Unity Best Practices

Performance

Unity Best Practices

- https://storm8.atlassian.net/wiki/display/UNITY/Best+Practices
- https://storm8.atlassian.net/wiki/display/UNITY/Performance+Guide
- https://storm8.atlassian.net/wiki/display/UNITY/UI+Optimization+Guide
- https://storm8.atlassian.net/wiki/display/UNITY/Hidden+Allocations

Presentation

- these slides

DebugObjectLogger Tool

- https://storm8.atlassian.net/wiki/display/UNITY/GameObject+Profiling

Incorporating Unity's advice to optimize our games and upcoming Unity on-site

- https://storm8.atlassian.net/browse/PIPE-399

General Goals

- Fewer GameObjects, Update()s and coroutines
- Never use LINQ or features that cause hidden GC
- Pool/recycle GameObjects as much as possible
- Limit use of GetComponent() and HasComponent()
- Try not to use Find() or FindObjectsOfType() at all
- Use basic for () and while () loops instead of foreach
- Try to batch UI and objects as much as possible
- Don't use GameObjects for non-visual purposes
- Don't use reflection methods such as GetMethod()


GameObject
Update()
CoRoutine
GameObject
GameObject
FixedUpdate()
CoRoutine

These aren’t inherently bad

The key is “everything in moderation”

Alternatives

Re-use GameObjects as much as possible

Re-use GameObjects as much as possible

Instead of creating new ones each time

Use Controllers to do the work

Use controllers to do the work

Instead of each object having an Update()


GameObject.Find()
GetComponent()
AddComponent()
foreach ()
FindObjectsOfType()
HasComponent()
LINQ

Restrictions

Never use GameObject.Find() or FindObjectsOfType()

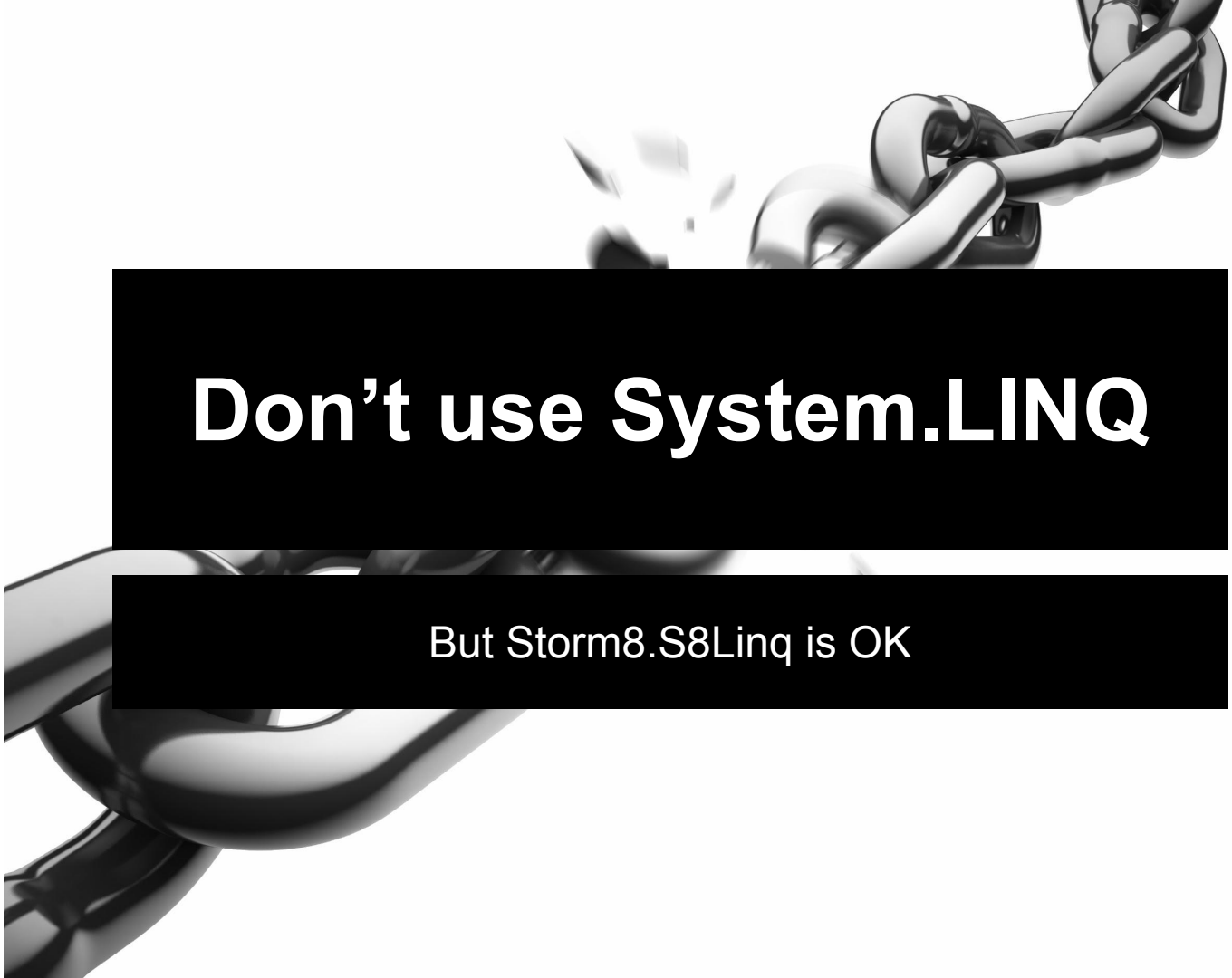
Only use GetComponent() and AddComponent() in Awake()/Start()

Don’t use LINQ (S8Linq OK) or “foreach” (creates GC)


Don’t use
GameObject.Find()


Don’t use
GameObject.Find()
Connect property “outlets” in the scene


Don’t use System.LINQ



Don’t use System.LINQ

But Storm8.S8Linq is OK


DrawCall()
DrawCall()
DrawCall()
DrawCall()
DrawCall()
DrawCall()
DrawCall()

Batch Draw Calls as much as possible

Batch Draw Calls as much as possible

Draw calls are expensive, fewer the better


Don’t use Reflection
until last resort


Don’t use Reflection until last resort
Reflection is slow, and can bloat binary