Multi-Threaded Applications in Unity

By Will Blackney

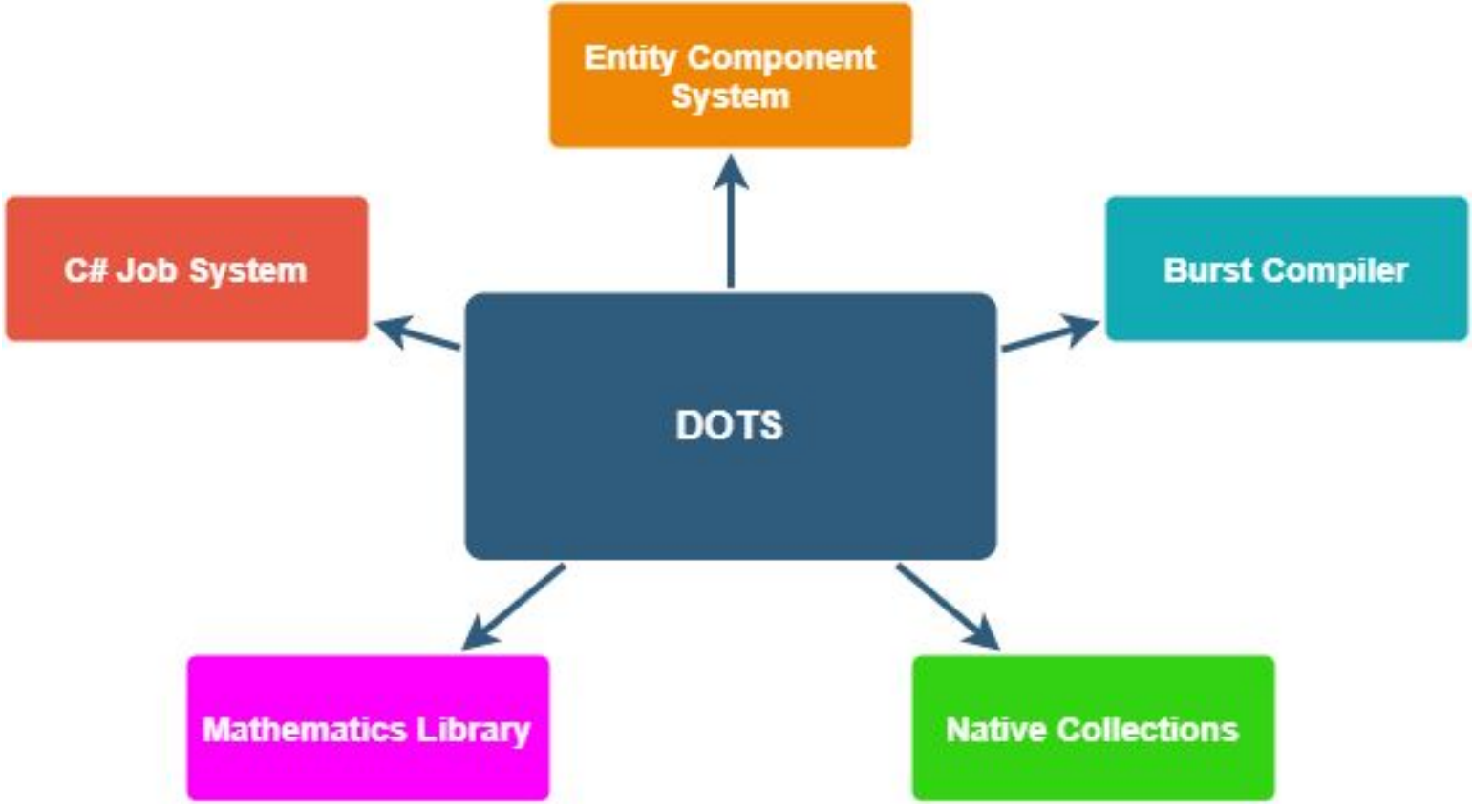
Main Question:

