**1. Design the 3D Avatar**

* **Modeling**: Use 3D modeling software like Blender, Maya, or 3ds Max to create the avatar. Ensure the model has a rigged skeleton for animation.
* **Texturing**: Apply textures to the model to make it look realistic or stylized according to your requirements.
* **Facial Rigging**: Rig the face with bones or blend shapes to enable facial expressions.

**2. Animate the Avatar**

* **Facial Animation**: Create key facial expressions (smile, frown, etc.) and blend shapes.
* **Body Animation**: Create idle animations and other necessary gestures.

**3. Integrate Speech and Facial Recognition**

* **Speech Synthesis**: Use a Text-to-Speech (TTS) engine like Google Text-to-Speech, Amazon Polly, or Microsoft Azure TTS to give the avatar the ability to speak.
* **Speech Recognition**: Use a Speech-to-Text (STT) engine like Google Speech-to-Text or IBM Watson to understand customer queries.
* **Emotion Detection**: Integrate emotion detection to change facial expressions based on the context of the conversation.

**4. Real-time Interaction**

* **Game Engine**: Use a game engine like Unity or Unreal Engine to integrate the avatar model and animations.
* **AI Integration**: Use a conversational AI platform like Dialogflow, Microsoft Bot Framework, or IBM Watson to handle the dialogue management.
* **Facial Animation SDK**: Integrate an SDK like Live2D, Faceware, or Nvidia's Audio2Face to handle real-time facial animation based on speech and emotion.

**5. Deployment**

* **Platform**: Choose a platform for deployment, such as a website, mobile app, or kiosk.
* **Backend**: Set up a backend to handle the communication between the avatar, speech recognition, and conversational AI.
* **Testing**: Test the avatar in various scenarios to ensure it responds accurately and appropriately.

**Example Workflow**

1. **Design & Model**:
   * Create a 3D model of the avatar.
   * Texture and rig the model.
   * Animate key facial expressions and body gestures.
2. **Integrate Speech & AI**:
   * Use TTS and STT APIs for speech capabilities.
   * Connect to a conversational AI for understanding and generating responses.
   * Use emotion detection to adjust facial expressions.
3. **Develop the Application**:
   * Import the avatar model into Unity or Unreal Engine.
   * Create scripts to handle the interaction logic.
   * Integrate facial animation SDKs for real-time expression changes.
4. **Test & Deploy**:
   * Test the avatar’s interaction capabilities.
   * Deploy on your chosen platform.
   * Monitor and refine based on user feedback.

**Tools and Technologies**

* **3D Modeling**: Blender, Maya, 3ds Max
* **Game Engine**: Unity, Unreal Engine
* **TTS/STT**: Google Cloud Text-to-Speech, Amazon Polly, Microsoft Azure, Google Speech-to-Text
* **Conversational AI**: Dialogflow, Microsoft Bot Framework, IBM Watson
* **Facial Animation**: Live2D, Faceware, Nvidia Audio2Face

**Process brief**

[NVIDIA Omniverse Avatar Cloud Engine (ACE)](https://developer.nvidia.com/nvidia-omniverse-platform/ace), a collection of cloud-native AI microservices for faster, easier deployment of interactive avatars

[NVIDIA Tokkio](https://developer.nvidia.com/nvidia-omniverse-platform/ace/tokkio-showcase), a domain-specific AI reference application that leverages Omniverse ACE for creating fully autonomous interactive customer service avatars.

Omniverse ACE is a collection of cloud-native AI models and microservices for building, customizing, and deploying intelligent and engaging avatars easily. These AI microservices power the backend of interactive avatars, making it possible for these virtual robots to see, perceive, intelligently converse, and provide recommendations to users.

When an avatar is created, it must also be integrated into an application and deployed. This requires powerful GPUs to drive both the rendering of sophisticated 3D characters and the AI intelligence that brings them to life. Monolithic solutions are optimized for specific endpoints, while cloud-native solutions are more scalable across all endpoints, including mobile, web, and limited compute devices such as augmented reality headsets.

Omniverse ACE uses [Universal Scene Description (USD)](https://www.nvidia.com/en-us/omniverse/usd/) and the [NVIDIA Unified Compute Framework](https://developer.nvidia.com/ucf) (UCF), a fully accelerated framework that enables you to combine optimized and accelerated microservices into real-time AI applications.

Every microservice has a bounded domain context (animation AI, conversational AI, vision AI, data analytics, or graphics rendering) and can be independently managed and deployed from UCF Studio.

