Unity
8. Conclusion





Content Pipeline

- Game projects of Assassin's Creed 2 with the in-house game engine "Anvil"
- Anvil's pipeline has 3D Studio Max as Editor
 1. Export Anvil asset to 3DSMax
 2. Load asset in 3DSMax and prepare using our own MAXScripts
 3. Export to FBX
 4. Import FBX to Unity (built-in support)
- Preparing and Polishing Assets in Düsseldorf and Chengdu

ASSASSIN'S
CREED
IDENTITY

Soldier





Environment Art

- Signature renaissance era environments
- Use exported assets as basis to create level art
- Defined tight memory budget of 100 MB for Mesh + Texture Data
- Even without open world enough space for interesting stories to tell



Concept for AC 2

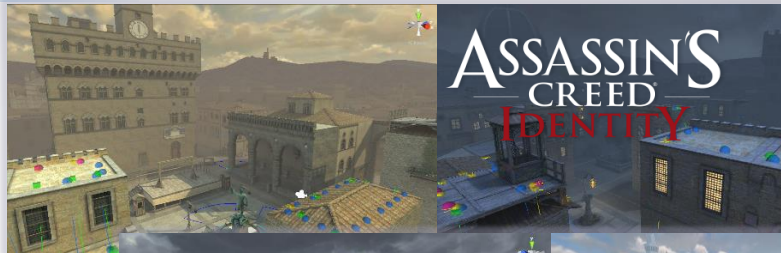


Environment Art

Using as much Unity as possible:

- Static batching
- Lightmaps and LightProbes
- NavMesh navigation, OffMeshLinks for Climbing

~40 MB Mesh Data, ~60 MB Texture data
(including Lightmap)



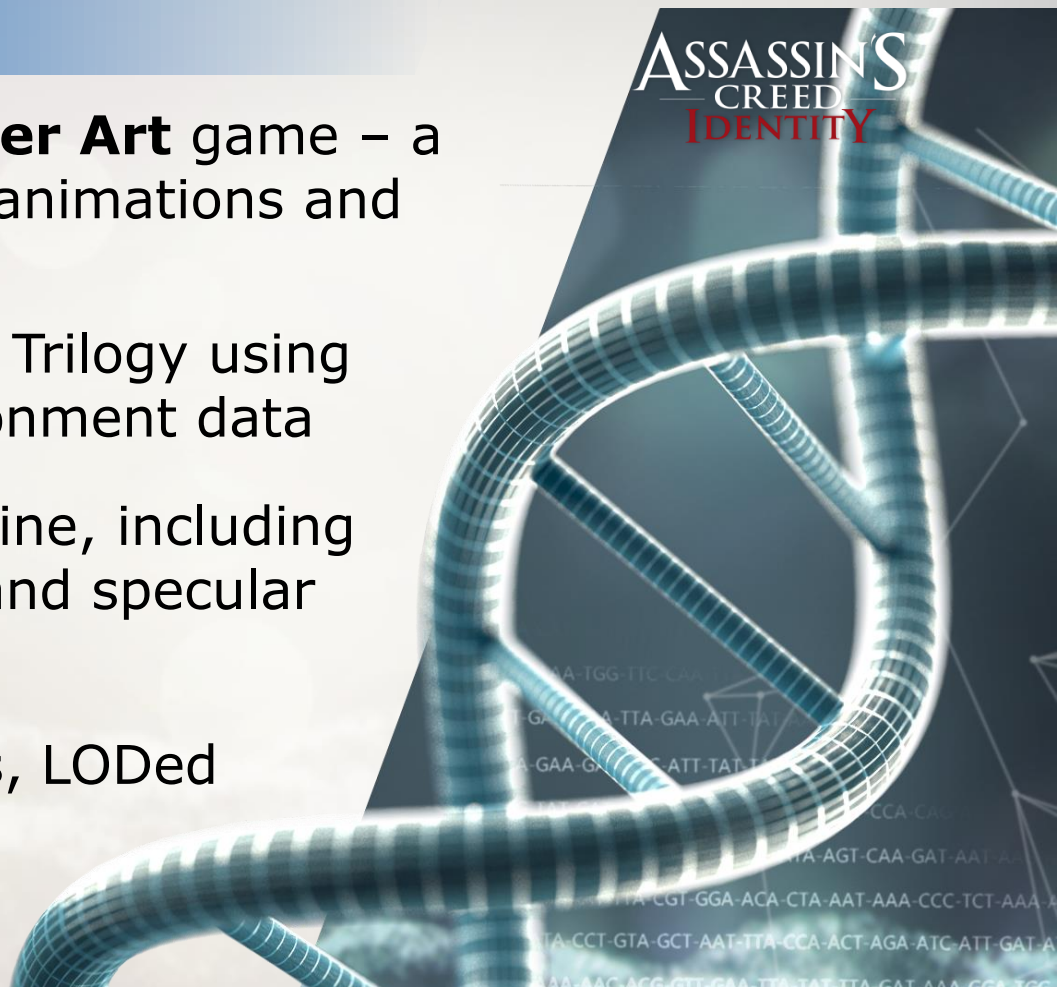


Character Art

Assassin's Creed is a **Character Art** game – a lot of Characters, diversity in animations and type.

