The Goal

The goal of the workshop ist to build your own individual Virtual Idol.

For that we could follow several pipelines, different in complexity and usability. Considering that most of us are not too familiar with tools and technology in this field, the choice for us is web-basedpipeline on – more or less – semiprofessional level.

The Pipeline

Building such a functional, web-based avatar pipeline requires a deliberate selection of tools that offer both low entry barriers and sufficient technical capabilities. The here chosen „web-based semi-pro pipeline“ is therefore based on a combination of established, open, and well-integrated technologies that together enable a continuous workflow - from avatar design to interactive rendering in the browser. At the core of this pipeline are four tools that fulfill different but complementary roles:

- VRoid Studio
- VRM
- Three.js
- @pixiv/three-vrm.

VRoid Studio serves as the entry point to the pipeline and is used to create humanoid 3D avatars. Its focus lies on a parameterized, visually guided modeling approach that allows users to generate consistent and ready-to-use characters without requiring deep expertise in 3D modeling. The resulting avatars then are exported in a file format called **VRM**. This format acts as a standardized representation of humanoid figures. VRM is based on glTF 2.0 and extends it with avatar-specific features such as humanoid rigging, facial blendshapes, and metadata related to usage and identity. This enables a high level of interoperability across different systems and platforms.

For the web-based visualization in the browser, **Three.js** takes on the role of the central rendering engine. As a widely used **JavaScript** library, it provides access to WebGL and allows efficient rendering of 3D content directly in the browser.

The actual bridge between the **VRM** format and the rendering environment is provided by **@pixiv/three-vrm**. This library extends Three.js with dedicated functionality for loading, interpreting, and controlling VRM avatars, particularly in terms of humanoid animation, gaze behavior, and facial expressions.

Together, these four components form a coherent and modular pipeline that covers the entire process - from avatar creation and standardized representation to web-based visualization. This approach represents – hopefully - a practical balance between standardization and interoperability on the one hand, and feasibility and development effort on the other. It is especially well suited for prototyping, educational contexts, and first steps toward more complex virtual idol or metaverse systems.

The Workflow

1. VRoid Studio — avatar creation
2. *.vrm — Export file
3. Three.js — Basic 3D-Scene
4. three-vrm — loading the avatar laden

Some Additional Information on Web Programming

Implementing a web-based avatar pipeline requires a runtime environment that is both stable and flexible -capable of rendering 3D content while also handling interaction and media streams. In the architecture presented here, this role is fulfilled by core web technologies:

- **HTML**
- CSS
- **JavaScript**

These techs together form the technological foundation on which all other components—especially rendering, animation, and interaction—are built.

HTML, the HyperText Markup Language, provides the structural backbone of the application. It defines the semantic organization of the user interface, including elements such as the canvas used for 3D rendering, input fields for user interaction, and control elements like buttons. In this way, HTML establishes the connection between the visual representation of the avatar and the ways users can interact with it.

CSS, or Cascading Style Sheets, complements this structure by adding visual design and layout. Beyond traditional styling tasks, CSS also plays a role in integrating the 3D viewport into the overall application - for example through responsive behavior, layering, and the design of interface elements surrounding the avatar. This ensures that the avatar is not presented in isolation, but embedded within a coherent and well-designed application context.

The central control layer is **JavaScript**, which acts as the execution logic of the system. It is responsible for initializing and controlling the 3D scene - typically via libraries such as **Three.js** - loading and manipulating the VRM avatar, processing user interactions, and integrating external services such as text-to-speech or AI-based dialogue systems. In addition, JavaScript enables the handling of real-time data, for instance to synchronize audio with lip movements or to dynamically adapt animations and behavioral states.

Together, **HTML**, CSS, and **JavaScript** form a standardized, platform-independent, and widely supported runtime environment that allows complex avatar-based applications to run directly in the browser. This technological foundation is essential for the goals of the semi-professional pipeline, as it combines a low barrier to entry with a high degree of flexibility, while also supporting the integration of modern web APIs and external services. As such, these web technologies not only provide the technical basis, but also enable further expansion toward interactive, speaking, and ultimately autonomous avatar systems.

Some helpful links

VRoid Studio

Official source:	https://vroid.com/en/studio
Introduction:	https://vroid.pixiv.help/hc/en-us
Tutorials:	https://vroid.pixiv.help/hc/en-us/articles/360000957141
	https://www.youtube.com/results?search_query=vroid+studio+tutorial
	https://vroid.pixiv.help/hc/en-us/articles/15760756822297

VRM

Official:	https://vrm.dev/en/
	https://vrm.dev/en/vrm/
Spezification:	https://github.com/vrm-c/vrm-specification
Documentation:	https://vrm.dev/en/univrm/

Three.js

Official:	https://threejs.org/
Documentation:	https://threejs.org/docs/
Samples:	https://threejs.org/examples/
Introduction	https://threejs.org/manual/
	https://discoverthreejs.com/

@pixiv/three-vrm

GitHub:	https://github.com/pixiv/three-vrm
Documentation:	https://pixiv.github.io/three-vrm/
Samples:	https://pixiv.github.io/three-vrm/examples/
NPM:	https://www.npmjs.com/package/@pixiv/three-vrm

HTML

Official reference: https://developer.mozilla.org/en-US/docs/Web/HTML

Introduction: https://developer.mozilla.org/en-US/docs/Learn/HTML

https://www.w3schools.com/html/

Interactive learning environments;

https://codepen.io/

https://jsfiddle.net/

CSS

Official reference: https://developer.mozilla.org/en-US/docs/Web/CSS

https://www.w3.org/Style/CSS/

JavaScript

Official reference: https://developer.mozilla.org/en-US/docs/Web/JavaScript

Introduction: https://developer.mozilla.org/en-US/docs/Learn/JavaScript

https://javascript.info/

Special Web-APIs:

- Speech: https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API
- Audio: https://developer.mozilla.org/en-US/docs/Web/API/AudioContext
- Media devices: https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices
- Canvas/ WebGL: https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API
- Animation: https://developer.mozilla.org/en-US/docs/Web/API/Window/requestAnimationFrame