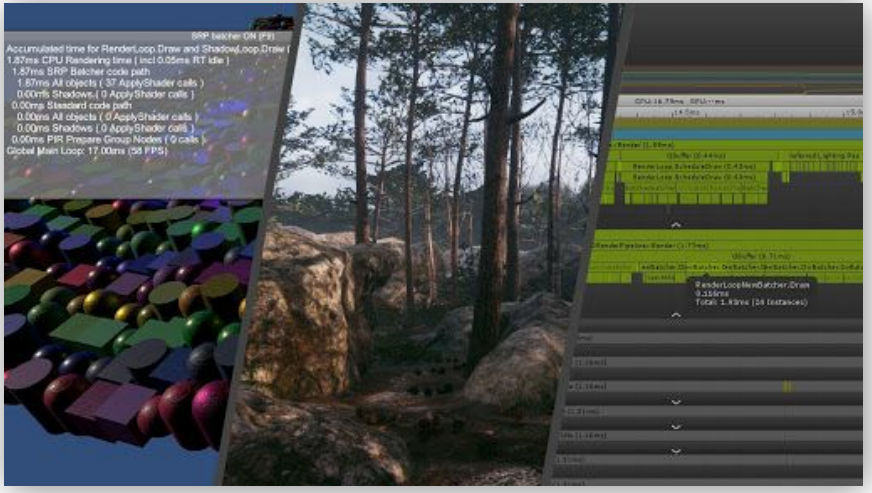
Unity SRP Batcher

For WebGL



Brendan Duncan | Senior Software Engineer | July 13, 2021

unity

Disclaimer

The following is intended for informational purposes only.

Unity is not committing to deliver any functionality, features or code.

The development, timing and release of all products, functionality and features are at the sole discretion of Unity, and are subject to change.

Unity Real-time Engine and Editor

— The world is a better place with more creators in it.

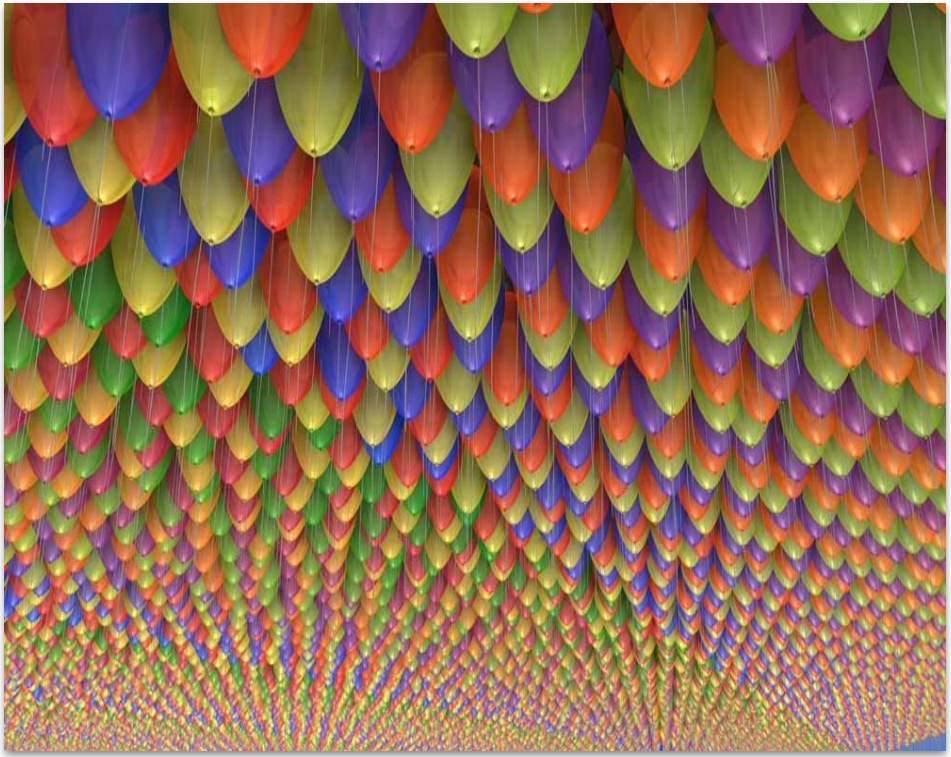
— Supports over 25 target platforms and technologies.

magic leap · Microsoft HoloLens · ARCore · tvOS · PlayStation.VR · androidtv · XBOX ONE · WebGL · iOS · PS4 · android

unity

What Problem Are We Try Trying To Solve?