- Exported Meshes from AC2 Trilogy using the same pipeline as environment data
- Full fledged rendering pipeline, including glossy surfaces, AO maps and specular maps
- High-Poly player characters, LODed game characters

ASSASSIN'S
CREED
IDENTITY





Character Art

Low-Poly: Crowd characters.
Made for diversity – heads and accessories as well as color variations randomized. Limited Animations



Medium-Poly: Enemy Characters.
Preset color settings, heads randomized, extended animations depending on type (e.g. Archer, Swords, Seekers...)

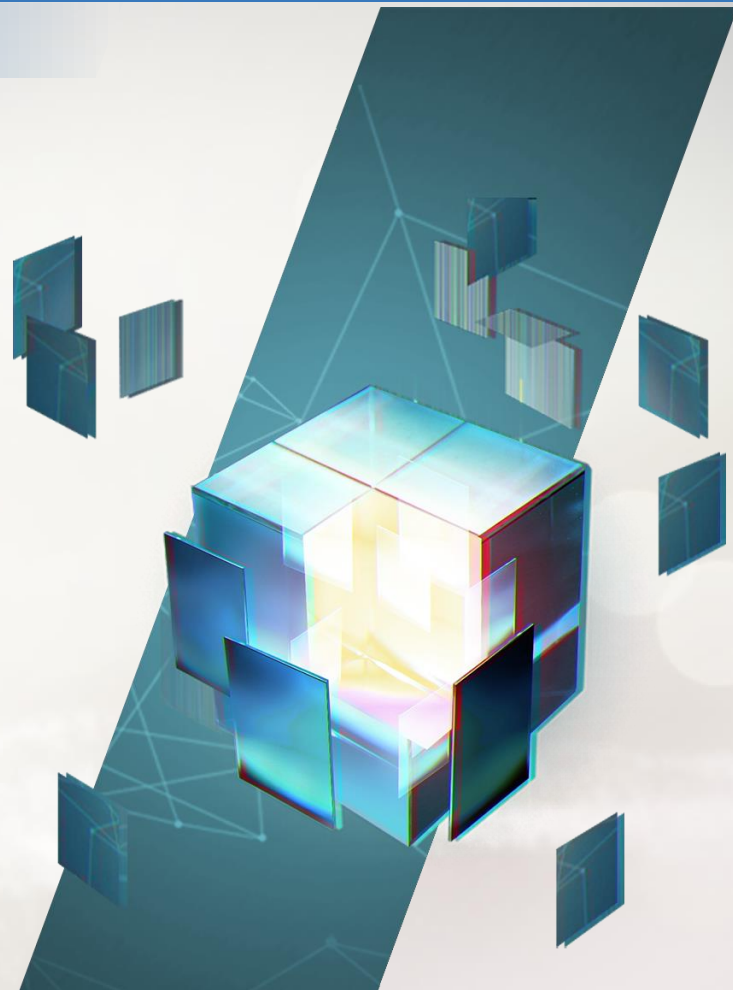
High-Poly: Player Characters. Color, Outfit, weapons and faces customizable, ~**800** animation clips with ~**2400** transitions





Agenda

1. Project introduction
2. Engine Selection
3. Content Creation
- 4. Architecture Overview**
5. Game Logic – Game Simulation
6. Unity Editor Extensions
7. C# for Unity
8. Conclusion