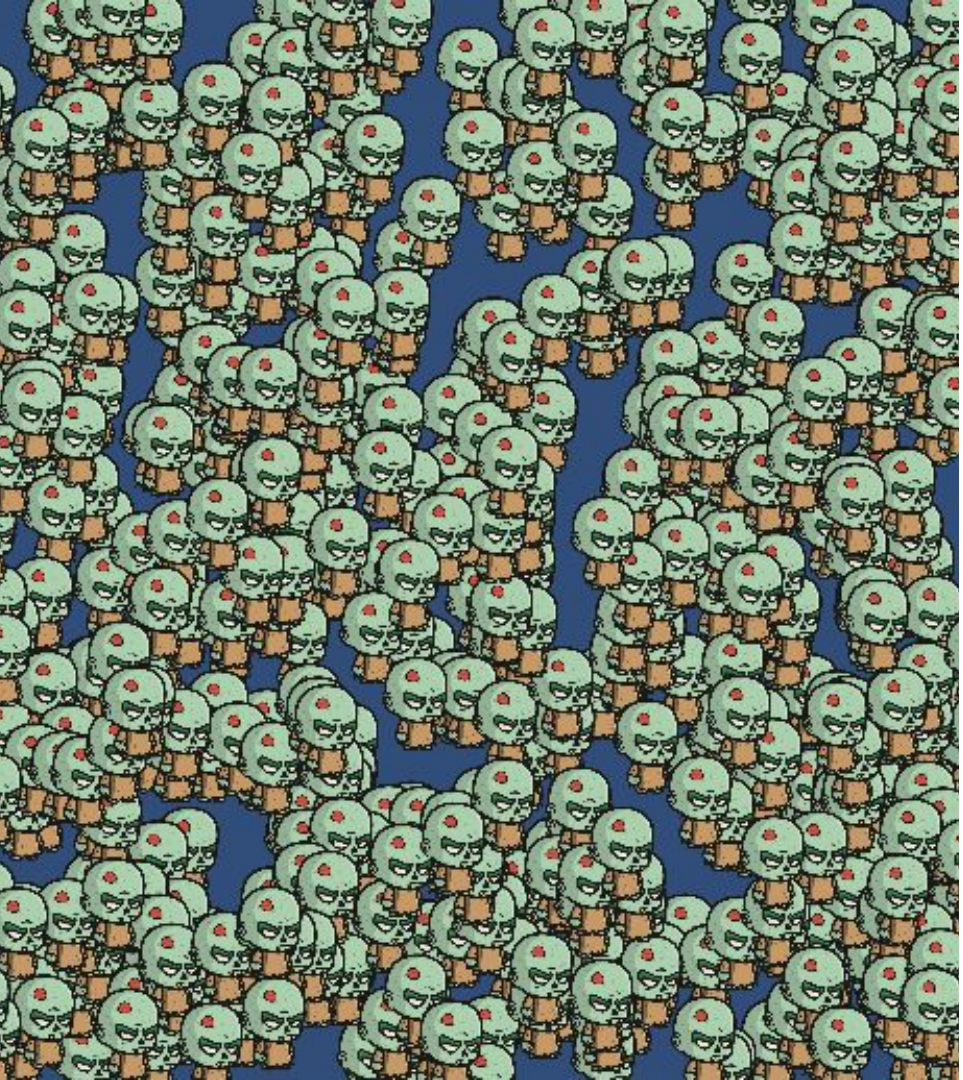
- How does Unity's *Data Oriented Technology Stack* (DOTS) change multi-threaded software development with Unity?
- Why DOTS and not the 'C# Job System'?

What exactly is DOTS?

- Data Oriented Technology Stack
- A suite of applications, packages and libraries for making high performance multi-threaded software
  - Entity Component System
  - C# Job System
  - Burst Compiler
- *“DOTS is Unity’s attempt at reshaping its internal infrastructure in a way that is faster, lighter and optimized for the current massive multi-threaded world”.* - Joachim Ante, Unity CTO & Co-Founder

Components of DOTS





One Thousand Zombies!

- A simple application that moves 1000 zombies up and down the screen
- Two versions: ‘Old School’ multithreading, and ‘C# Jobs System’ multithreading
- Demonstrates the power and performance capabilities of DOTS

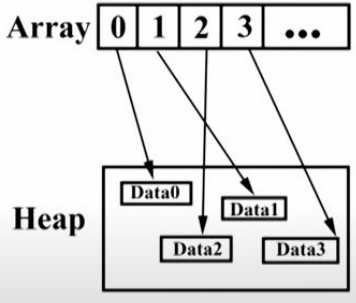
How does DOTS change multi-threaded software development within Unity?

- Doesn't just change MT implementation: it changes implementation of everything
- Architecture change: Object Oriented Design to Data Oriented Design
- Multithreading is more prevalent. All code written can (and is) run on child threads
- Developers have more low level control
- Many new software possibilities
- Improved performance/functionality on pre-existing applications

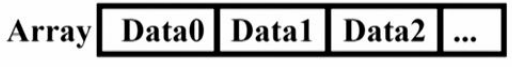
DOTS Memory Management

- Where viable, DOTS implementation dictates the use of structs over classes.
- DOP memory allocation is linear, and therefore more efficient
- Classes are always assigned to random locations in memory
- No randomly jumping around memory
- Takes advantage of modern CPU prefetching
- Prevent loading of unnecessary data into cache

Object-oriented programming



Data-oriented programming



Thank you for listening!