- Rendering optimizations often focus on reducing draw calls.
- Objects with same material can be batched together: shader set once, multiple objects drawn.
- Objects with unique materials require separate shader updates, can’t be batched.



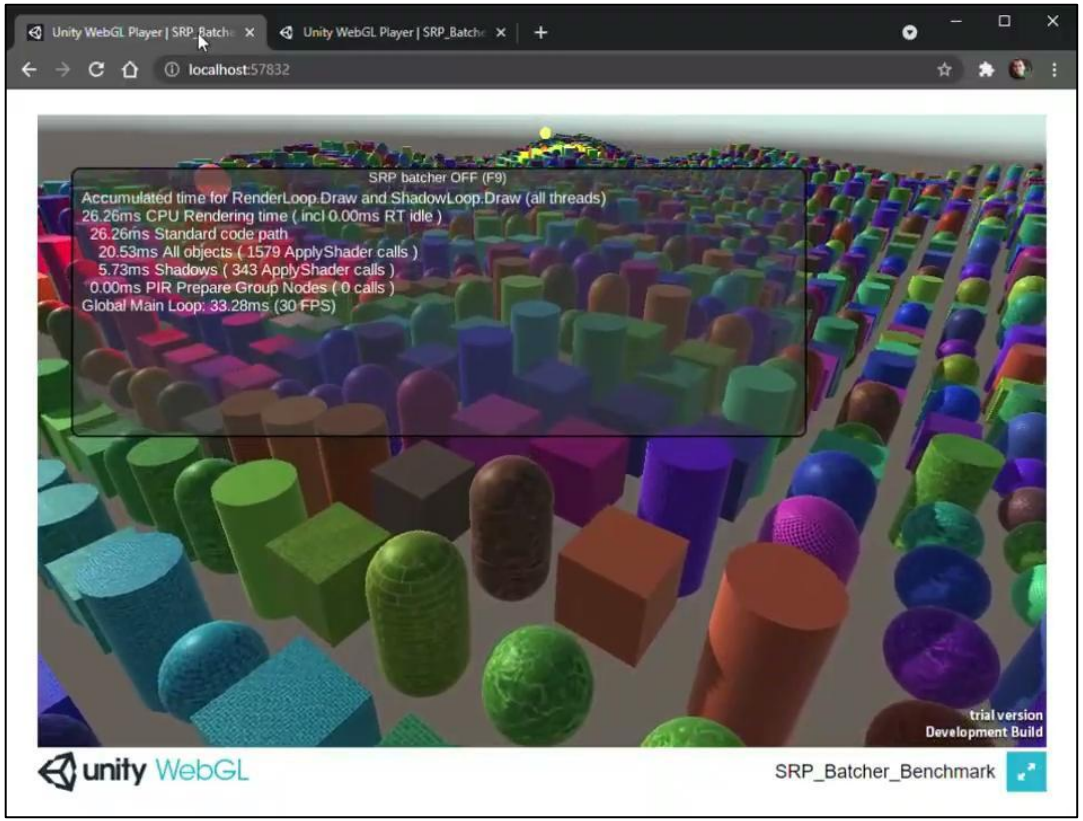
Benchmark Test: Worst Case Scenario

1600 Objects With Unique Materials

4 Realtime Lights

1 Directional Shadow Map

Avg: 33ms / Frame



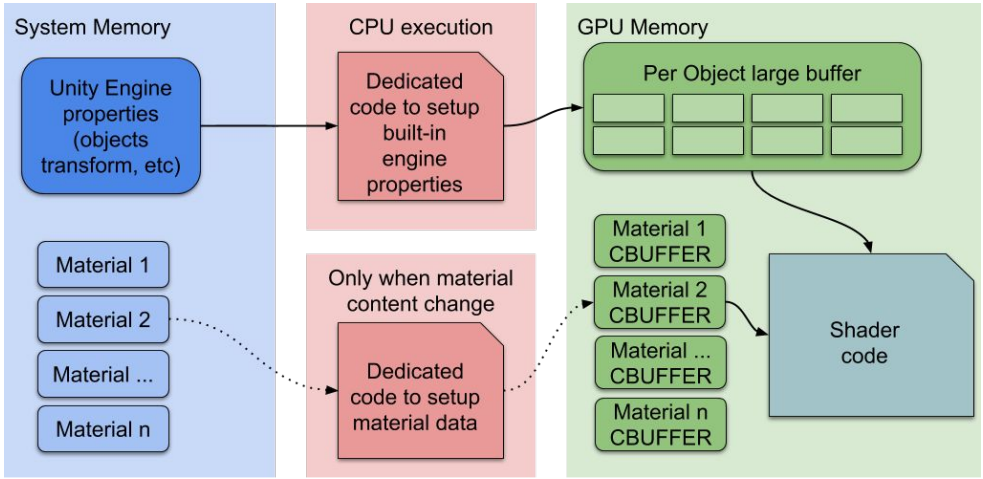
unity

What Is SRP Batcher?

- A feature of Unity's Scriptable Render Pipeline.
- Batches together draw calls for objects that share the same shader variant, even with different materials.
- Separate uniform buffers manage built-in properties, and per object material properties.
- Materials that don't change require no CPU overhead, persistent int GPU memory.

How SRP Batcher Works

- Separate uniform buffers for built-in engine properties like object transforms, per-object material uniform buffers.
- The CPU only needs to update data for built-in engine properties.
- All materials have persistent GPU uniform buffers and only need to be updated when modified.



unity

WebGL Challenges

- SRP Batcher requires GLSL **uniform layout locations** and **buffer binding points**.
- These features are only available in GLSL ES 3.1.
- WebGL2 only supports GLSL ES 3.0, and cannot use these features.

Shader Requirements For SRP Batcher

```
#define UNITY_UNIFORM
#define UNITY_LOCATION(x) layout(location = x)
#define UNITY_BINDING(x) layout(binding = x, std140)

UNITY_BINDING(0) uniform UnityPerDraw {
       UNITY_UNIFORM vec4 hlslcc_mtx4x4unity_ObjectToWorld[4];
    ...
};
UNITY_BINDING(1) uniform UnityPerMaterial {
