unity

unity

Certified

Expert

Technical Artist:
Shading & Effects

Exam Objectives

Unity Certified Expert
Technical Artist:
Shading & Effects

The role

The Shading & Effects Technical Artist focuses on delivering the visual intentions behind the game. Artists with skills and competency in shading and effects often work with other Technical Artists and Effects Artists to prepare assets or enhance prepared assets. Shading and Effects artists are responsible for implementing the look, style, theme, and aesthetics of the game.

Artists with these core skills implement real-time and baked lighting, create and customize shaders and rendering systems, and create particle systems and effects that interact with other assets.

Job titles for this role

- Shader Writer
- Lighter
- Effects Artist

Prerequisites

This expert certification is recommended for people who have spent several years in this field and have accrued a variety of advanced, practical application experience, such as:

- Experience in a video game development studio, with at least two shipped titles
- Strong knowledge of physically based lighting techniques and workflows
- Expert level understanding of material authoring for physically based rendering pipelines
- Expert level understanding of color correction and post effects
- Strong knowledge of photographic concepts
- Experience writing shaders in HLSL, CgFX or other shading languages
- Knowledge of scripting/coding using languages such as C++, C#, or Unityscript
- Strong knowledge of particle systems, dynamic simulations, and interchange formats such as Alembic
- Fluency with asset-creation tools such as Adobe Creative Suite, Substance Designer, Substance Painter, Quixel Suite, etc.
- Strong understanding of 2D and 3D math concepts

Core skills

The Shading & Effects Technical Artist certification verifies that candidates have the skills necessary to effectively implement the look, style, theme and aesthetics of the game. Successful candidates will have advanced proficiency in the following areas.

Prototyping

- Create and evaluate material and shader prototypes

Shaders and Materials

- Construct and test custom shaders to:
    - Simulate phenomena
    - Change dynamically in response to gameplay events
    - Extend the functionality of standard shaders to support the look development workflow
    - Implement custom lighting models and non-photorealistic (NPR) looks
- Design, construct, and implement procedural materials and material effects that adapt to scene design and inputs
- Implement custom material UI using ShaderGUI
- Create custom Inspectors using OnInspectorGUI()
- Implement post-effects (e.g., depth of field, color correction, bloom, screen space reflections, motion blur, and fog) to match specific cinematography referenced in GDD
- Script the use of Render Textures to manage real-time reflections

Rendering and Lighting

- Understand the different types of lights and their performance impacts
- Understand the different types of shadows and their performance impacts
- Understand the difference between forward and deferred rendering paths
- Determine rendering API requirements and constraints per platform
- Adapt and extend the rendering pipeline using the Unity API, command buffers, and the Graphics library

Particle Systems

- Simulate atmospheric phenomena using multiple Particle Systems
- Implement typical game effects such as fire, explosion, smoke and water
- Create complex particle effects including Particle System with Sub-Emitters, Line and Trail Renderers
- Script Particle System events to occur during gameplay in response to player and NPC behavior and other runtime events
- Import and render externally generated simulation data
- Dynamically assess Collider and Transform data to implement interactions with Particle Systems

Performance and Optimization

- Understand the target platform specifications and limitations
- Optimize shaders, Particle Systems, post effects, lighting, fog, shadows, etc., to run on target platform
- Understand when to use optimization techniques and problem-solve (billboarding, alpha sorting issues, draw calls, fill-rate issues, CPU/GPU bound scenarios) where necessary
- Analyze and evaluate rendering issues with the Frame Debugger and platform-specific frame-capture tools

Certification Exam Topics

Tooling and Pipeline

- Asset customization
- Process improvement through custom tools and Editor customization

Rendering

- Render pipeline
- Post-processing effects
- Cameras in Unity

Shaders

- Shader construction, prototyping and customization
- Render Setup shader knowledge
- Scripting knowledge pertaining to shaders

Particles and Effects

- Particle System customization and extension
- Effects techniques

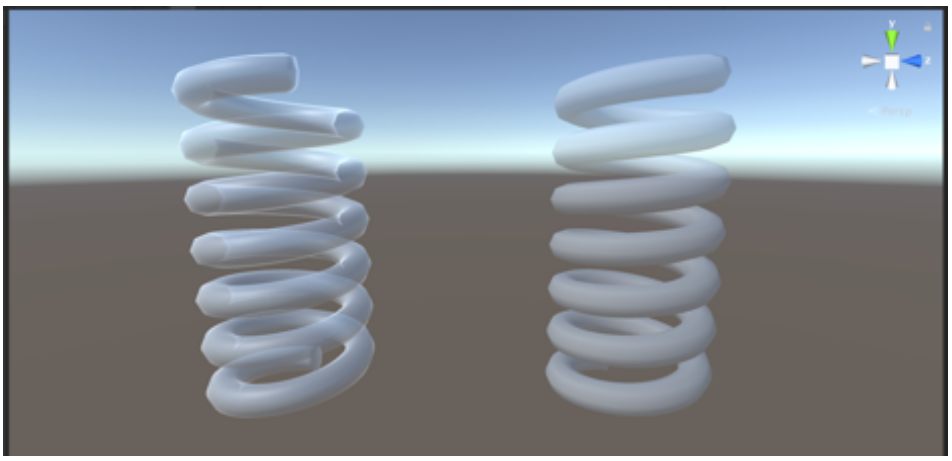
Performance

- Scene optimization

Sample Questions

Question 1

Refer to the exhibit:



A Standard Shader with the Rendering Mode set to Transparent has been applied to a single mesh. Some of the faces overlap and have normals that face in the same direction. The model on the left is rendering incorrectly. The concept art calls for an effect that produces the image on the right.