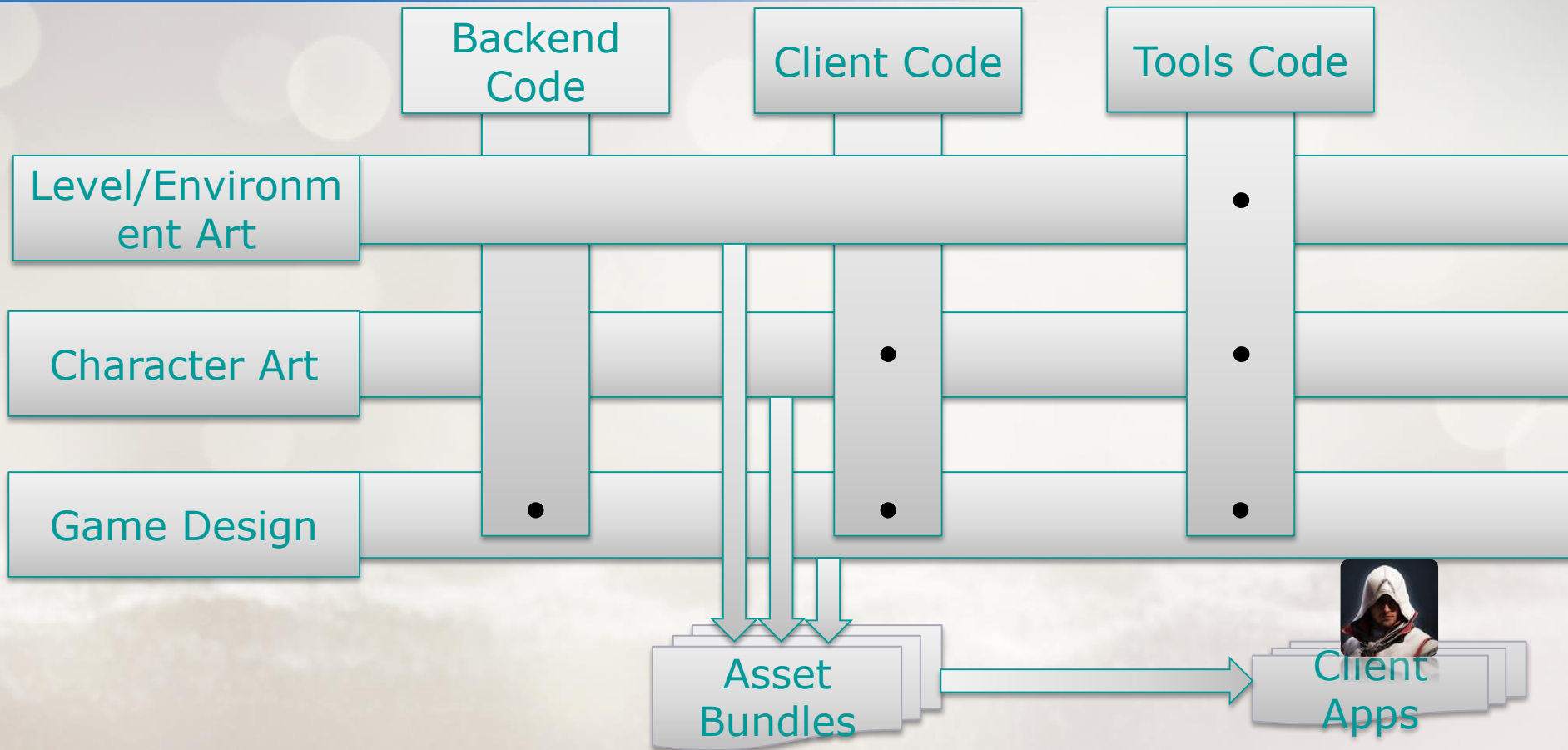
Architecture Overview

- Priority 1 – Collaboration
foster cross-department working
- Priority 2 – Let people work
Don't let them break your build.
- Priority 3 – stay agile
Decouple schedules and streamline processes





Code Architecture

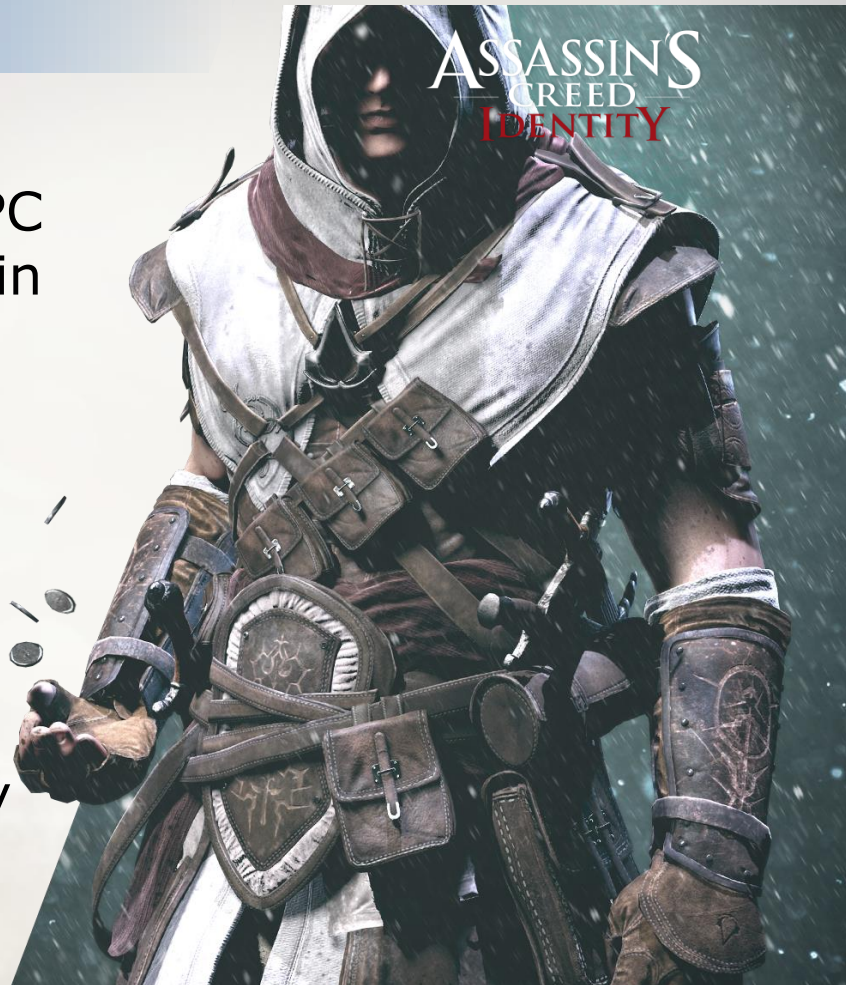




Gameplay Engineering

Prerequisites:

- A lot of experience with Console/PC Games using proprietary engines in the team
- Controlling and Debugging Unity Engine Monobehaviors figured out to be challenging
- Separating Game Logic from Unity Logic as much as possible

