unity

unity

Certified
Expert
Technical Artist:
Rigging &
Animation

Exam Objectives

Unity Certified Expert
Technical Artist:
Rigging & Animation

The role

The Rigging & Animation Technical Artist serves as a vital bridge between game artists and programmers, delivering assets for systematic integration into games while respecting artistic vision and platform limitations. The Technical Artist's assets are flexible enough to adapt to the changing needs of apps and play experiences, while also responding to evolving user needs. The Artists' in-depth Rigging and Animation skills help improve the game's production pipeline as they assist with animation, game development, and character creation.

Additionally, the game-development team depends on the Rigging & Animation Technical Artist's core skills to script complex animations and GameObjects, including audio and animation. The artists also create and support pipeline tools – optimizing these assets for various platforms – and ensure that they adhere to the game's Technical Design Document (TDD).

Job titles for this role

- Technical Artist
- Rigger
- Technical Animator
- Technical Character Designer

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
- Knowledge of scripting/coding using languages such as C++, C#, or Unityscript
- Experience in the full game lifecycle, working from early concept through launch
- Experience with rigging/character setup and animation
- Familiarity with digital content creation (DCC) scripting languages such as Python, MEL, and MaxScript
- Understanding of game animation pipelines, including character and environment setups
- Solid organization skills regarding file structures, naming conventions and established protocols
- Fluency with asset-creation tools such as Adobe Creative Suite, Substance Designer, Substance Painter, Quixel Suite, Autodesk Maya and 3ds Max, Pixologic ZBrush, Motion Builder, etc.

Core skills

The Expert Technical Artist: Rigging & Animation certification verifies that candidates have the skills necessary to effectively integrate rigging and animation assets into a game. Successful candidates will have advanced proficiency in the following areas.

Prototyping

- Evaluate a Game Design Document (GDD) to determine animation-focused Editor tools that will enable the rest of the design team to construct the game and “stay on design”
- Create and evaluate prototypes to develop best practices and refine the Technical Design Document (TDD) per platform performance specifications
- Evaluate and recommend technical solutions for animation and rigging problems

Pipelining

- Customize and automate the importing of assets
- Procedurally modify and vary attributes of GameObjects
- Control parameters on multiple GameObjects
- Implement behaviors and animations procedurally

Preparing GameObjects

- Prepare asset Prefabs with levels of detail (LODs) for game implementation
- Implement and map Humanoid and Generic Rig animation types
- Rig and script complex assemblies using Joints, Cloth, Rigidbodies, and physics components for placement as Prefabs
- Create and test custom Physic materials to facilitate gameplay
- Evaluate and optimize compound and Mesh Colliders, and Physic Materials

Preparing Animations

- Create Blend Trees in State Machines
- Script complex (multiple layers with active states), high-performance behaviors in State Machines
- Create State Machine layers to layer animation
- Create states and behaviors to weigh and transition Blend Shapes
- Set up frame-based audio/FX in animation clips

Performance and Optimization

- Understand the target platform specifications and limitations
- Understand the differences between an FK and IK rig and their relative performance impacts
- Test and optimize complex assemblies per platform requirements
- Evaluate scene performance and identify bottlenecks using the profiler
- Evaluate CPU and GPU optimization with respect to rig complexity, batching, and vertex shader methods

Certification Exam Topics

Tooling and Pipeline

- Editor customization
- Asset customization
- Process automation using custom tools

Prototyping

- Rigging and animation prototyping

Animation and Rigging

- Configuration of animation system state machines and animation events
- Rig setup and animation
- Physics components for dynamic animation

Performance

- Scene optimization

Sample Questions

Question 1

A Game Design Document (GDD) for a combat game specifies that the Non-Player Characters (NPCs) will have a range of animations including idle, run, attack, and defend. The NPCs will all be armed with and use two-handed swords in combat.

The GDD specifies that the NPCs will have different poses determined by a pose Layer to vary their appearance in the game. The Brute pose animation clip has the shoulders of the character forward, and the back hunched. The Hero pose animation clip has the character posing proudly, with shoulders back and chest expanded. When the game is tested, the hands on the NPCs do **NOT** line up on the handle of the sword once the pose Layers are applied.

