

BUNGIE®



Halo Reach Effects Tech

Chris Tchou



BUNGIE



Effects Directions

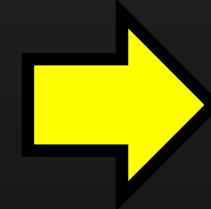
- Atmospheric
- Dense/Visceral
- Pretty
- Easy-To-Use
- Fast



BUNGIE



Covered Techniques



- Cheap Colliding Particles
- 'Shields' & Depth Effects
- Low Resolution Transparents



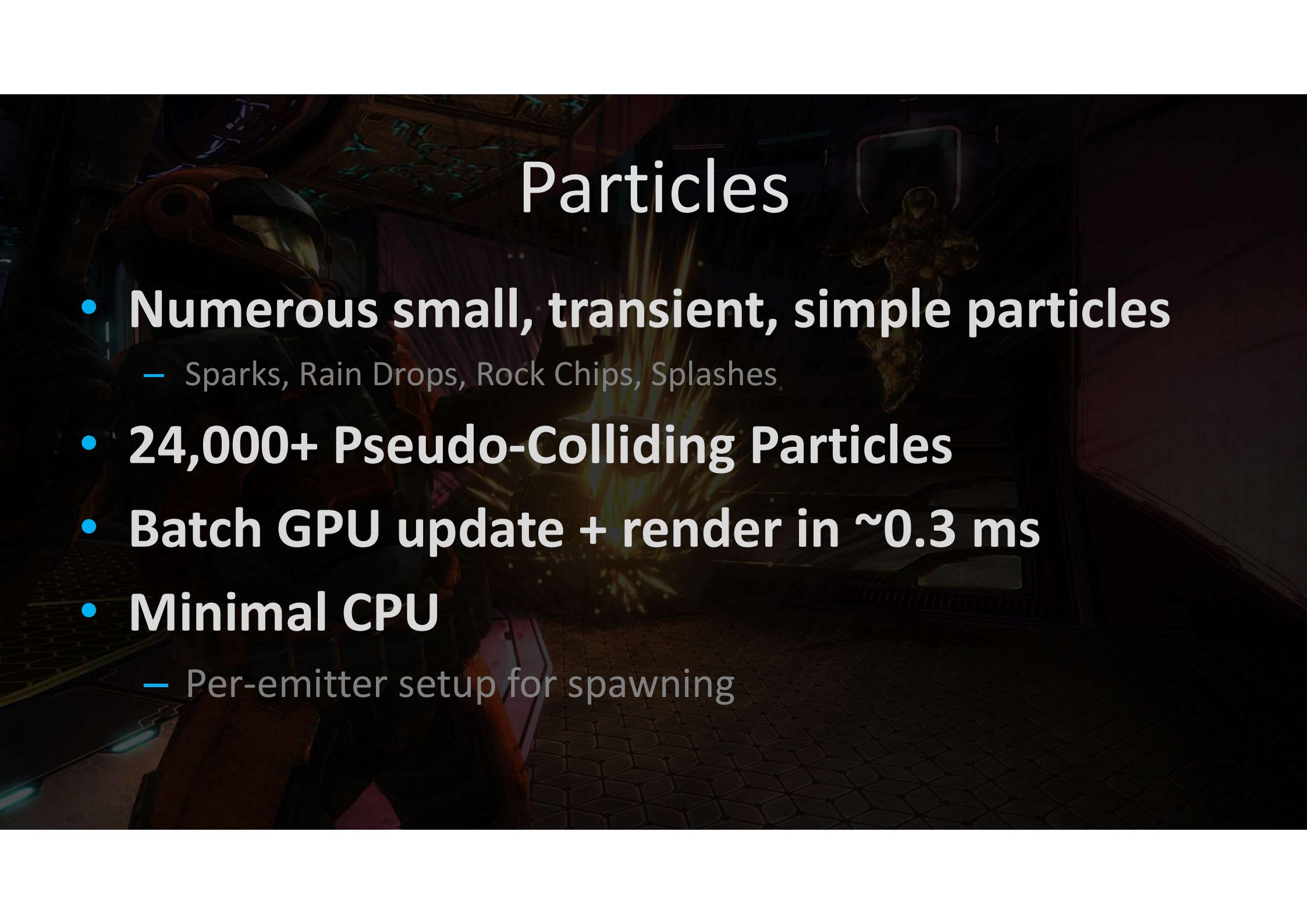
BUNGIE

Particles!







The background of the slide is a dark, atmospheric scene from a video game. It features a character in a futuristic, metallic suit with glowing blue and orange elements. The character is positioned in a dark, industrial-looking environment with pipes and structures. Numerous small, glowing particles are scattered throughout the scene, creating a sense of motion and depth.

Particles

- **Numerous small, transient, simple particles**
 - Sparks, Rain Drops, Rock Chips, Splashes
- **24,000+ Pseudo-Colliding Particles**
- **Batch GPU update + render in ~0.3 ms**
- **Minimal CPU**
 - Per-emitter setup for spawning

Dynamic Particle State

- GPU data (28 bytes/particle)

position.x	position.y	position.z	age
------------	------------	------------	-----

- 32f
- sign(age) signifies 'at rest' vs. 'moving' particles

velocity.x	velocity.y	velocity.z	Δ age
------------	------------	------------	--------------

- 16f



BUNGIE

Dynamic Particle State

- GPU data (28 bytes/particle)

up.x	up.y	brightness	type
------	------	------------	------

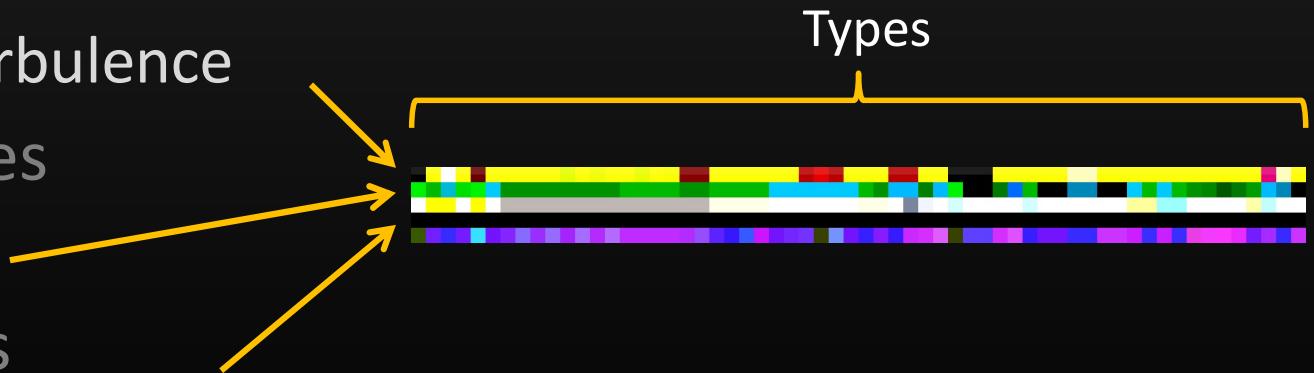
- 8-bit integer
- 2D rotation + scale (as up vector)
- brightness encodes simple illumination
- type points to static particle properties



BUNGIE

Particle Types

- **Static type data stored in a library texture**
 - Physics properties
 - drag, gravity, turbulence
 - Collision properties
 - elasticity
 - Render properties
 - VS: size, orientation (velocity or screen facing), motion blur
 - PS: color tint, texture index



BUNGIE

Particle Render

- Single GPU draw call to render all particles
- Read particle state + type texture
- Create particle card geometry in VS
- Apply render properties in PS



BUNGIE

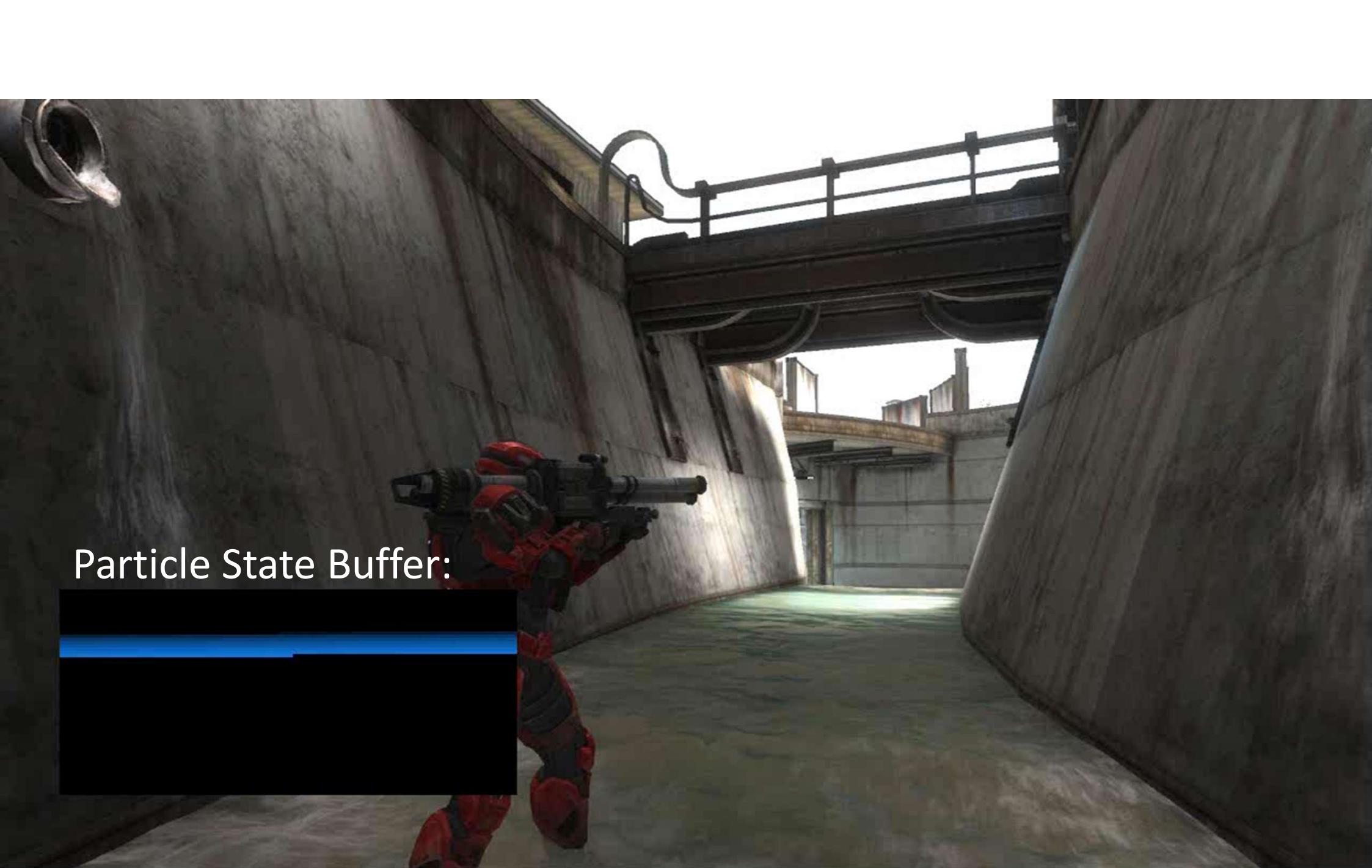
Particle Spawning

- **For each emitter:**
 - CPU determines spawn count
 - CPU issues GPU draw to the dynamic state buffer
- **GPU generates new particle state**
 - Sample randomly from a texture, and apply artist transform
 - Default textures have random values
 - Custom textures can generate complex spawn shapes