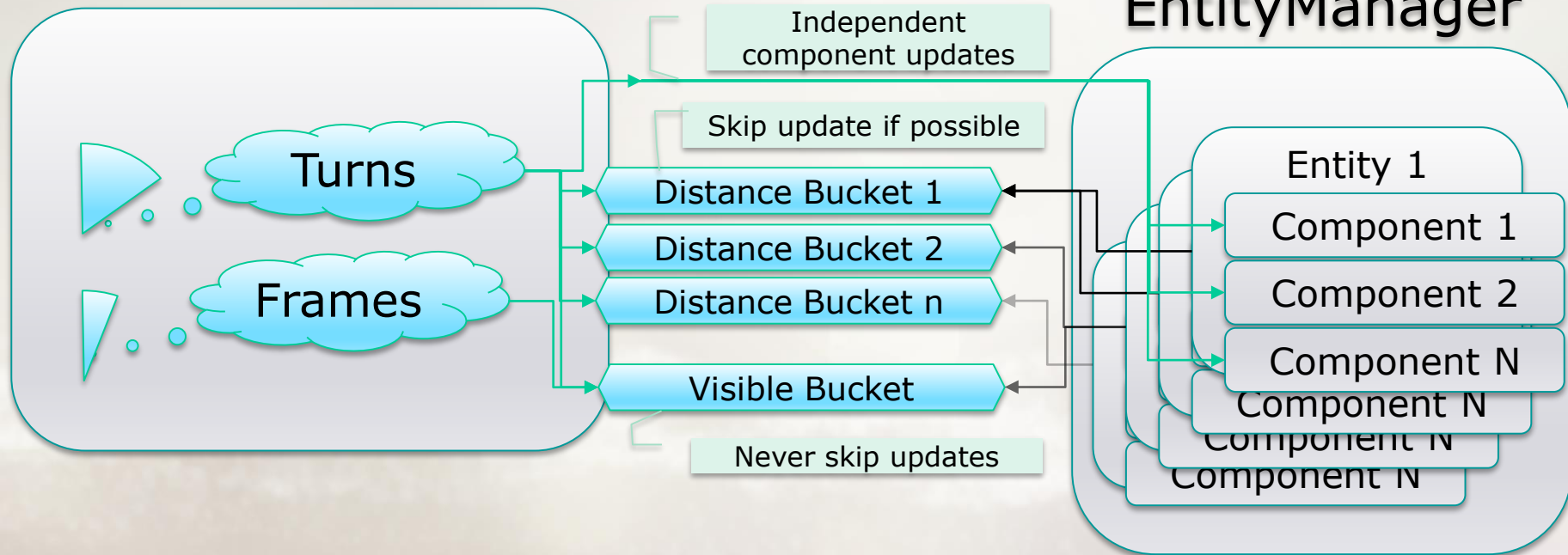




Updating Strategies

Example #1: Optimizing update times for game logic entities

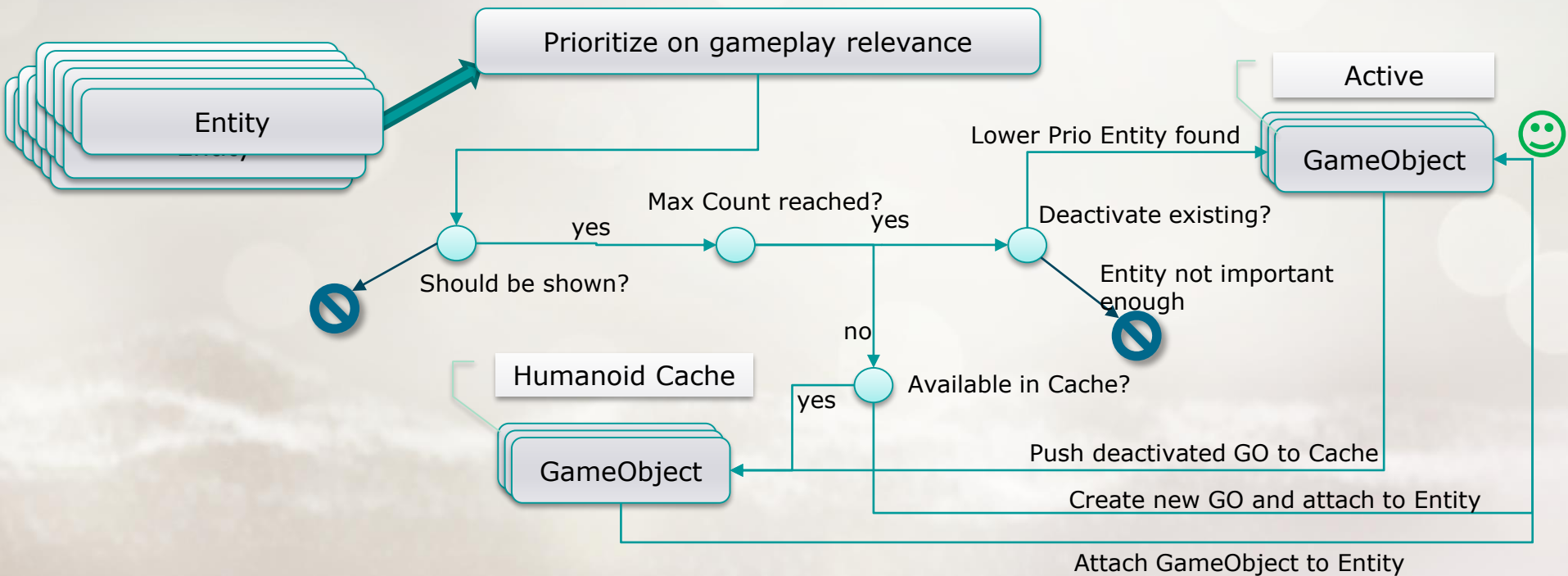
TurnHandler





Updating Strategies

Example #2: (re)use a limited amount of 3D characters





Gameplay Pitfalls

Why rewriting an Entity-Component Model for the Unity Engine?

- Compound Entities are only clone-able by prefabs, and wanted something like this:

```
public class PlayerCharacter : AssassinCharacter, IPlayerCharacterStateMachinesCarrier,  
    IPlayerEventHandlerCarrier, IParticipantCarrier, IVisionCarrier {  
    public PlayerCharacter()  
        : base(entity => new PlayerCharacterStateMachines(entity)) {...
```

- Start, Awake, Update and FixedUpdate did not allow the granularity of control we wanted
- Was it worth doing it? So - So ???



Gameplay Pitfalls

Pros and Cons of our approach?

```
public class PlayerCharacter : AssassinCharacter, IPlayerCharacterStateMachinesCarrier,