       UNITY_UNIFORM vec4 _BaseMap_ST;
    ...
};
UNITY_LOCATION(0) uniform mediump samplerCube unity_SpecCube0;
...
```

— GLSL ES 3.1 features are required: uniform layout locations and buffer binding points.

— WebGL does not use integer uniform locations, making WebGL implementation of these features not possible.

Emscripten GLSL Preprocessing

- Unity uses Emscripten compile the engine and project to Web Assembly.
- Emscripten bridges OpenGL to WebGL. It uses a dictionary to associate integer OpenGL locations to WebGL opaque location objects.
- We extended Emscripten to emulate the required GL features.
- The emscripten extension is enabled by a command-line flag. Shader sources are preprocessed by Emscripten before they are sent to WebGL.
  - Detect, record, and remove uniform and buffer layout directives.
  - Use recorded uniform and buffer locations to associate specific integers locations with WebGL resources.

Preprocesed GLSL Shader

```
/*layout(binding=0, std140)*/ uniform UnityPerDraw {
    vec4 hlslcc_mtx4x4unity_ObjectToWorld[4];
    ...
};
/*layout(binding=1, std140)*/ uniform UnityPerMaterial {
    vec4 _BaseMap_ST;
    ...
};
/*layout(location=0)*/ uniform mediump samplerCube unity_SpecCube0;
...
```

- Because layout directives can be in preprocessor macros, Emscripten does a full preprocessor pass on the shader.
- Binding and location layout directives are recorded and removed from the shader. The resulting shader is compatible with WebGL.
- The program can access the uniforms and buffers by these locations without needing to query them.
- Unity has contributed the WebGL extensions to the open source community as part of the Emscripten project.