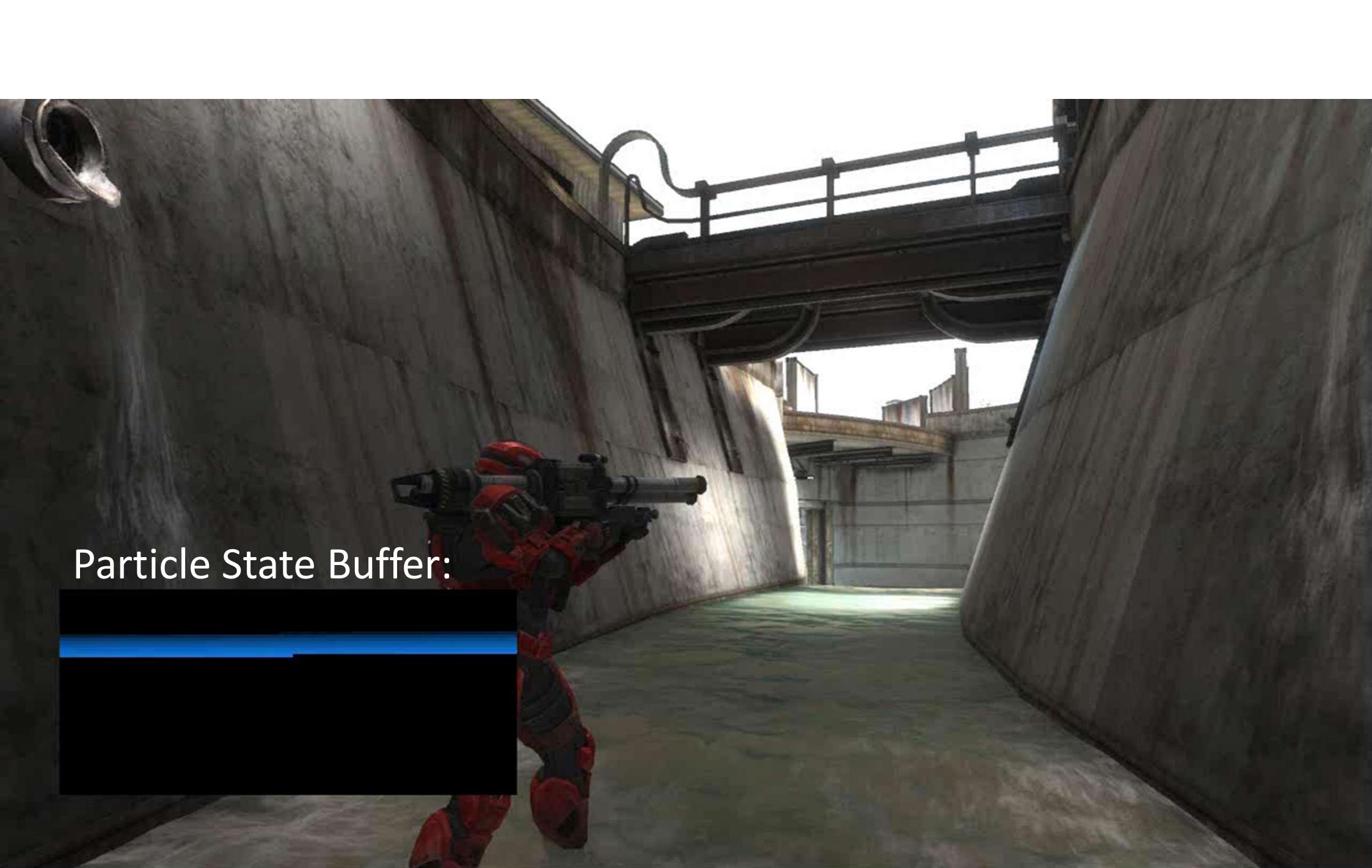


BUNGIE

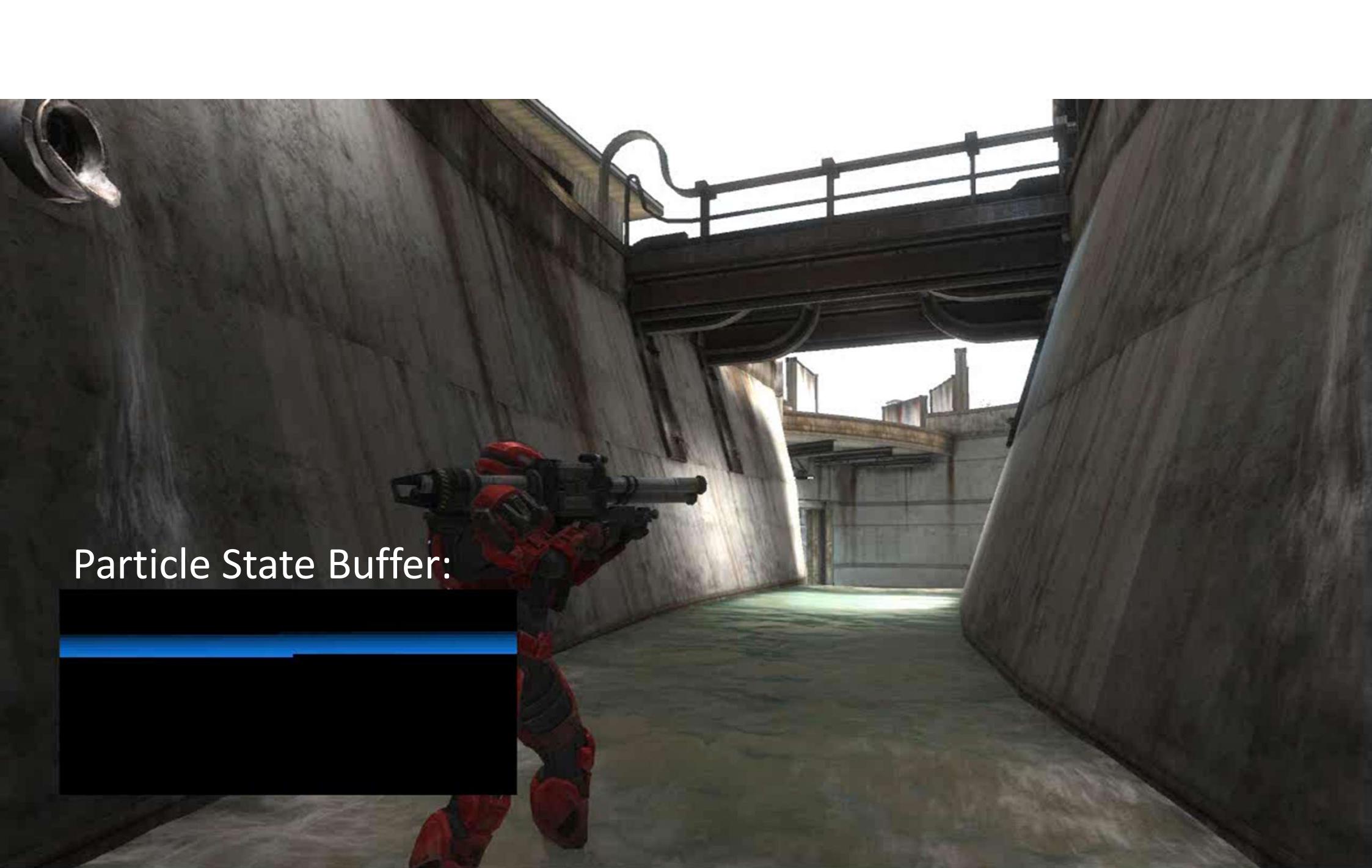
Particle State Buffer:



Particle State Buffer:

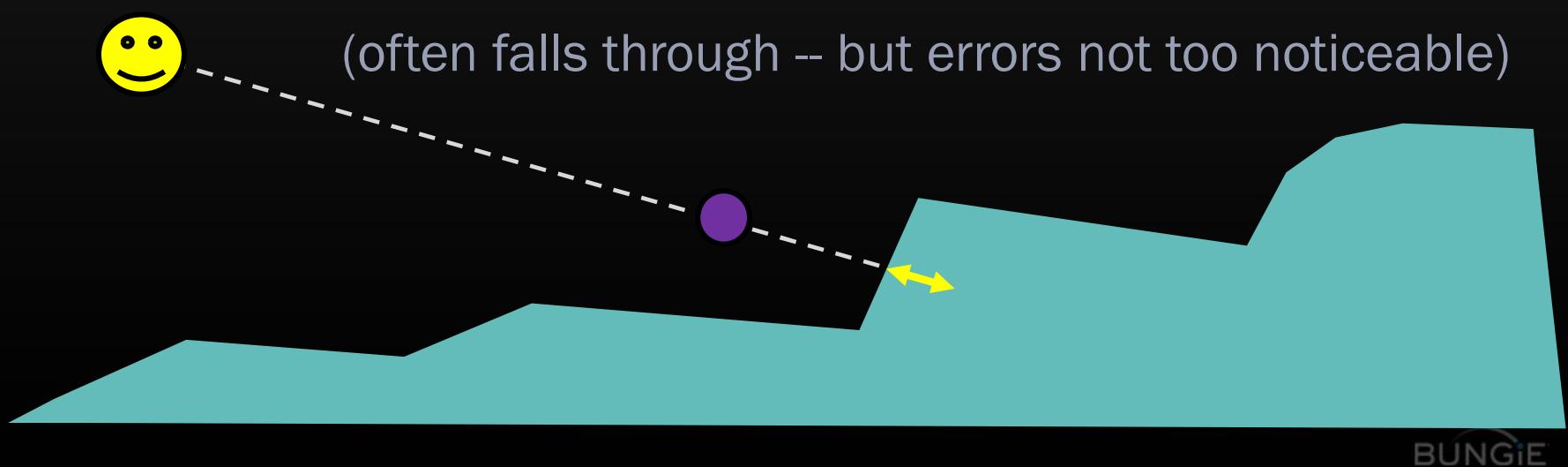


Particle State Buffer:



Particle Collision

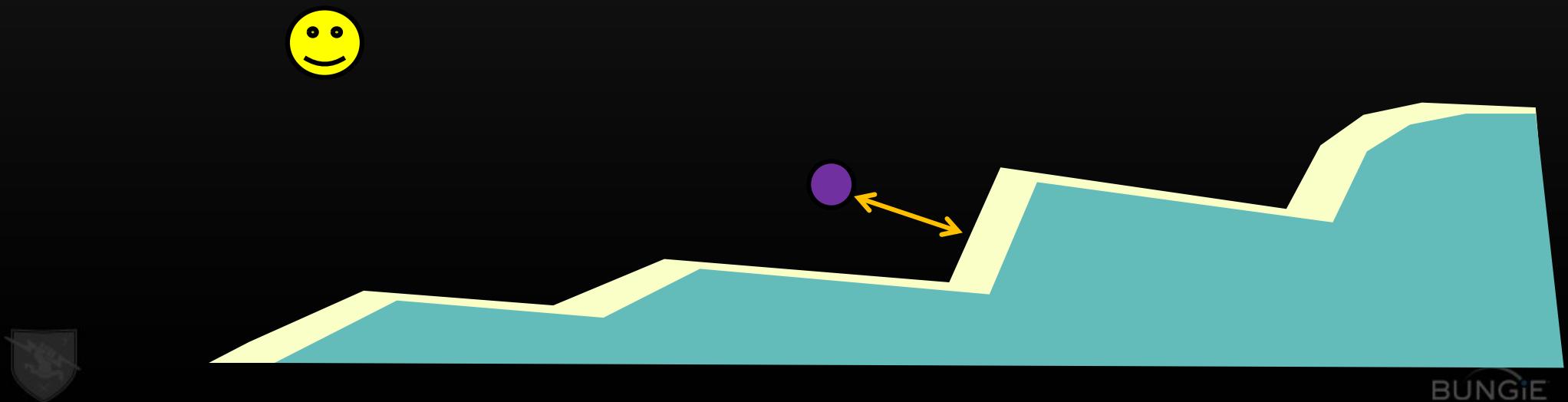
- **Sample depth buffer for local collision**
 - Particles collide in a range behind the depth buffer
 - Only bounces on-screen, against visible surfaces