    IPlayerEventHandlerCarrier, IParticipantCarrier, IVisionCarrier {
    public PlayerCharacter()
        : base(entity => new PlayerCharacterStateMachines(entity)) {...
```

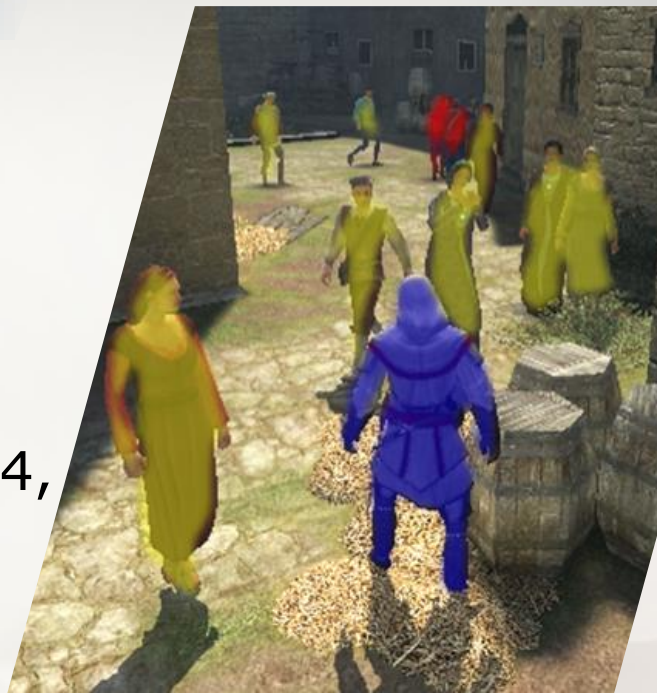
- Code complexity grew overall app size
- Using Interfaces + Generics caused some trouble with Unity Mono
- IL2CPP translations turned out to be challengin






Performance Result

Some numbers, please!

- Campaign-missions have **100+** relevant entities
- Simulation takes **4 ms** per frame on iPad 4, **+8 ms** for turn updates every 100 ms
- Logic Pooling configured to max 10 visible + 5 cached



-  Player entity – full logic, always updated
-  Enemy entities – full logic simulation, updated as needed
-  Crowd entities – no logic simulation



Agenda

1. Project introduction
2. Engine Selection
3. Content Creation
4. Architecture Overview
5. Game Logic – Game Simulation
- 6. Unity Editor Extensions**
7. C# for Unity
8. Conclusion

