Chapter 1: Evaluating Performance Problems
Chapter 2: Scripting Strategies

GetComponent(string) finished: 6413.00ms total, 0.006413ms per test for 1000000 tests
UnityEngine.Debug.Log(Object)

GetComponent(ComponentName) finished: 89.00ms total, 0.000089ms per test for 1000000 tests
UnityEngine.Debug.Log(Object)

GetComponent(typeof(ComponentName)) finished: 95.00ms total, 0.000095ms per test for 1000000 tests
UnityEngine.Debug.Log(Object)
<table>
<thead>
<tr>
<th>Clear</th>
<th>Collapse</th>
<th>Clear on Play</th>
<th>Error Pause</th>
<th>Connected Play</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Icon" /></td>
<td><img src="image2.png" alt="Icon" /></td>
<td><img src="image3.png" alt="Icon" /></td>
<td><img src="image4.png" alt="Icon" /></td>
<td><img src="image5.png" alt="Icon" /></td>
</tr>
<tr>
<td>UnityEngine.Debug:Log(Object)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A new enemy was created! Tom</td>
<td>UnityEngine.Debug:Log(Object)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A new enemy was created! Dick</td>
<td>UnityEngine.Debug:Log(Object)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A new enemy was created! Harry</td>
<td>UnityEngine.Debug:Log(Object)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A new enemy was created! Tom</td>
<td>UnityEngine.Debug:Log(Object)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A new enemy was created! Tom</td>
<td>UnityEngine.Debug:Log(Object)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Chapter 3: The Benefits of Batching
Dynamic Batching

Batches: 9

Statistics

Audio:
Level: -74.8 dB
Clipping: 0.0%
DSP load: 0.8%
Stream load: 0.0%

Graphics:
303.5 FPS (3.3ms)
CPU: main 3.0ms render thread 2.4ms
Saved by batching: 0
Txs: 31k Vts: 2.2k
Screen: 691x610 ~ 4.8 MB
SetPass calls: 9 Shadow casters: 0
Visible skinned meshes: 0 Animations: 0
Network: (no players connected)
Chapter 4: Optimizing Your Art Assets
128x128 (each)

16 Textures
16 Materials
16 Draw Calls

512x512

1 Texture
1 Material
1 Draw Call

ATLASING
Chapter 5: Faster Physics

The physics cycle may happen more than once per frame if the fixed timestep is less than the actual frame update time.
Two objects in motion

Discrete Collision Detection

Two very fast objects in motion

Discrete Collision Detection

Continuous Collision Detection

Convex

Concave
### Layer Collision Matrix

<table>
<thead>
<tr>
<th></th>
<th>Default</th>
<th>TransparentFX</th>
<th>Ignore Raycast</th>
<th>Water</th>
<th>UI</th>
<th>Player</th>
<th>Enemies</th>
<th>Player Missiles</th>
<th>Enemy Missiles</th>
<th>Powerups</th>
<th>World</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TransparentFX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ignore Raycast</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Water</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Player</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>Enemies</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>Player Missiles</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enemy Missiles</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Powerups</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td></td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>World</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td></td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>
Chapter 6: Dynamic Graphics

CPU

Game → Graphics API (OpenGL, DirectX) → Hardware Driver

CPU-GPU Boundary

Front End

Rasterization ← Primitive Assembly ← Vertex Shaders

GPU

Back End

Pixel (Fragment) Shaders → Frame Buffer

---

[22]
Scene View  
Overdraw

Occlusion Culling Disabled

Occlusion Culling Enabled

Particle System

- Emission
- Shape
- Velocity over Lifetime
- Limit Velocity over Lifetime
- Inherit Velocity

Automatic culling is disabled because:
- Local space simulation is not being used.
- Trails module is enabled.
Chapter 7: Optimizations for Virtual and Augmented Reality

[Diagram showing the lifecycle of a technology trigger and its impact on visibility over time, with stages labeled: Peak of Inflated Expectations, Plateau of Productivity, Trough of Disillusionment.]
Chapter 8: Masterful Memory Management

1. Empty heap space

2. Four 64-byte objects allocated

3. Objects A and C deallocated

4. New 128-byte object allocation

---

Memory
- Total Allocated
- Texture Memory
- Mesh Memory
- Material Count
- Object Count
- Total GC Allocated
- GC Allocated

41.9 MB
5.6 GB
1.4 MB
14.8 MB
18.0 KB
11 Orcs (5 active, 6 inactive)
8 Trolls (3 active, 5 inactive)
5 Ogres (2 active, 3 inactive)
1 Dragon (1 active)

**Heap Memory**

<table>
<thead>
<tr>
<th>Orc1</th>
<th>Orc2</th>
<th>Orc3</th>
<th>Orc4</th>
<th>Orc5</th>
<th>Orc6</th>
<th>Orc7</th>
<th>Orc8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orc9</td>
<td>Orc10</td>
<td>Orc11</td>
<td>Troll1</td>
<td>Troll2</td>
<td>Troll3</td>
<td>Troll4</td>
<td>Troll5</td>
</tr>
<tr>
<td>Troll6</td>
<td>Troll7</td>
<td>Troll8</td>
<td>Ogre1</td>
<td>Ogre2</td>
<td>Ogre3</td>
<td>Ogre4</td>
<td>Ogre5</td>
</tr>
<tr>
<td>Dragon</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Pooling System**

**Inactive**

- Orc7
- Orc9
- Orc6
- Orc10

**Active**

- Orc1
- Orc5
- Orc2
- Orc8
- Orc11
- Troll3
- Troll1
- Troll7

**New Orc is spawned**

1. Determine which Pool corresponds to the given Prefab
2. The first inactive Orc in the Inactive Group (Orc7) is activated - the corresponding object in the Heap is therefore activated
3. Newly-spawned Orc is moved to Active group

**Heap Memory**

<table>
<thead>
<tr>
<th>Orc1</th>
<th>Orc2</th>
<th>Orc3</th>
<th>Orc4</th>
<th>Orc5</th>
<th>Orc6</th>
<th>Orc7</th>
<th>Orc8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orc9</td>
<td>Orc10</td>
<td>Orc11</td>
<td>Troll1</td>
<td>Troll2</td>
<td>Troll3</td>
<td>Troll4</td>
<td>Troll5</td>
</tr>
<tr>
<td>Troll6</td>
<td>Troll7</td>
<td>Troll8</td>
<td>Ogre1</td>
<td>Ogre2</td>
<td>Ogre3</td>
<td>Ogre4</td>
<td>Ogre5</td>
</tr>
<tr>
<td>Dragon</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Pooling System**

**Inactive**

- Orc3
- Orc4

**Active**

- Orc1
- Orc5
- Orc8
- Orc11
- Orc2
- Troll1
- Troll7
- Ogre3
- Ogre5

1. Orc7
2. Orc1
3. Orc2
Ogre3 is despawned
1. Determine which Pool corresponds to the given object
2. Deactivate Ogre3 - the corresponding object in the Heap is therefore deactivated
3. Move Ogre3 to Inactive group

New Dragon is spawned
1. Determine which Pool corresponds to the given Prefab
2. Inactive group is empty, so a new instance of Dragon must be created
3. Instantiate a new Dragon from the Prefab on the heap
4. Add the newly created Dragon to the Active list
Chapter 9: The Data-Oriented Technology Stack
Unity Data-Oriented Technology Stack (DOTS)

- C# Job System
- Entity-Component System (ECS)
- Burst Compiler
Chapter 10: Tactical Tips and Tricks
[ERROR] This is a *very specific* kind of log message