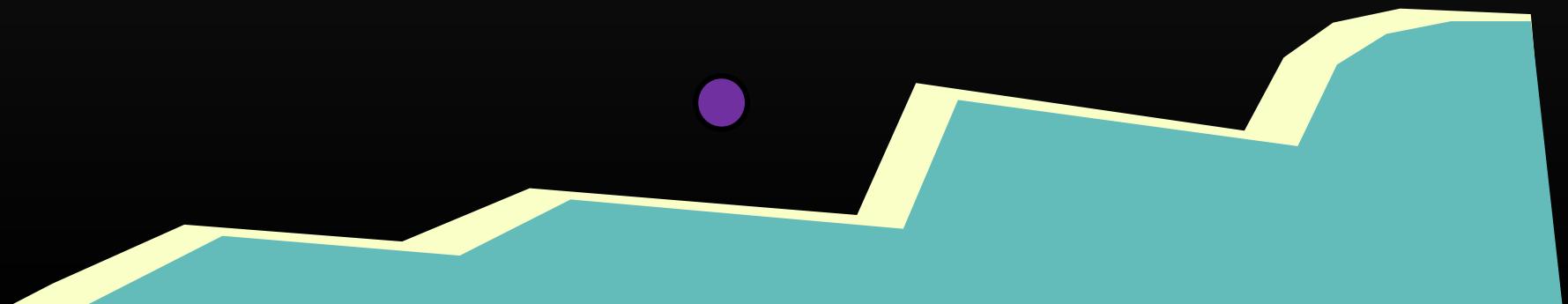
BUNGIE

Particle Collision

- **'At rest' particles don't move**
 - If not close to depth sample, set to 'moving'
 - Allows them to fall if the surface moves



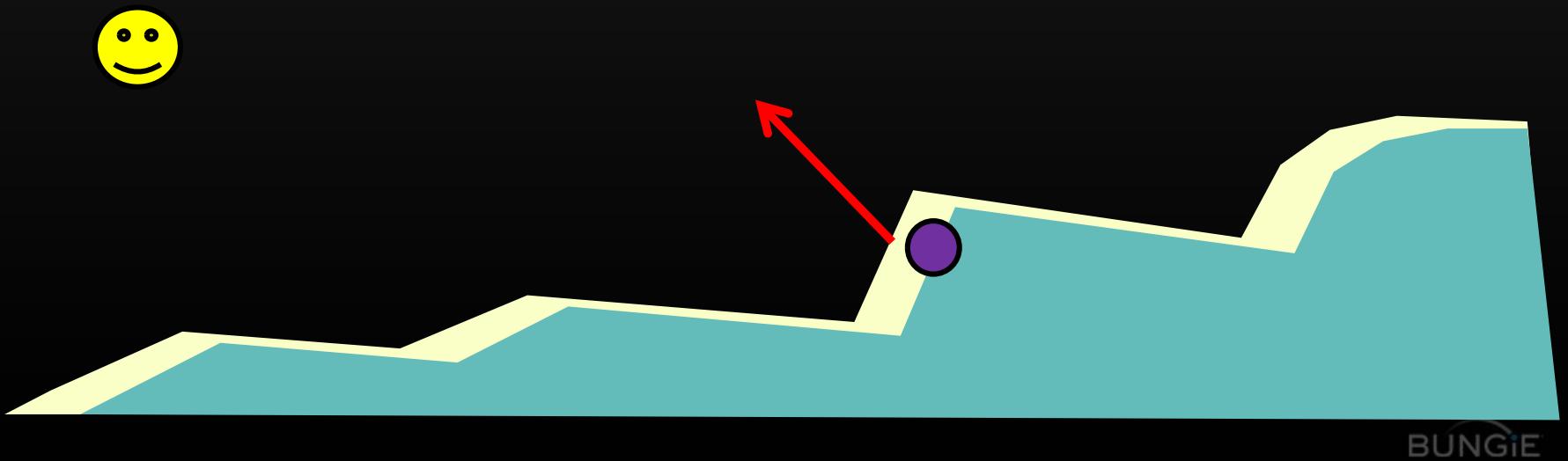
Particle Collision



BUNGIE

Particle Collision

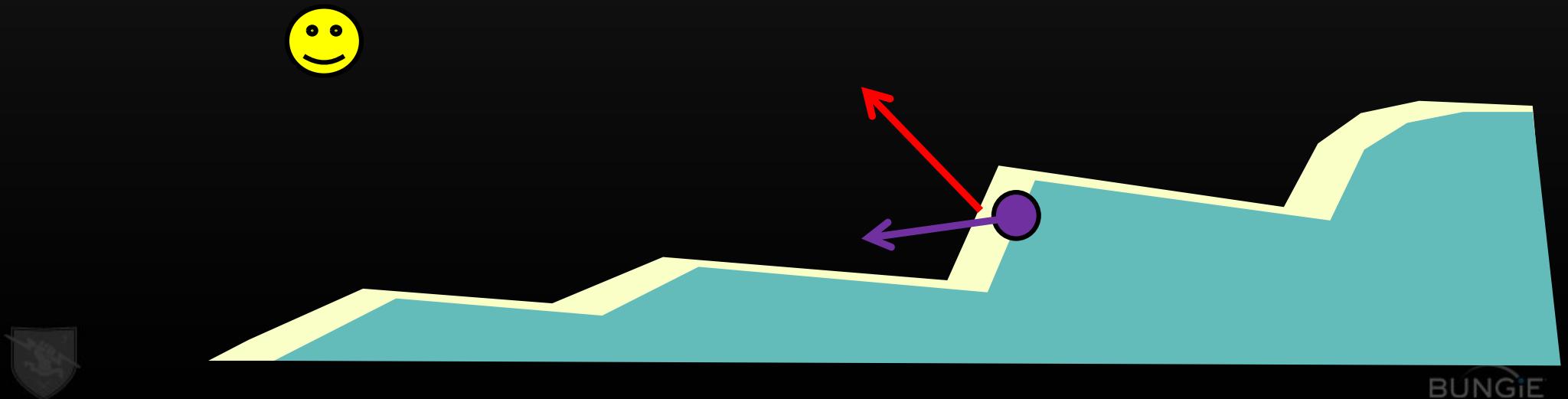
- Sample normal buffer



BUNGIE

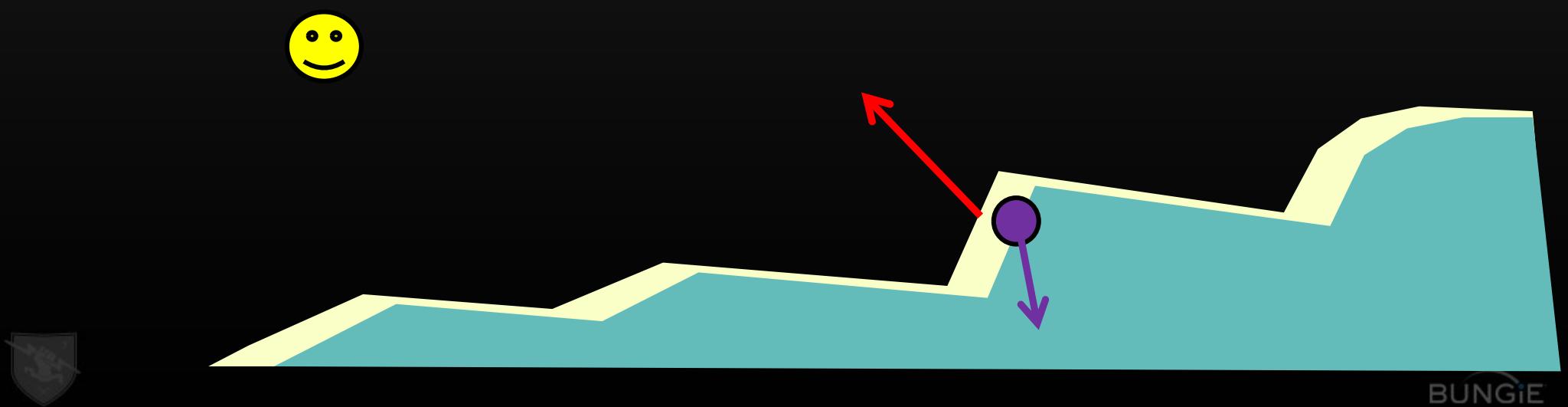
Particle Collision

- Compare normal + particle velocity
- Outgoing particles are destroyed
 - Stops particles from bleeding through visible surfaces



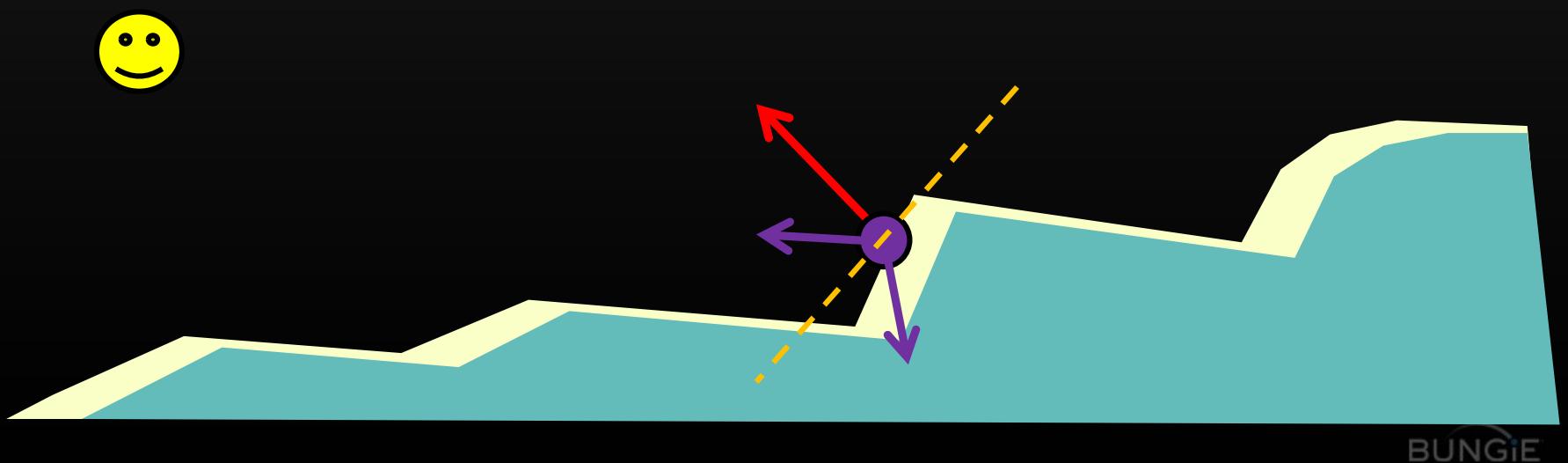
Particle Collision

- Incoming particles are bounded:
 - Ensure particle is not embedded:
 - Re-project position onto surface



Particle Collision

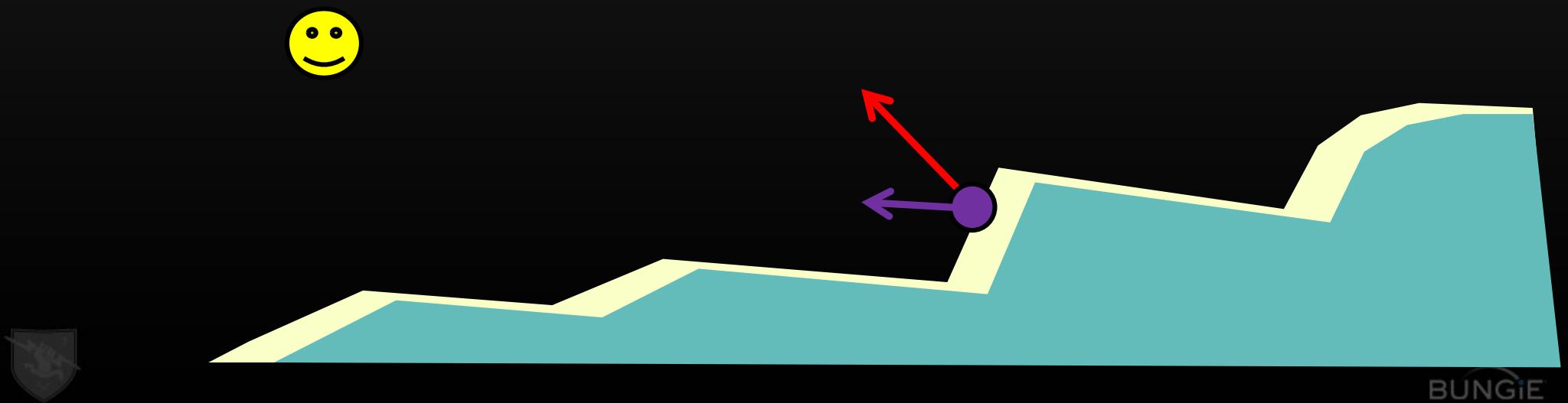
- Incoming particles are bounded:
 - Reflect velocity around normal
 - Reduce velocity by elasticity



