Continue

Setting Up DirectX 11 for 3D Rendering with Braynzar Soft Tutorials #### Collection: 1519 Downloads This tutorial series covers the basics of setting up and utilizing DirectX 11 for 3D rendering. The following lessons are a part of this collection: * Setting Up in VS 2010 (Lesson 1) * Initializing Direct3D 11 (Lesson 2) * Begin Drawing! (Lesson 3) * Color (Lesson 4) * Indices (Lesson 5) * Depth (Lesson 6) * World View and Local Spaces (static Camera) (Lesson 7) * Transformations (Lesson 8) * Render States (Lesson 9) * Textures (Lesson 10) * Blending (Lesson 11) The following lessons will cover additional aspects of DirectX 11 for 3D rendering: * Note: The provided text has been paraphrased to improve readability and understanding. This tutorial aims to cover various advanced topics in graphics rendering using Direct3D 11. It starts with adding blending effects to primitives, allowing for stained glass-like appearances. The lesson then delves into the issue of transparent objects not always being fully transparent when rendered together. Next, it briefly covers pixel clipping and its importance in efficiently drawing pixels on the screen. Moving on, it tackles a more complex topic: rendering simple fonts within Direct3D 11. Due to Microsoft's change in API structure, using fonts directly with D3D 11 is challenging. The tutorial explores how to use Direct2D and DirectWrite for font implementation alongside D3D 10.1. Subsequent lessons include creating a high-resolution timer for precise timing of game elements' updates, regardless of the frame rate, and a lesson on simple lighting using directional light sources. It then proceeds to cover point lights, demonstrating how they can be used in conjunction with other graphics elements, followed by an explanation of Direct Input methods from users through keyboards, mice, or joysticks. Lastly, it discusses implementing a first-person camera view that simulates the player's perspective while moving around the scene, as well as transitioning to fullscreen mode and exiting the application smoothly. Getting errors... Let's start with 3D textures! In this series, we'll learn about Cube Mapping (Skybox), which involves using a 3D texture to texture a sphere. This technique is perfect for creating a skybox. Then, we'll move on to Spotlights, where we'll learn how to implement a spotlight as a flashlight. We'll build upon the previous lesson, First-Person Camera. Next up is Loading Static 3D Models (.obj Format), which will teach us how to load a static 3D model from an .obj file. This lesson will lay the groundwork for upcoming lessons on Specular Lighting and Normal Mapping. We'll also explore Picking, where we'll learn how to turn a 2D screen position into a 3D ray in world space. This allows us to check if that ray intersects with any objects on the screen. Finally, we'll cover Bounding Volumes, which are essential for optimizing performance when dealing with complex scenes and many models. The upcoming lessons will cover collision detection and model loading techniques. Lesson 26 focuses on bounding volume collision detection, where objects are simplified into bounding volumes for faster computation. Students will learn to build a pyramid of bottles and throw one when clicking the mouse button, with collision resulting in score increase. Lesson 27 introduces MD5 models, which come in two files: "md5mesh" and "md5anim". The first lesson covers loading the MD5 model from the "md5mesh" file and setting up vertex positions based on joint layout. Students will also learn about quaternions and how bones work in skinned models. The second part of Lesson 27, Skeletal Animation, teaches students to animate the MD5 model using a skeletal system. This system is memory-efficient compared to keyframe animations and allows for more complex movements like rag-doll physics. In the subsequent lessons, students will learn about a free-look camera, heightmap (terrain) loading from grayscale BMP images, and sliding camera collision detection using an ellipsoid technique. We'll dive into "Detection and Response" by Kasper Fauerby, exploring collision detection between a swept sphere and triangle. This results in a smooth "sliding" camera that navigates terrain, stairs, and small objects without getting stuck on edges. We'll also implement gravity to prevent floating away. This technique is versatile and can be applied to any object or even modified to create bouncing effects instead of sliding. Next, we'll tackle a simple 3rd person camera lesson, teaching you how to create a vector camera, smoothly rotate your character towards their destination, and rotate the camera around them. In another lesson, we'll delve into instancing with indexed primitives, rendering a forest using this efficient technique. We'll draw 400 trees with 1000 leaves each, resulting in over 400,000 leaves! This method can significantly boost performance by reducing the number of objects being rendered. The next lesson focuses on CPU-side frustum culling, which allows us to process 4000 trees in our scene efficiently. We'll learn how to check if an object's AABB is within the camera's view and only send relevant data to the GPU for more accurate rendering. Finally, we'll create a map by rendering terrain onto a texture and drawing it in the backbuffer's bottom right corner. This lesson provides a straightforward introduction to this technique. We will be building upon this concept in the next two lessons. A method called billboarding is used to draw distant objects by rendering a single quad per object rather than drawing the entire geometry. This technique significantly reduces the complexity of rendering far-off trees, which can contain thousands of faces. In our lesson today, we'll utilize the Geometry shader to create perspective-facing billboarders. The process involves sending a single point to the shaders and expanding it into a quad using the geometry shader. LearnD3D11 is an educational resource for anyone looking to learn Direct3D11, commonly referred to as DirectX 11. This guide provides a comprehensive overview of basic Direct3D11 usage and general graphics programming topics without requiring prior experience in the field. The codebase is written in C++, but its concepts can be easily adapted to other languages with equivalent skills. Please note that this project is still in progress, and some content may be missing or contain errors. If you notice any typos, bugs, or have questions, feel free to open an issue or send a pull request on our platform. You're also welcome to join our Discord server for further collaboration. To get started with LearnD3D11, simply dive into the first chapter and begin your learning journey.

Direct3d 11 install. What is direct3d 11. Direct3d tutorial. Direct3d 9 vs 11. Direct3d 11 tutorial c++.