What is a solution to this problem?

A Use OnAnimatorIK() to set the position, rotation, and related weights to move the hands to the weapon.

B For each of the Avatars, set the Per-Muscle Settings for the chest to move the hands appropriately.

C Use a StateMachineBehaviour that overrides the OnStateMove() and call animator.MatchTarget() to adjust the hand positions.

D For each of the Avatars, set additional Optional Bones to allow the hands to hold the weapon from closer or farther away.

Question 2

A Game Design Document (GDD) has the following requirements:

- A weapon should be able to switch hands.
- The weapon is parented to a joint named PropWeapon.
- PropWeapon is a child of the root of the character.

In the Digital Content Creation (DCC) package, the Animation team used a constraint on the PropWeapon joint to animate the weapon switching hands. The animations have been baked on every frame and exported using FBX. The Technical Artist notices that when the game is played in Editor, the weapon has a jittery movement that was **NOT** present in the DCC package.

What is the cause of the jittery animation?

A An animation playing on the opposite hand is influencing the PropWeapon joint.

B The position error on the compression settings is too low.

C The weapon attachment joint is **NOT** parented to the hand directly.

D The Animator’s root motion is influencing the position of the PropWeapon joint.

Question 3

A Game Design Document (GDD) for a survival game features a human player character being hunted by security robots. The robots have rigid exoskeletons and feature visible hydraulic pistons at their major joints. The Animation Director has requested that the robots have a stiffer, more mechanical motion. The human and robot characters have Humanoid Rig Animation Types to reuse locomotion animations.

How should the Technical Artist set up the Avatars for the characters to highlight the difference in motion style?

A Use the Per-Muscle Settings to restrict the robots' range of motion, and leave the human's settings at the default.

B Set the robots' Avatar in an A-pose, and set the human as a T-pose.

C Increase the robots' upper arm and upper leg bone lengths, and leave the human's upper arm and upper leg bone lengths as they were authored.

D Use Optional Bones on the robots to show increased range of motion, but do NOT enable Optional Bones on the human..

Question 4

A Technical Artist is working on a system for humanoid characters to plant their feet accurately on the environment during movement. The solution is applied to characters of various sizes. The solution involves having two collision meshes: one for the humanoid character's capsule to move over and another more accurate collision mesh for the feet to be planted on.

Part of the system requires projecting the position of the feet onto the ground and is shown below:

```
  Vector3 ProjectPositionOnGround(Vector3 position)
   {
       Vector3 ret = position;

       RaycastHit hitInfo = new RaycastHit();
       if (Physics.Raycast(position + new Vector3(0, 0.5f, 0),
new Vector3(0, -1, 0), out hitInfo, 1.0f, m_LayerMask))
       {
           ret = hitInfo.point;
       }

       return ret;
   }
```

The Technical Artist notices that on some characters the IK planting fails to work as expected. How should the code be modified to fix this problem?

A Dynamically set Layers to ensure that all characters are raycasting against the appropriate collision mesh.

B Use an offset based on character size to vary the Raycast origin.

C Scale the Raycast direction vector to ensure it always reaches the Collider.

D Use an offset based on character size for Raycast origin and maxDistance value.

Question 5

A character with a high, stiff collar on a cloak has a rig set up in the Digital Content Creation (DCC) package that prevents it from penetrating with the head and jaw during animation. To achieve the desired articulation, the collar setup uses 12 additional bones NOT driven by physics. All characters in the game are imported as Humanoid Rigs and reuse the same animation set.

The animation data takes up too much space for the target build platform.

What is the most efficient way to optimize the animation while matching the existing animation behavior?

A Edit the DCC Rig and reduce the joint count on the cloak collar.

B Edit the DCC Rig and replace the joint setup with BlendShapes on the cloak collar.

C Mask out the 12 cloak collar bones when importing animation, and recreate the collar collision behavior using Unity Physics components.

D Mask out the 12 cloak collar bones when importing the animation, and create a script that implements a behavior that matches the DCC Rig setup.

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