BUNGIE

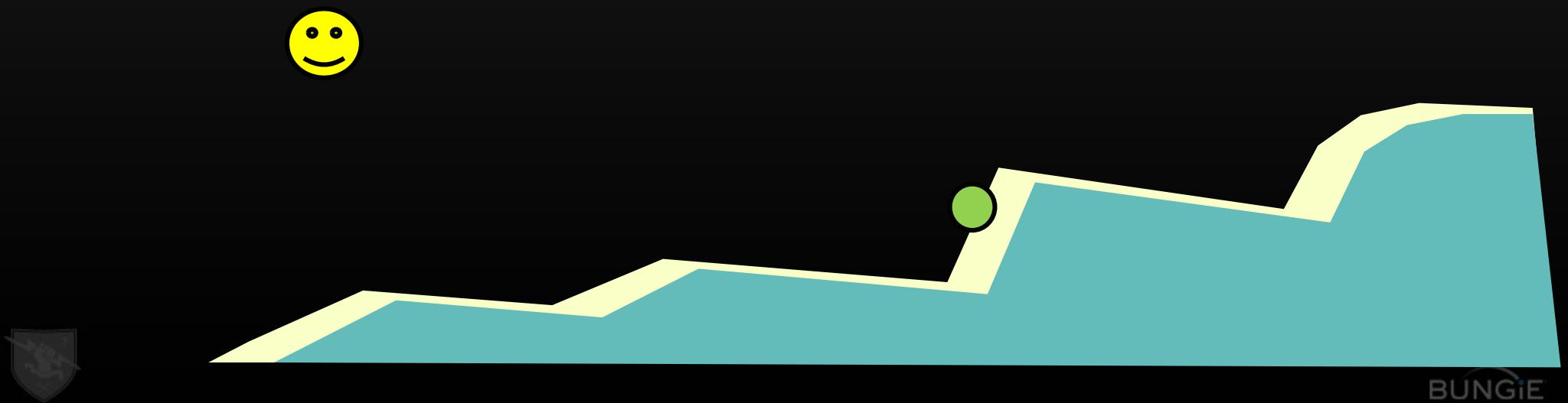
Particle Collision

- Incoming particles are bounded:
 - Switch particle type (optional)
 - If velocity is below a threshold, set ‘at rest’



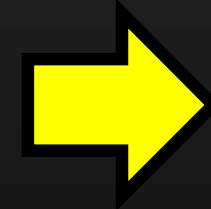
Particle Collision

- Incoming particles are bounded:
 - Switch particle type (optional)
 - If velocity is below a threshold, set ‘at rest’





Covered Techniques

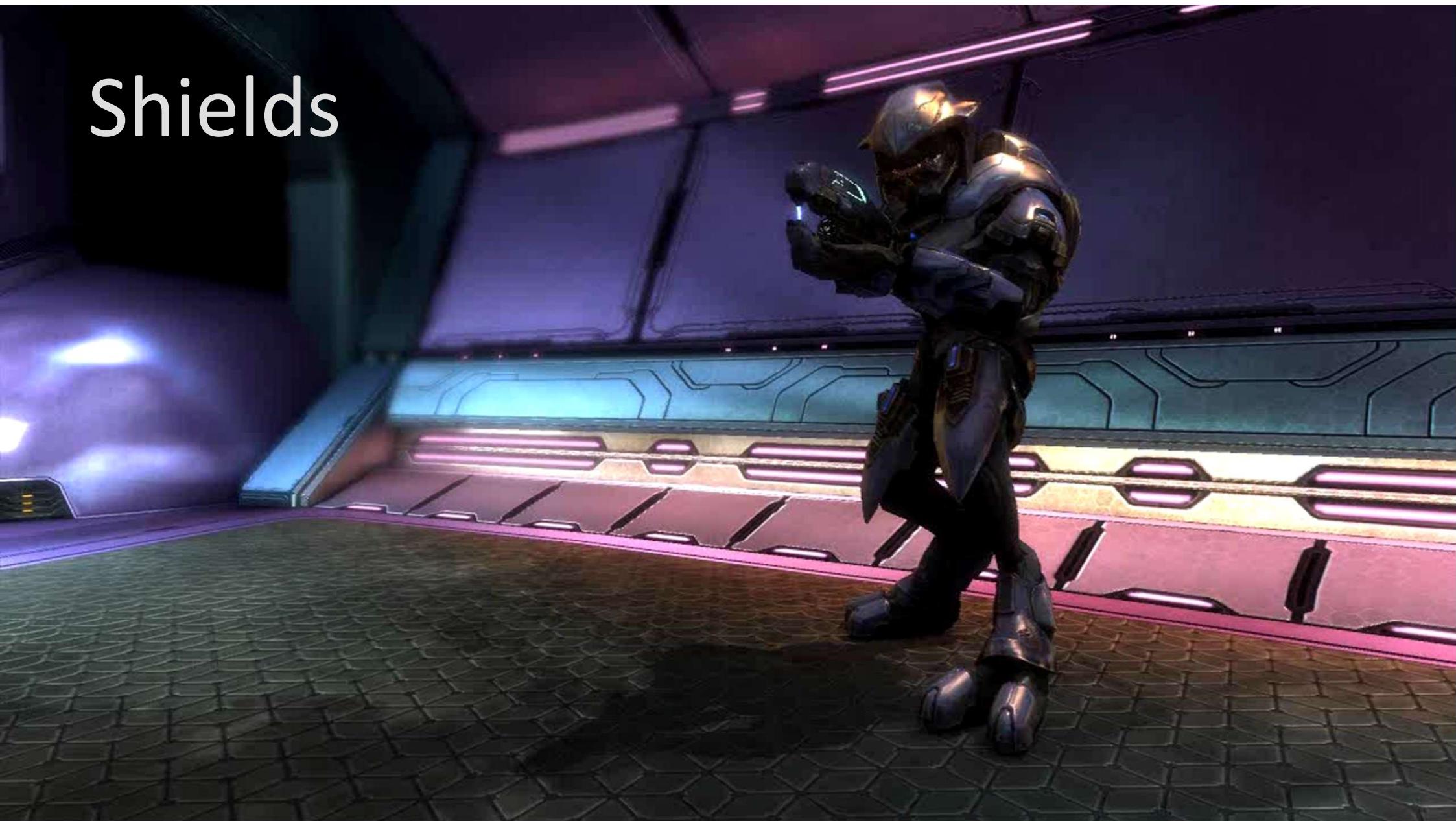


- Cheap Colliding Particles
- 'Shields' & Depth Effects
- Low Resolution Transparents