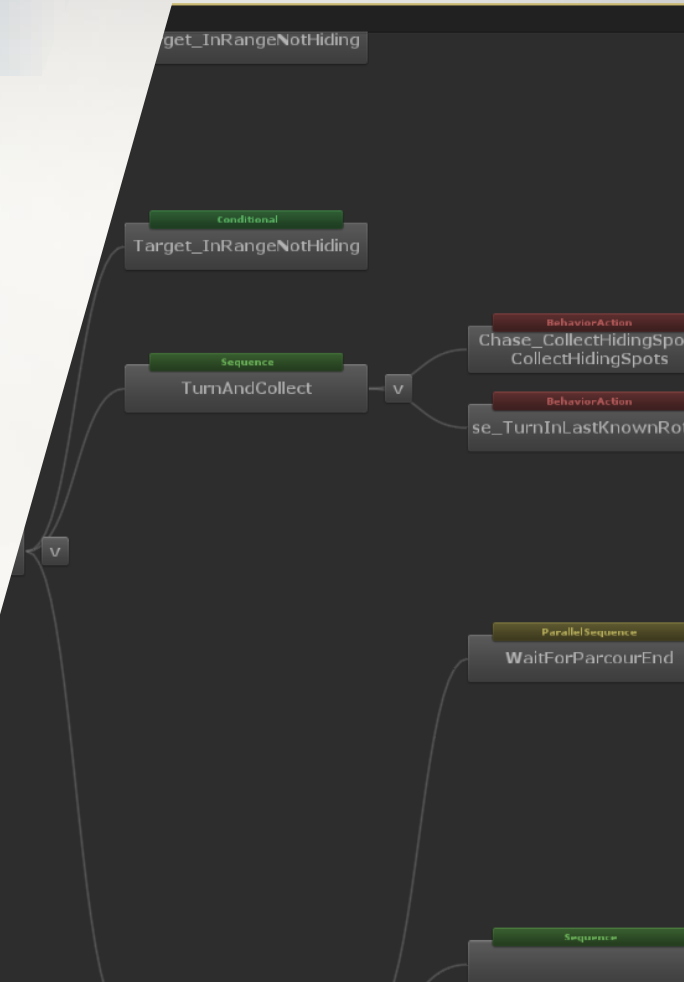




Behaviour Tree Editor

AI is driven by behavior trees:

- Enemy chasing, seeking and patrolling
- Crowd reacting on external influences (Smoke bombs, fights, assassinations etc.)
- In Combat, the behavior defines how participants attack, block or flee
- Custom Unity Editor implemented to **create** and **watch** tree transitions visually





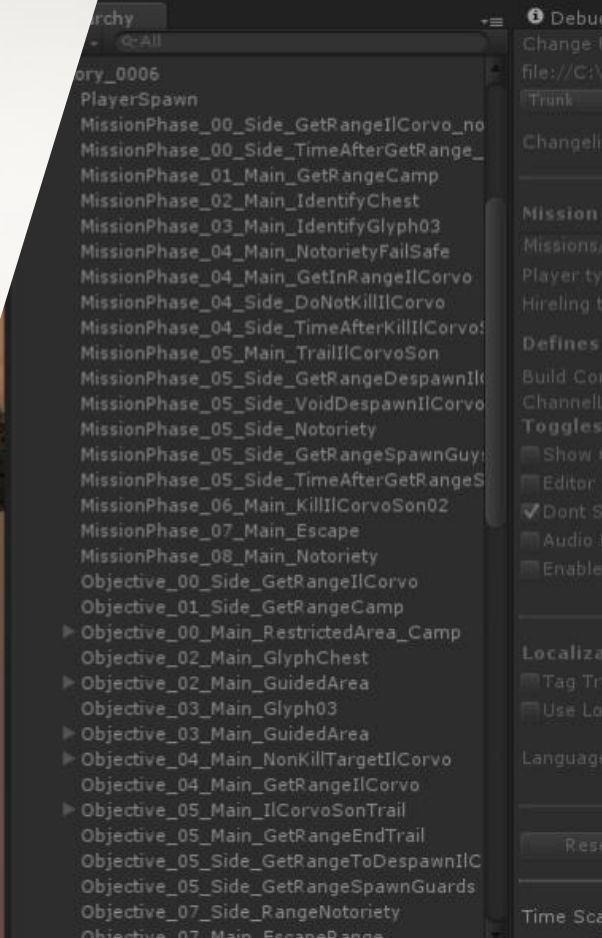
Mission Editor

Missions completely authored in Unity Editor:

- Mission nodes are embedded flawless into the Unity GameObject Hierarchy
- Mission data added as components to the GameObjects

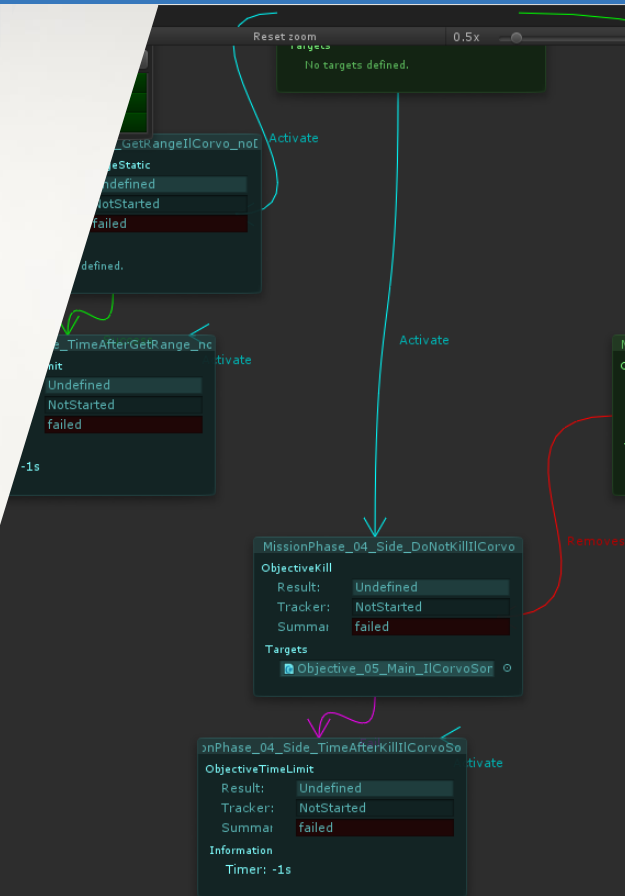
2 use cases:

- Hand-crafted Campaign missions
- Templates for the procedural mission generation for nearly endless variations and challenge missions





- Efficiently serialized to Protobuf blobs – 23 Kb contain:
 - ~100 Entities
 - 30 Patrols and 200 waypoints
 - Objective tree
- Unity Editor extension added to use graphViz to visualize the mission flow and state at runtime
- Several Editor Inspectors written to have a completely visual mission editor flow

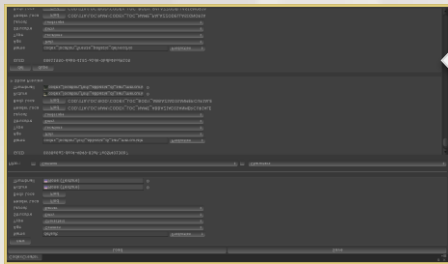




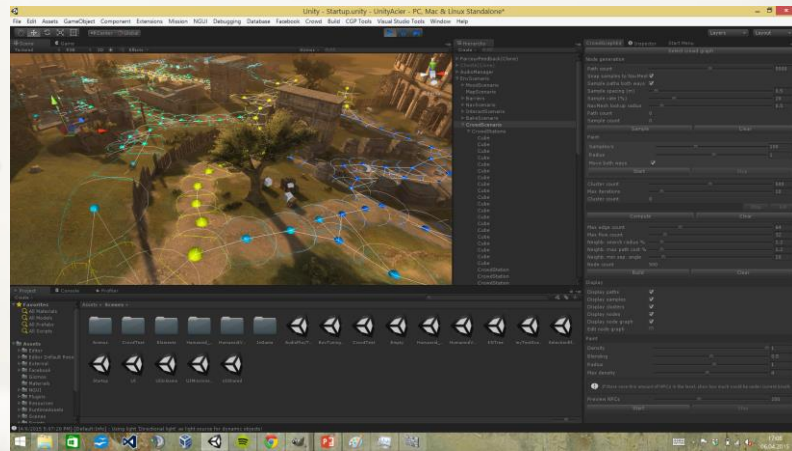
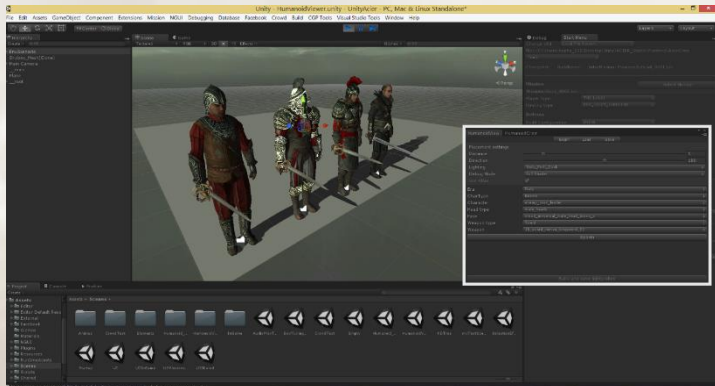
Editor Extensions

Creating Editor Extensions in C# is easy and we used it a lot!

Game definition editors to bundle graphic assets to text and additional information – no more Excel or XML editing required



Tweaking the Crowd simulation input data while the game is running



Preview any combination of character asset using the correct game rendering pipeline



Agenda

1. Project introduction
2. Engine Selection
3. Content Creation
4. Architecture Overview
5. Game Logic – Game Simulation
6. Unity Editor Extensions
- 7. C# for Unity**
8. Conclusion





C# for Unity

- Mono for Unity is not Xamarin Mono: Due to Licensing with Novell, the state of features is almost frozen at mid 2010 (v2.6x)
<http://docs.unity3d.com/412/Documentation/ScriptReference/MonoCompatibility.html>
- Compatibility with .Net 3.5 compiled C# DLL
- IDEs: Monodevelop (heavily customized) and Visual Studio tools for Unity
- Source code freely available at
<https://github.com/Unity-Technologies/mono/>





C# for Unity

- Careful with Interface/Generics, generic Collection and everything that is not in Unity's micro corlib!
- Beware the Code-Bloat due to AOT/cross-compilation
- Be aware of the "boxing" overhead when using value types in a reference type manner
- For this reason, avoid "foreach" or using default "object" type parameters

<https://msdn.microsoft.com/en-us/library/yz2be5wk.aspx>

<http://makegamessa.com/discussion/1493/it-s-official-foreach-is-bad-in-unity>



C# for Unity

- Use Unity Profiler in “Deep Profile” mode:
<http://docs.unity3d.com/Manual/Profiler.html>
- allocation differs on device: use remote profiling or allocation tracking in XCode
- Write your own simplistic measures to profile across coroutines and frames