Benchmark Test: SRP Batcher Enabled

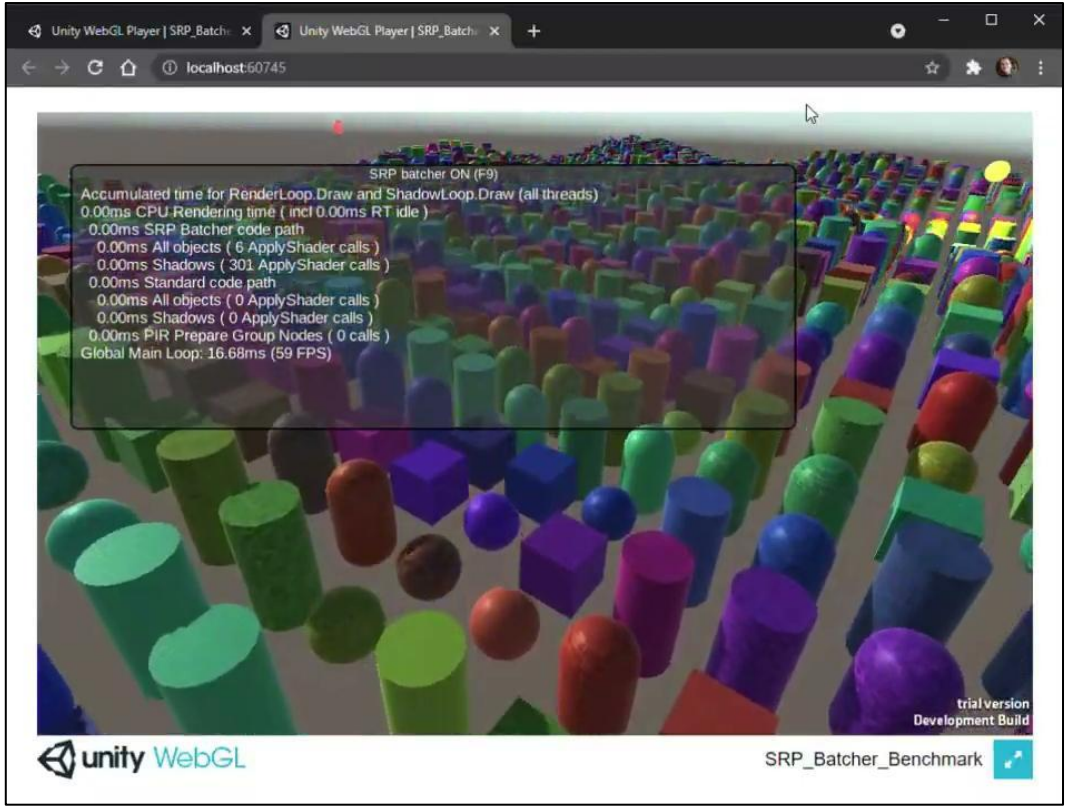
1600 Objects + Materials

4 Realtime Lights

1 Directional Shadow Map

Avg: 16ms / Frame

2x automatic performance gains for worst case scenario



unity

Fast WebGL Rendering With SRP Batcher

- Now Unity WebGL can support optimized rendering with multi-material batching!
- Potentially significant performance gains, reduced CPU overhead, and benefits for mobile.
- Automatic for all compatible shaders.
- **Coming soon to Unity 2021.2**

Thank you.

#unity3d

Special thanks to Jukka Jylänki for open sourcing the Emscripten extensions!

Unity Docs: SRP Batcher
https://docs.unity3d.com/Manual/SRPBatcher.html

Unity Blog: SRP Batcher
https://blog.unity.com/technology/srp-batcher-speed-up-your-rendering

WebGL Extensions For Emscripten
https://github.com/emscripten-core/emscripten/tree/main/docs