BUNGIE

Shields



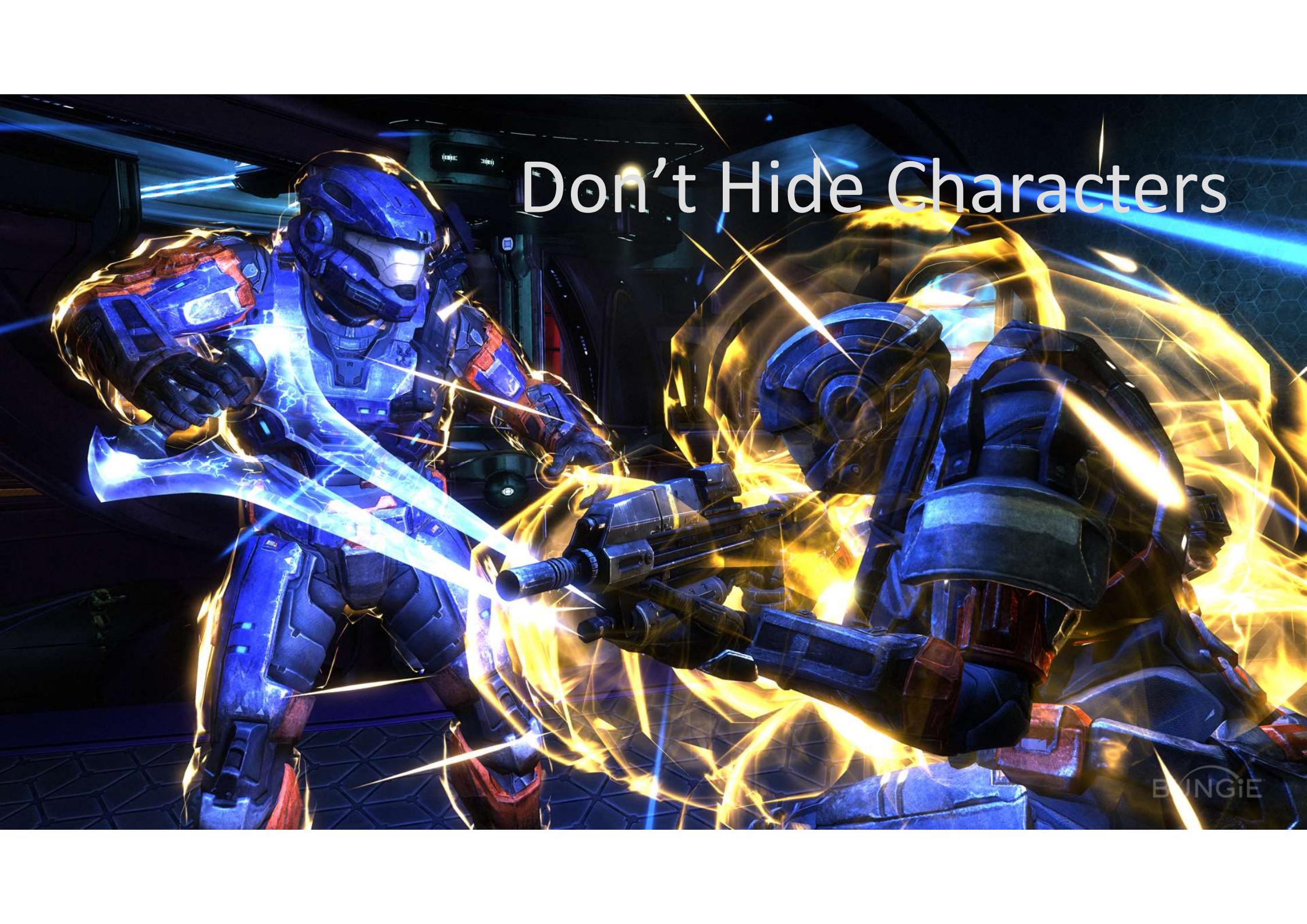
Halo 3



Consistent Look

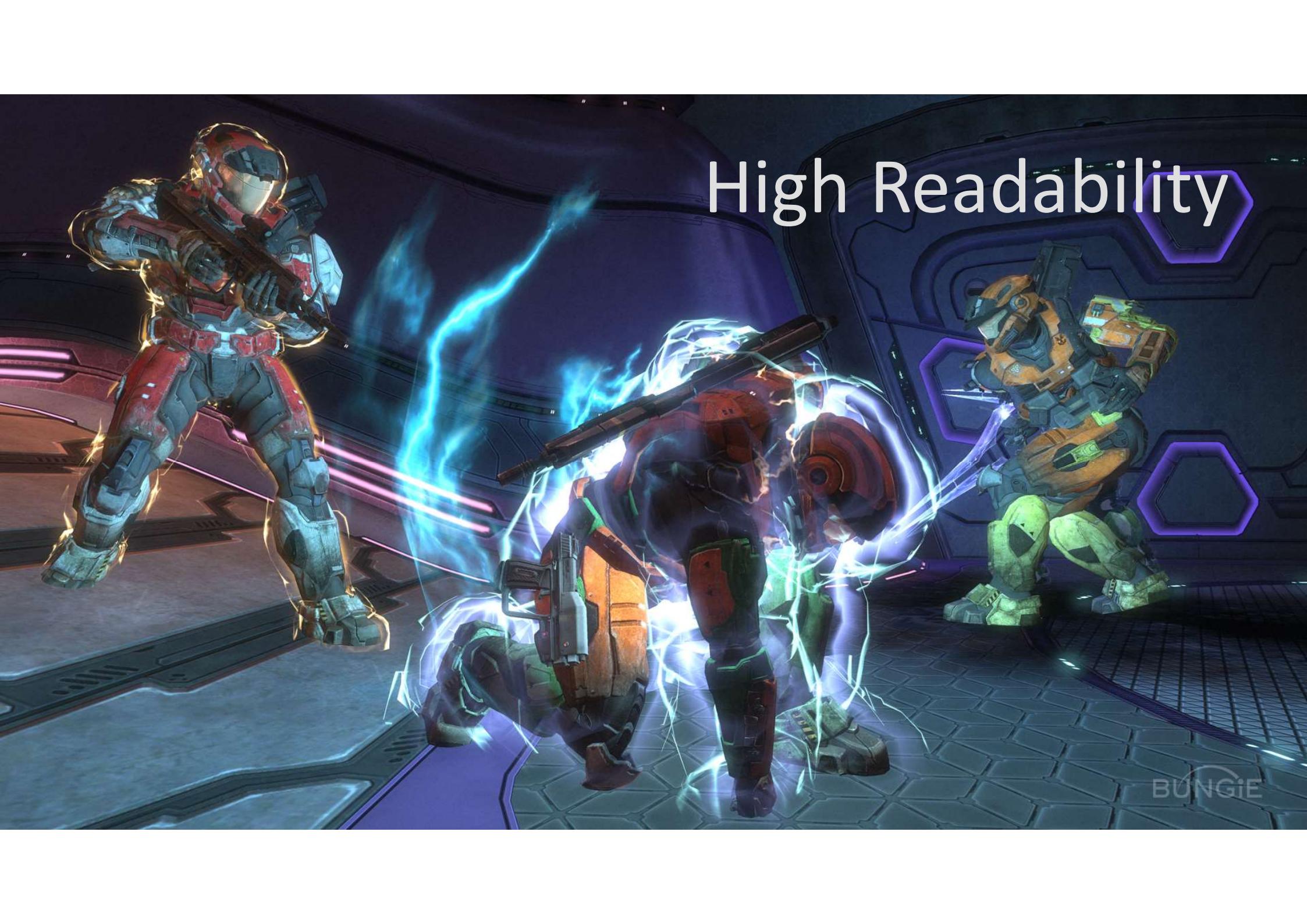


BUNGIE | REACH MULTIPLAYER BETA



Don't Hide Characters

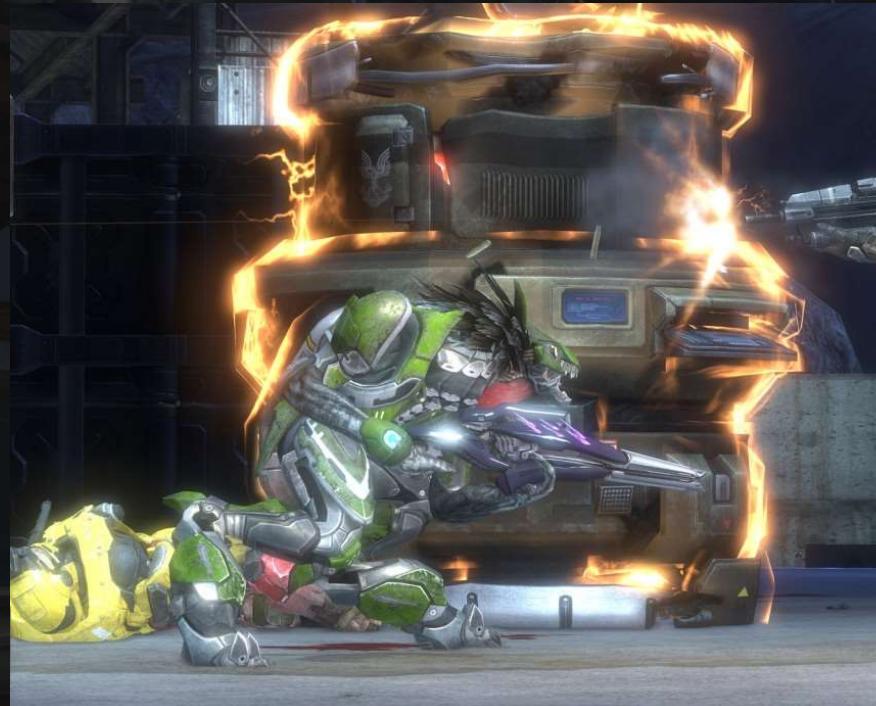
BUNGIE



High Readability

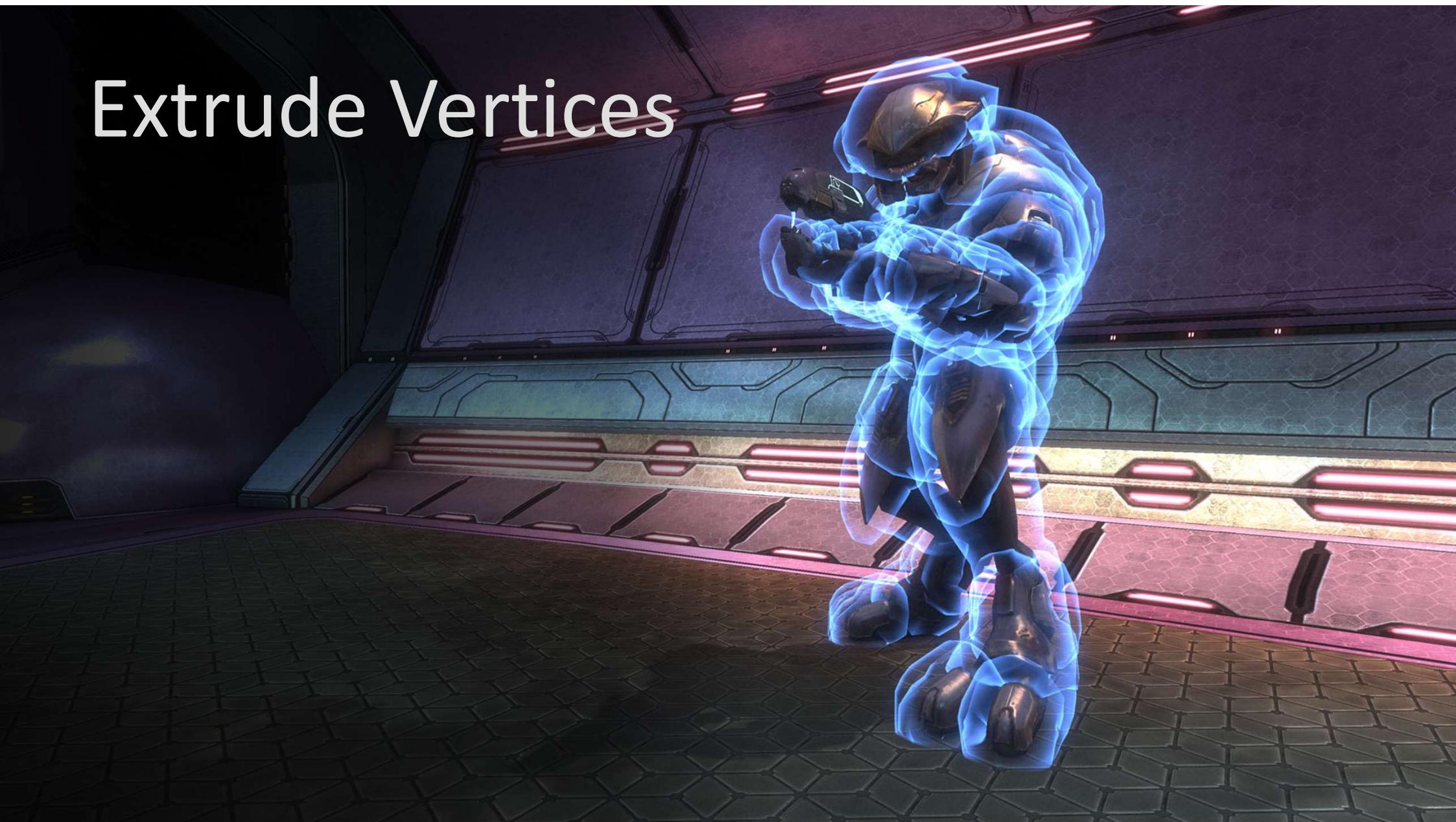
BUNGIE

Minimal Art Overhead

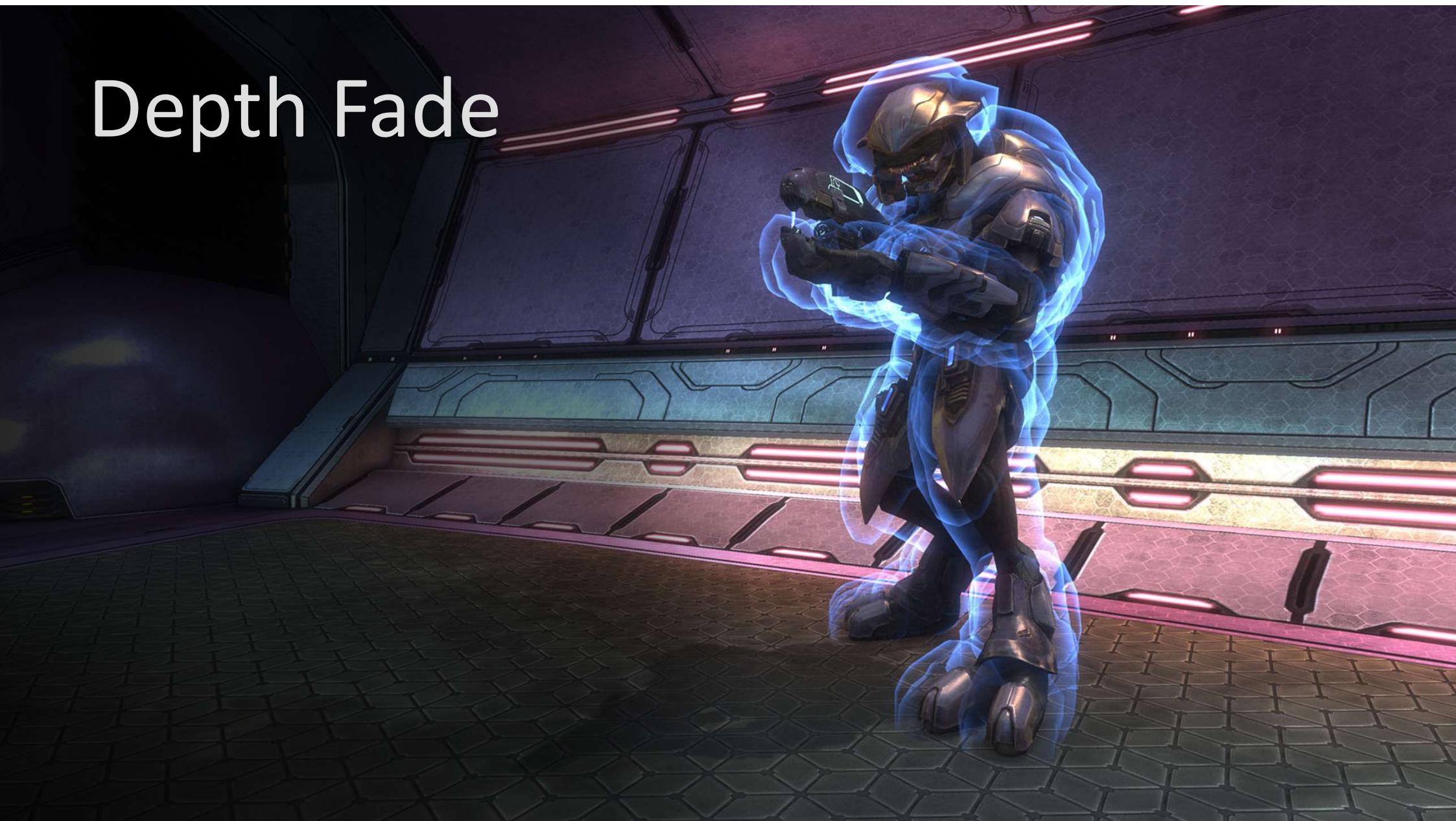


BUNGIE | REACH MULTIPLAYER BETA

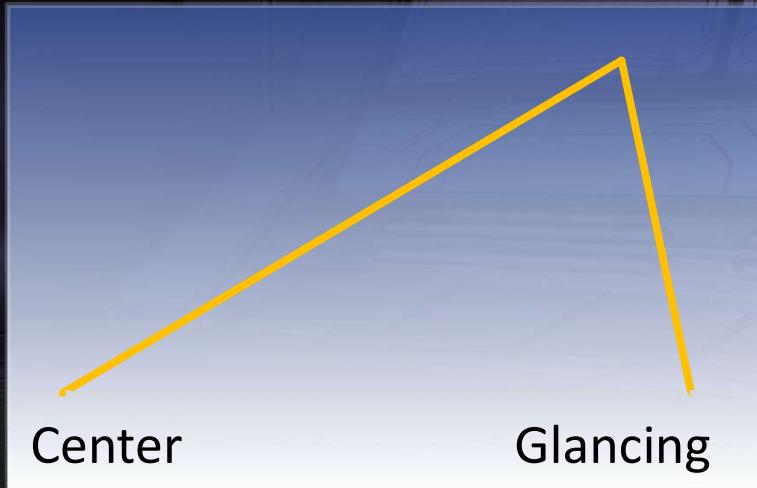
Extrude Vertices



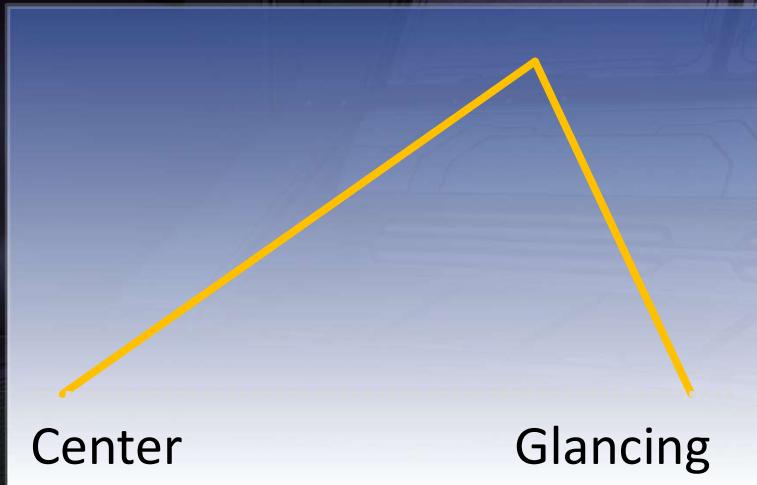
Depth Fade



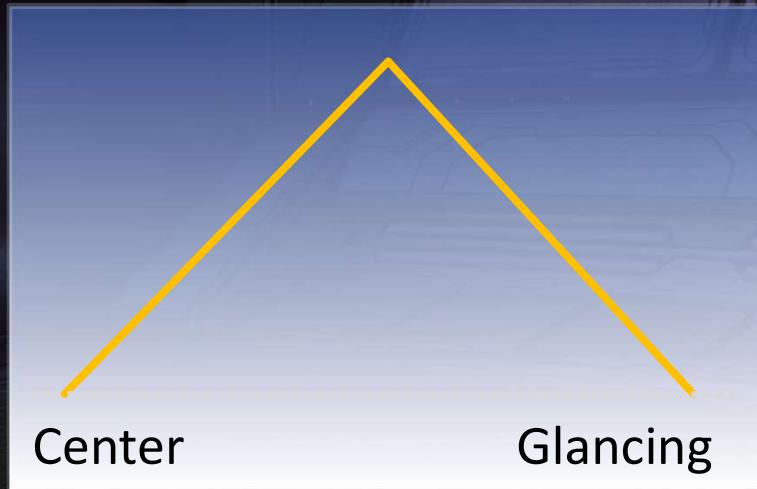
Custom Edge Fade



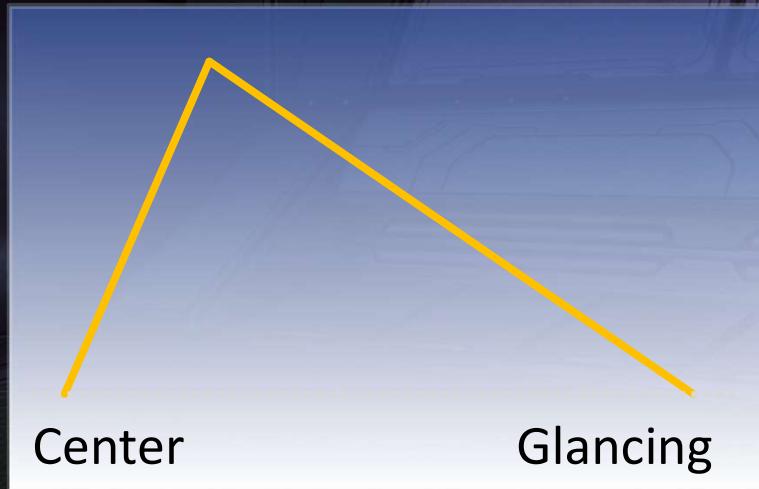
Custom Edge Fade

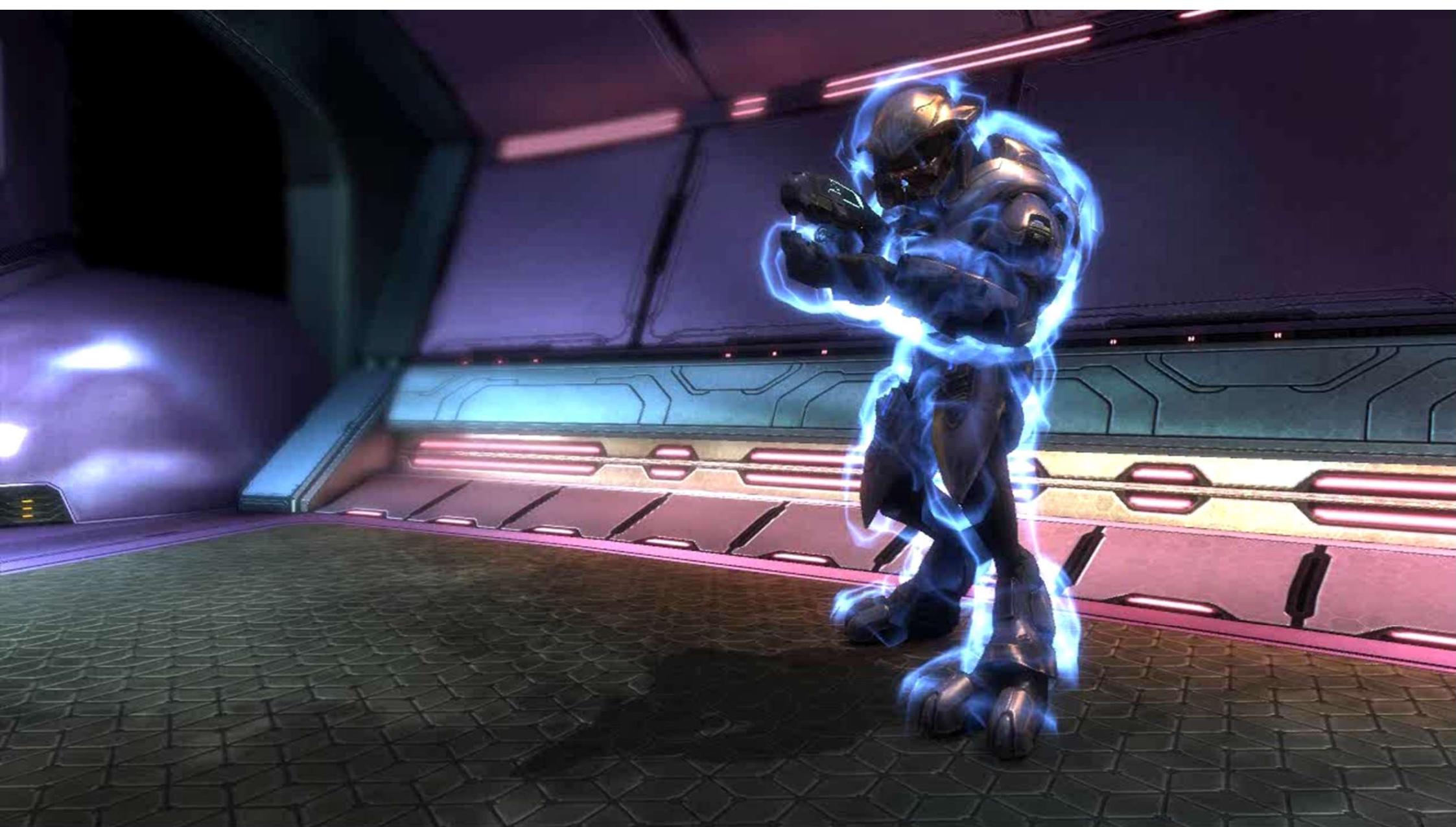


Custom Edge Fade

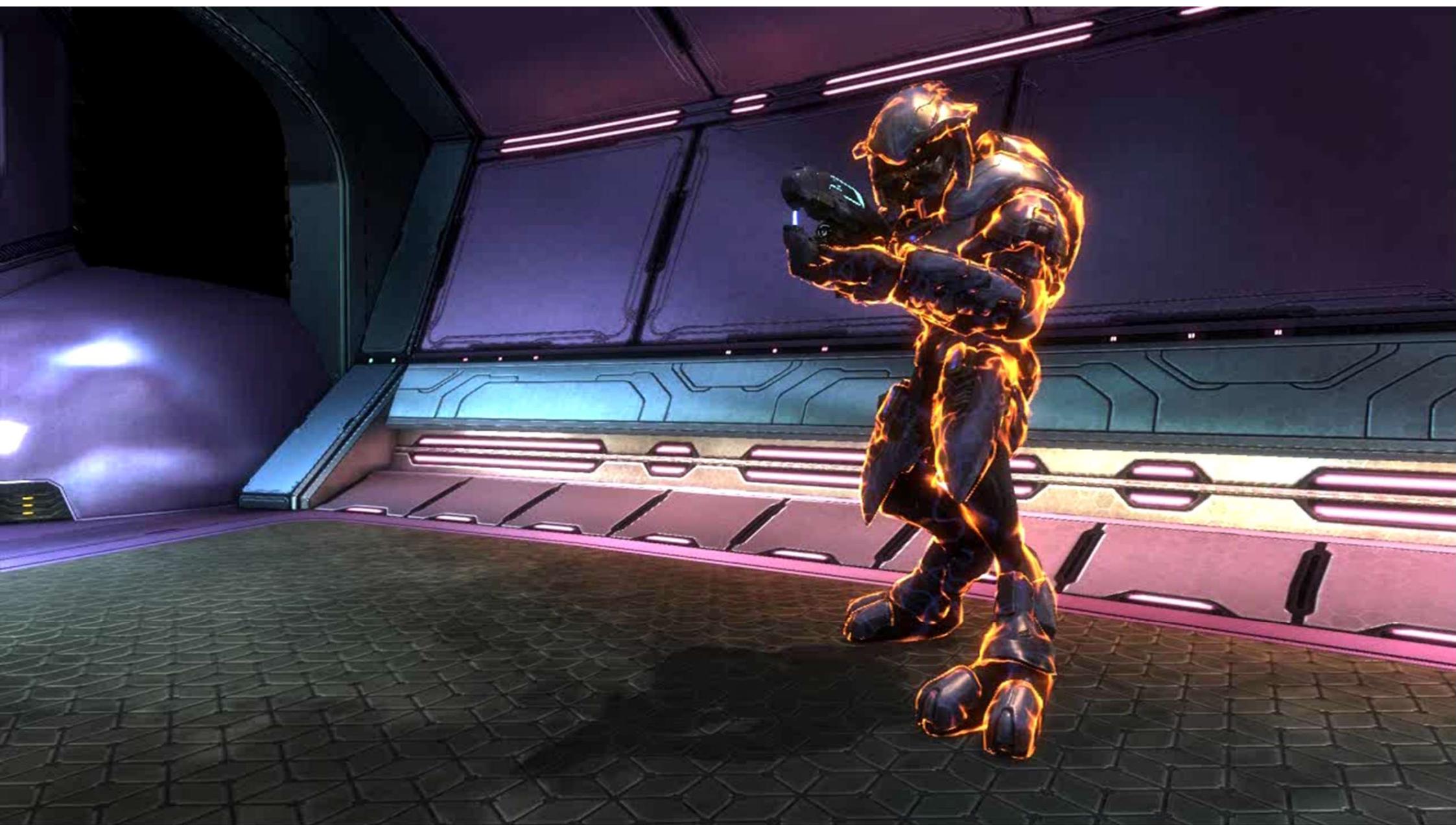


Custom Edge Fade















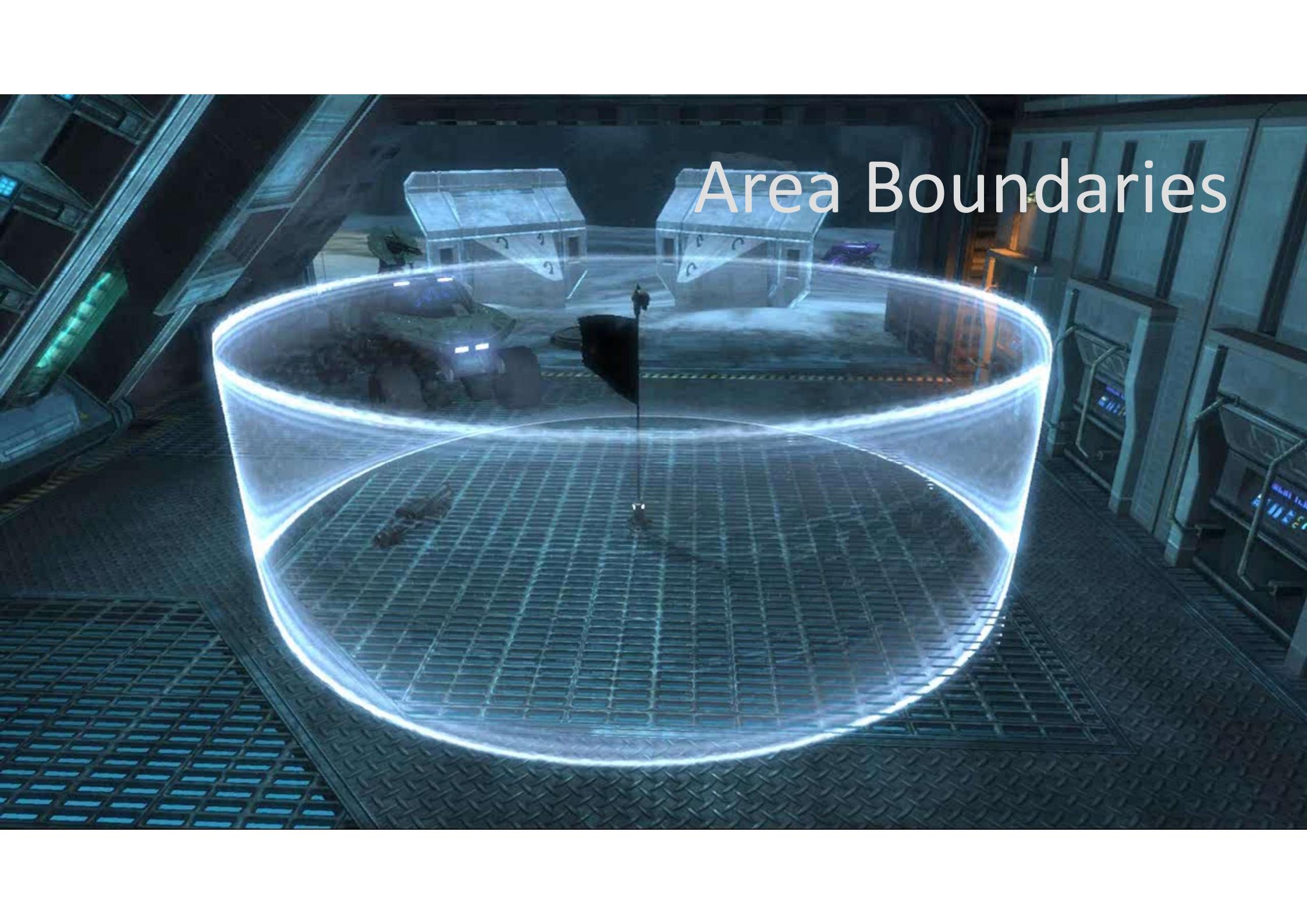