Hierarchy							CPU: 77.77ms GPU: 7.95ms						
Overview							Object	Total	Self	Calls	GC Alloc	Time ms	Self ms
▼ Main.Update()	57.7%	0.0%	1	98.8 KB	44.95	0.00	N/A	0.9%	0.0%	1	1.0 KB	0.77	0.00
▼ Profiling.Start()	57.7%	0.0%	1	98.8 KB	44.94	0.01	N/A	0.1%	0.0%	1	384 B	0.13	0.00
▼ MissionManager.Update()	55.4%	0.0%	1	97.2 KB	43.10	0.00	N/A	0.1%	0.0%	1	384 B	0.12	0.00
▼ Profiling.Start()	55.4%	0.0%	1	97.2 KB	43.10	0.00	N/A	0.1%	0.0%	1	384 B	0.12	0.00
▶ MissionPhase.Update()	36.4%	0.0%	1	76.8 KB	28.37	0.01	N/A	0.1%	0.0%	1	184 B	0.11	0.00
▼ LogicMain.UpdateFrame()	18.4%	0.0%	1	19.8 KB	14.36	0.00	N/A	0.1%	0.0%	1	184 B	0.10	0.00
▼ Profiling.Start()	18.4%	0.0%	1	19.7 KB	14.35	0.00	N/A	0.1%	0.0%	1	184 B	0.09	0.00
▼ TurnHandler.Execute()	18.4%	0.0%	1	19.7 KB	14.31	0.00	N/A	0.1%	0.0%	1	184 B	0.08	0.00
▼ Profiling.Start()	18.4%	0.0%	1	19.7 KB	14.31	0.01	N/A	0.1%	0.0%	1	184 B	0.07	0.00
▼ Profiling.Start()	17.3%	0.0%	1	19.4 KB	13.49	0.00	N/A	0.0%	0.0%	1	104 B	0.07	0.00
▼ TurnHandler.UpdateEntities()	17.3%	3.2%	1	19.4 KB	13.48	2.50	N/A	0.0%	0.0%	1	184 B	0.07	0.00
▶ StateMachines.DoTurnUpdate()	4.7%	0.1%	73	9.4 KB	3.73	0.09	N/A	0.0%	0.0%	1	184 B	0.07	0.00
▶ Patrol.DoTurnUpdate()	1.7%	0.0%	3	1.7 KB	1.36	0.04	N/A	0.0%	0.0%	1	184 B	0.06	0.00
▶ EntityStats.DoTurnUpdate()	1.5%	0.0%	40	2.6 KB	1.17	0.05	N/A	0.0%	0.0%	1	24 B	0.06	0.00
▶ PlayerStats.DoTurnUpdate()	1.2%	0.0%	1	1.6 KB	0.98	0.00	N/A	0.0%	0.0%	1	184 B	0.05	0.00
▶ CharacterAi.DoTurnUpdate()	1.0%	0.0%	2	1.7 KB	0.85	0.00	N/A	0.0%	0.0%	1	184 B	0.04	0.00
▶ EntityStats.DoFrameUpdate()	1.0%	0.0%	49	1.2 KB	0.81	0.05	N/A	0.0%	0.0%	1	184 B	0.04	0.00
▶ SpacialEntity.DoTurnUpdate()	0.9%	0.3%	84	0 B	0.76	0.30	N/A	0.0%	0.0%	1	184 B	0.04	0.00
▶ Buffs.DoTurnUpdate()	0.3%	0.0%	40	0.9 KB	0.28	0.04	N/A	0.0%	0.0%	1	184 B	0.04	0.00



C# for Unity 5

- Unity 5 brought IL2CPP for iOS: C#->CIL->C->Native
- Compile times increase massively for large codebase – 15 min for app (+13 min), 30 min with xcode instruments (+26 min)
- iOS C-Language projects can be prepared on Windows machines which might be faster to check code transformation
- No iOS debugging app size limitation anymore – use XCode!
- IL2CPP internals blog pages provide lots of insights:
<http://blogs.unity3d.com/2015/05/06/an-introduction-to-ilcpp-internals/>



C# for Unity

- NO JIT allowed: iOS use ahead-of-time compilation - every type must be fully qualified and referenced at compile time
- Use as few Reflection as possible – if you have to, be aware of link.xml to add “linker-exceptions” to enforce the code to be compiled into player (for Android with Stripping enabled)
- Prevent forwarding generic types as much as possible
- Try to not rely too much on Interface + Generics in combination, as it is likely to create code ambiguity and leads to compiler crashes



Conclusion

Unity supports big Codebase even on mobile

- more than 450k lines of code already

Decide as early as possible:

- Use Micro-Corlib
- Multithreaded updates?

Careful with Coding style:

- Prevent Value-type Generics
- Keep It Simple, Stupid! ;)
- Profile, Profile, Learn, Profile





Conclusion

Mind the build times!

- Building one environment might take 9 hours and more with our project
- Asset Bundling is version dependent – extra care with custom importers!

Some obvious but valid points:

- Prevent any asset loading or big memory allocation at runtime
- 2k uncompressed UI atlas still weights 16 MB *twice* at loading





Adapt, Improve, Find Bliss





Thank you! Q&A

tobias.tost@bluebyte.de

We are hiring!

<http://bluebyte.de/jobs/>

ASSASSIN'S
CREED
IDENTITY



Blue
Byte

A UBISOFT STUDIO