How should a Technical Artist achieve this effect?

A Use a custom Culling Mode.

B Use a custom render queue.

C Write a custom multi-pass shader.

D Write a custom shader using GrabPass.

Question 2

The Game Design Document (GDD) for a mobile strategy game with a minimum platform support of OpenGL ES 3.0 specifies a flat map with procedurally placed gold mines spawned at runtime. The gold mines are partially underground and partially above ground.

What is the most efficient way to reveal the entrances into the gold mines?

A Procedurally generate the map mesh at runtime.

B Use a Compute Shader on the map.

C Use a Parallax Shader on the map.

D Use a shader with a stencil mask.

Question 3

A first-person adventure game is set in a limited geographic area of mountain meadows with dynamic lighting. A Technical Artist needs to add a procedural cloud system over the procedural sky color, and add persistent, unreachable mountains in the distance. The mountains must be affected by the dynamic light and fog in the scene.

How should the Technical Artist avoid interactions between the mountains and the clouds?

A Add the mountains using CameraEvent.BeforeSkybox.

B Add the mountains into the scene as low resolution geometry and Levels of Detail (LODs).

C Add the mountains using CameraEvent.AfterEverything.

D Add the mountains as World Space UI Images, with each set on their own Canvas.

Question 4

A Technical Artist needs to emulate rainfall on a globe. Data is retrieved from weather servers in realtime. The weather data displays as colored rainfall images that line up with the globe's regular diffuse textures. A function is provided to sample a location on the globe to get the intensity of the rainfall. The Game Design Document (GDD) calls for the rainfall to be emitted as falling rain drops, similar to a TV news weather map.

What is the most efficient way for the Technical Artist to pass the rainfall distribution and intensity data to an Emitter during runtime?

A

1. Use a Shape module set to emit downwards from a scaled instance of the globe mesh.
2. Apply a rainfall texture to a particle material so that particles in area of no rainfall are invisible or clipped.

B

1. In a scripted update, get random points on the visible area of the globe and retrieve the rainfall data.
2. Apply this data to style particles in an emit function.

C

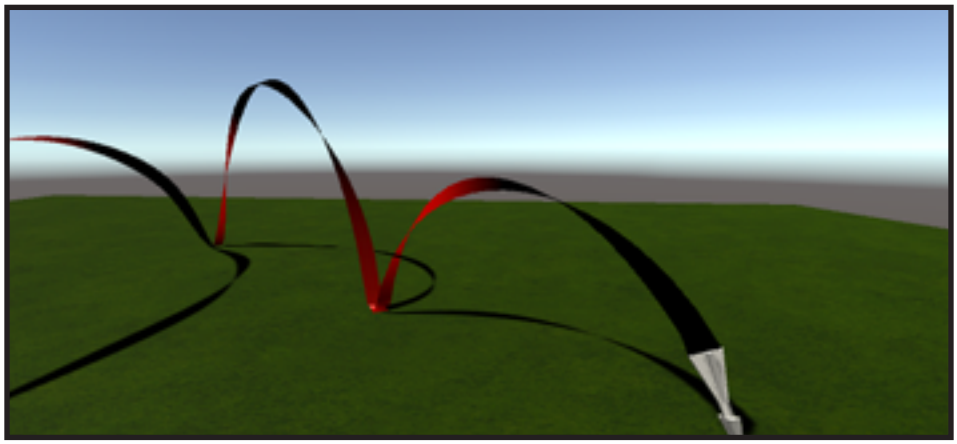
1. Use a Shape module set to emit downwards from a sphere.
2. Apply a rainfall texture to a particle material so that particles in area of no rainfall are invisible or clipped.

D

1. In a scripted update, get a list of active particles from a spherical emitter.
2. Use their positions to retrieve rainfall data, remove particles in areas of no rain, and update the intensity of valid rain particles.

Question 5

Refer to the exhibit:



A game project includes a player-controlled weapon that fires a bouncing projectile that spins on axis and leaves a twisting ribbon that displays a trail. As shown in the exhibit, the ribbon must react correctly to the scene lighting.

What is the most efficient solution to create the trail effect?

A Create a Trail Renderer on a GameObject parented to the projectile.

B Create a Line Renderer driven by a script, on a GameObject parented to the projectile.

C Create a script using the Mesh class to implement the effect on the projectile.

D Create a script to drive a rigged ribbon model with a SkinnedMeshRenderer.

Correct Answers: C, D, A, B, C