A large, dark grey, futuristic military-style vehicle, possibly an amphibious assault vehicle (AAV), is shown from a low-angle perspective. It has multiple blue plasma energy beams or weapons firing from its front and sides. The vehicle is set against a dark, rocky, and sandy terrain under a dark sky.

Depth Fade Plasma



Shield Doors

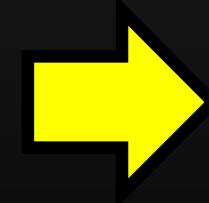


A screenshot from a video game, likely StarCraft II, showing a futuristic industrial facility. In the foreground, a large circular area is outlined by a bright blue energy field. Inside this field, several small, dark units are visible. The ground is made of a dark, textured material, possibly metal or stone. In the background, there are several large, white, cylindrical structures with blue glowing elements on top. The overall atmosphere is dark and industrial.

Area Boundaries

Covered Techniques

- Cheap Colliding Particles
- ‘Shields’ & Depth Effects
- Low Resolution Transparents



BUNGIE

Overdraw Costs

- **Too many layers?**
 - Limited by fill rate & pixel shader
- **Render Fewer Pixels!**
 - Low resolution
 - Alias buffer as 4x MSAA
 - But want smooth upsampling...



BUNGIE

Reach Low Res Transparents

- **Render to separate buffer**
 - 1/2 resolution, 4x MSAA, use existing depth buffer
- **Inline with normal high-res transparency**
- **Low -> High transition: resolve and composite**
 - Smooth bilinear upsampling ☺
 - Transitions are expensive! ☹



BUNGIE

Optimizing Transitions

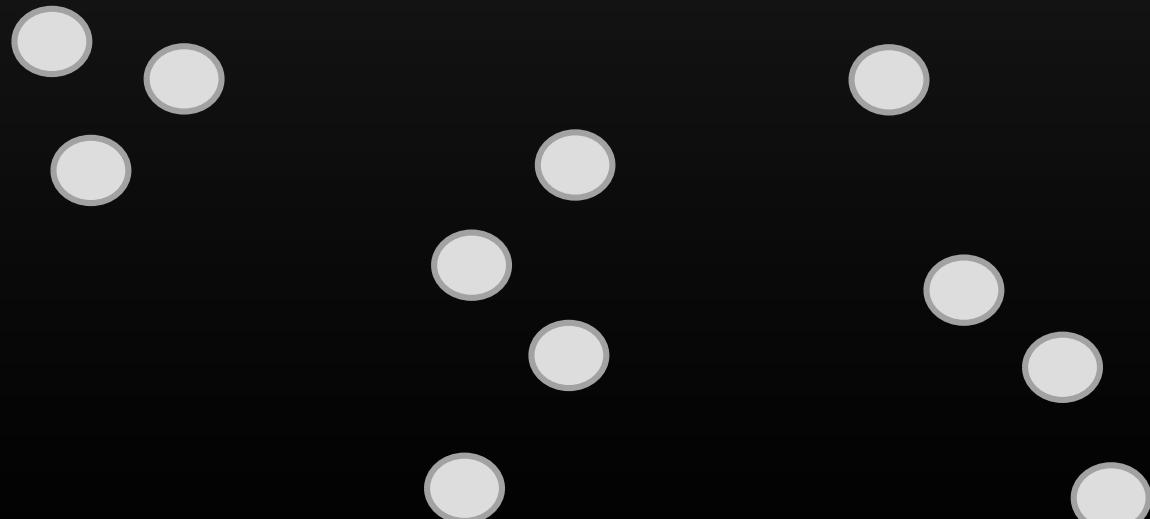
- **Use low-res only when you expect savings**
 - Automatic switch for flagged transparents
- **Stencil touched areas to reduce composite fill**
- **Limit transitions by bucketing during sort**
 - N buckets means at most $N-1$ transitions
 - $N = 3$ worked well



BUNGIE

Well-Behaved Bucketing

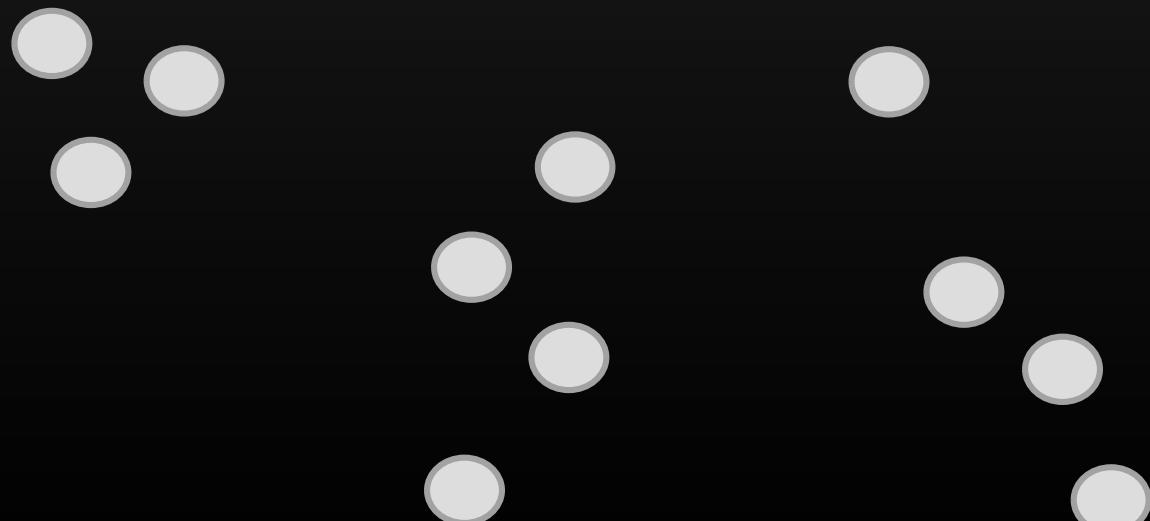
- How to group these to minimize sort pops?



BUNGIE

Best Behaved Bucketing

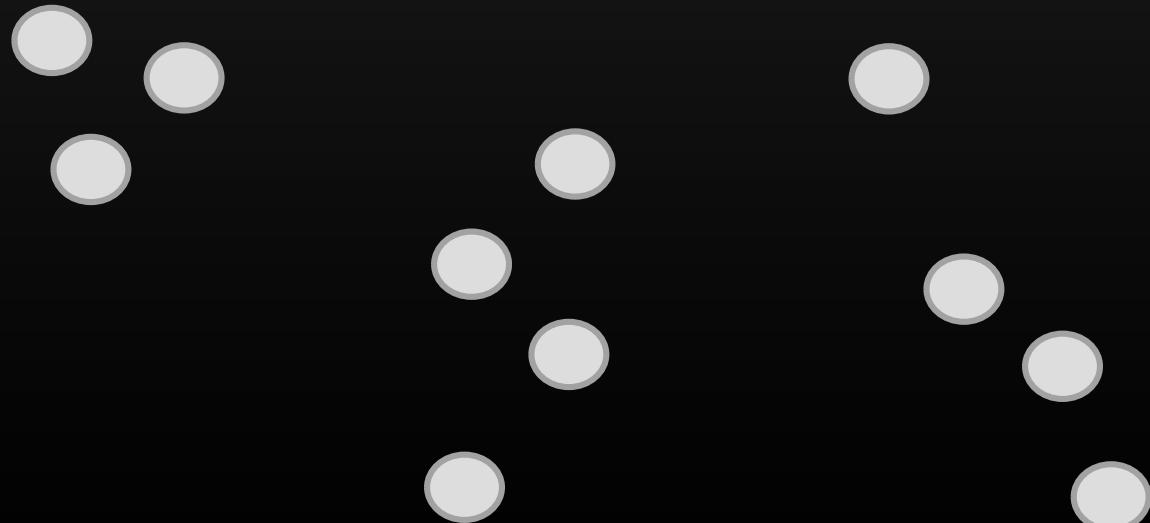
- For N buckets, find $(N-1)$ largest gaps



BUNGIE

Best Behaved Bucketing

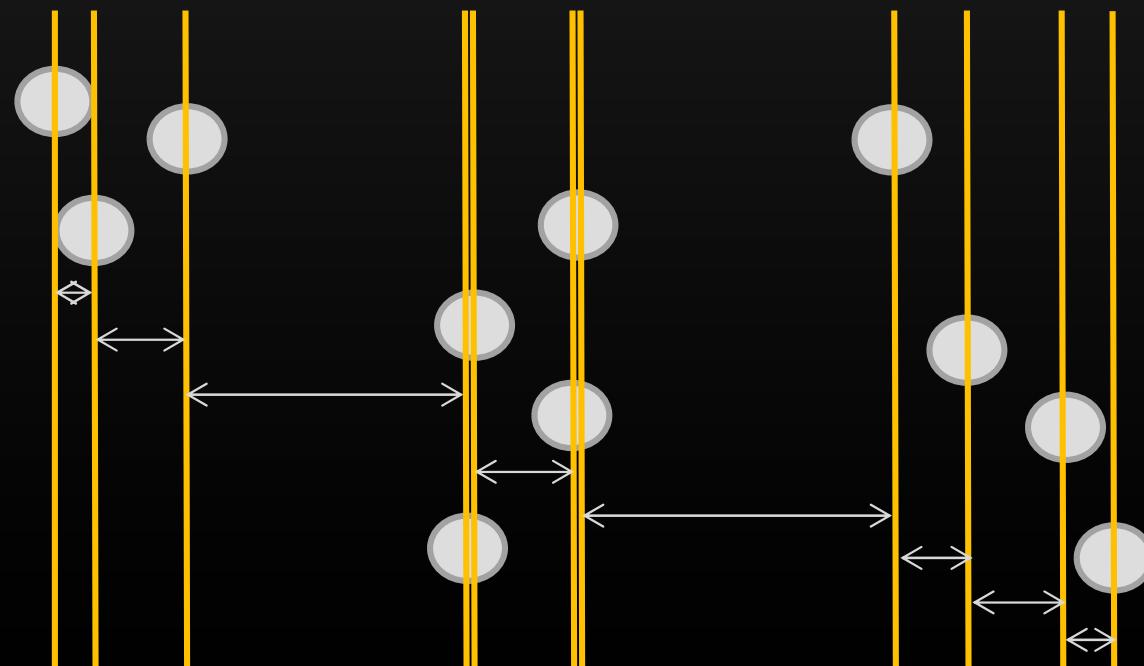
- Sort by Depth



BUNGIE

Best Behaved Bucketing

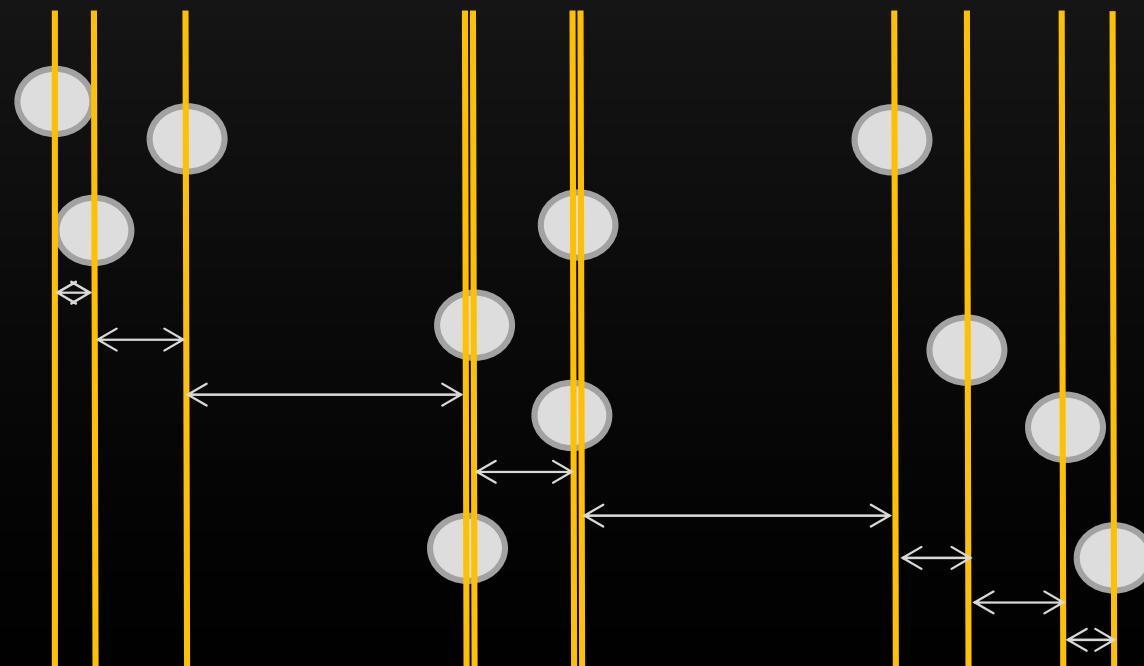
- **Distance between neighbor elements**
 - Use heap to grab $(N-1)$ largest



BUNGIE

Best Behaved Bucketing

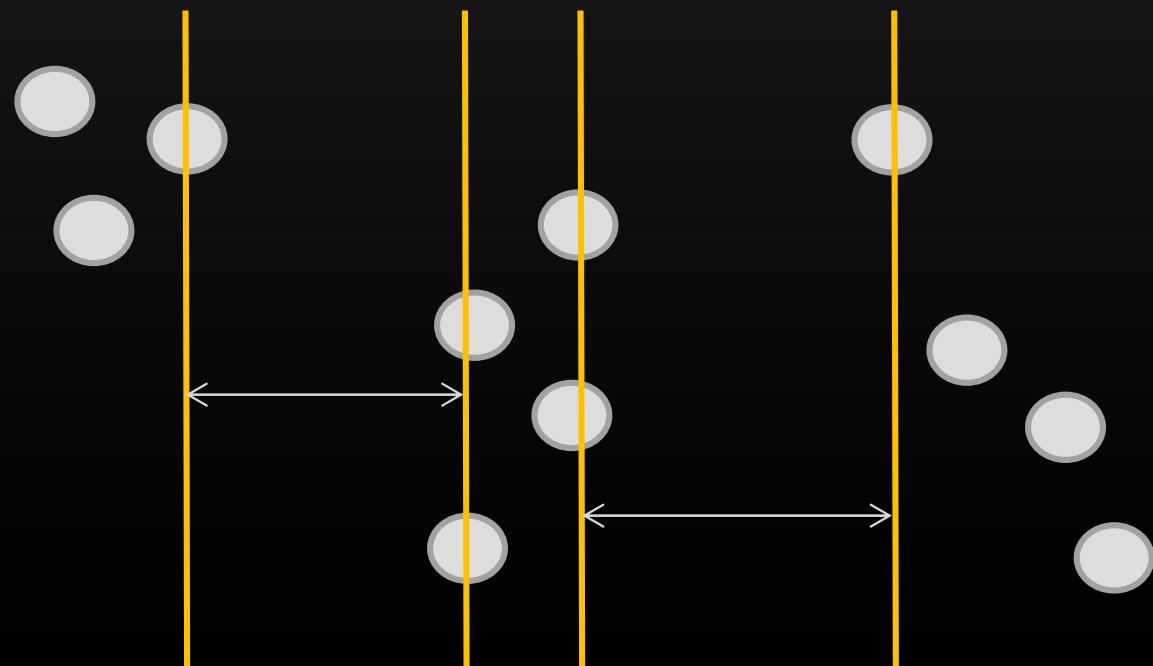
- **Distance between neighbor elements**
 - Use heap to grab $(N-1)$ largest



BUNGIE

Best Behaved Bucketing

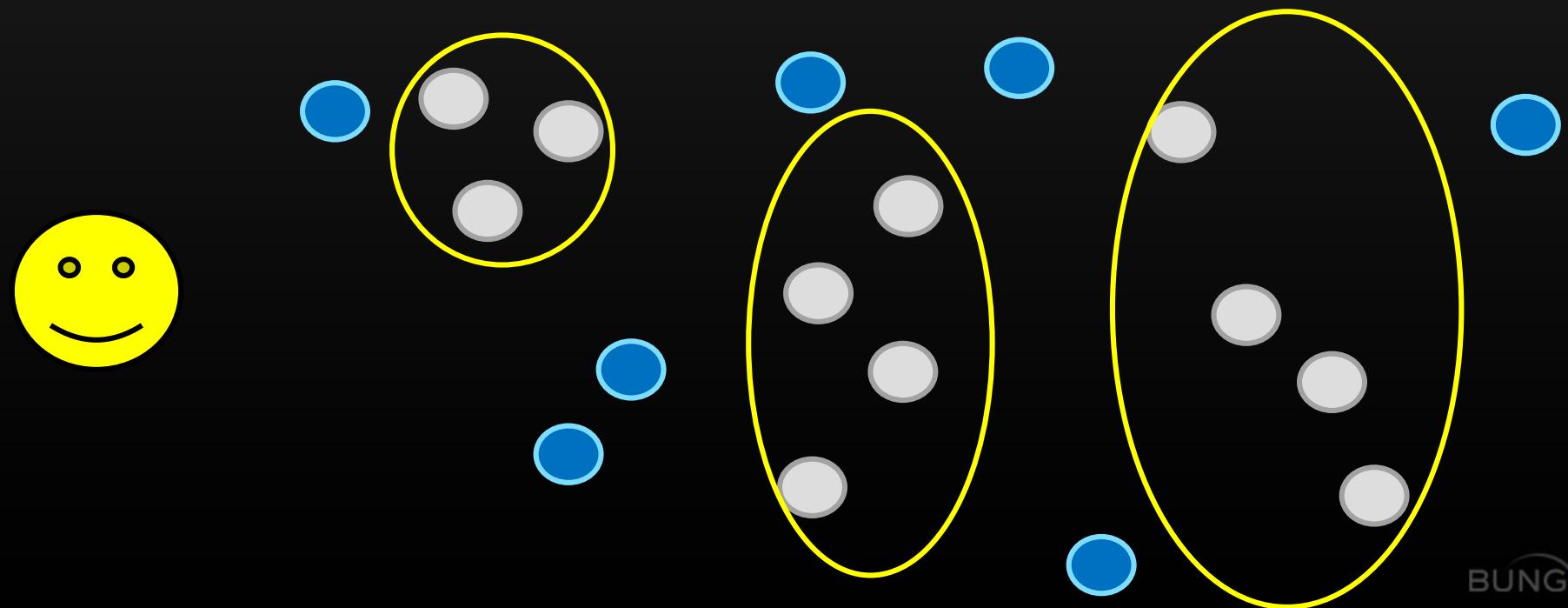
- **Distance between neighbor elements**
 - Use heap to grab $(N-1)$ largest



BUNGIE

Best Behaved Bucketing

- **Divide at these gaps**
 - Render each group together, mixed in with high-rez content



BUNGIE



Covered Techniques

- Cheap Colliding Particles
- ‘Shields’ & Depth Effects
- Low Resolution Transparents



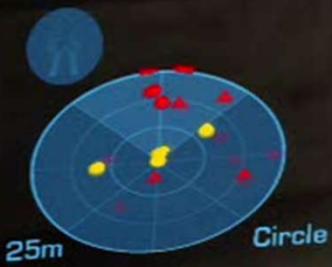
BUNGIE

Summary

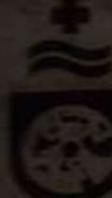
- **Understand the desires of your content guys**
 - Big explosions, smoke, stuff flying everywhere!
- **Figure out where they are limited and fix!**
- **You can often sacrifice correctness for speed**
 - And minimize artifacts with work arounds
- **Depth buffer is incredibly useful**



BUNGIE



02



E SE S SW

6:11
Firefight

01





We're hiring!

[careers@bungie.com](mailto:cCareers@bungie.com)

<http://bungie.net/careers>

Chris Tchou

ctchou@bungie.com

