**🎮 Presentation Title:**

**"Godot Engine vs Custom ECS Game Engine: A Structural Perspective"**

**🗂️ Slide-by-Slide Outline (20 Minutes)**

**🔹 Slide 1: Title Slide (0:30 min)**

* Presentation title
* Your name and course info

**🔹 Slide 2: Agenda (0:30 min)**

* What is Godot?
* Godot’s key features
* Godot’s game architecture
* My ECS vs Godot comparison
* Pros and cons of Godot

**🟢 Slide 3: What is Godot? (1 min)**

* Free, open-source game engine
* Developed by the community, maintained by Godot Foundation
* Used for 2D and 3D game development
* Gained traction since ~2018, especially in indie community

🧠 **Say**: “It’s especially loved for its small footprint, built-in editor, and ease of scripting.”

**🟢 Slide 4: Is Godot Popular? (1 min)**

* Stars on GitHub (50k+)
* Used in indie games like *Cassette Beasts*, *The Garden Path*
* Compared to Unity, Unreal — smaller market, but growing fast

📊 Show a chart or GitHub growth snapshot

**🟦 Slide 5: Godot Key Features (2 min)**

* Visual scene editor
* Full 2D and 3D support
* Built-in physics and animation system
* Powerful debugger
* Multi-language scripting (GDScript, C#, C++)

🎯 Demo screenshot of the Godot editor

**🟦 Slide 6: GDScript (1 min)**

* Python-like language designed for games
* Tight integration with Godot scene system
* Supports signals and coroutines
* Fast iteration and prototyping

🧠 **Say**: “Compared to C++ in our ECS engine, GDScript trades raw performance for developer productivity.”

**🟨 Slide 7: Godot’s Architecture Overview (3 min)**

* **Scene**: reusable game object with logic
* **Node**: building block (Sprite, Audio, Camera, etc.)
* **Instance**: a copy of a scene used in another scene
* **Signal**: observer/event system for decoupled communication

🖼️ Diagram: scene tree → Player (with Sprite, CollisionShape2D, etc.)

Scene tree

🧠 **Say**: “This is Object-Oriented. Logic and data are grouped inside nodes. It’s different from our ECS approach.”

**🔴 Slide 8: Our ECS Engine Overview (1.5 min)**

* Entity = ID
* Component = data (position, velocity, sprite)
* System = logic
* Game loop: input → update → render
* Manual resource and event management

🧩 Diagram: ECS triangle or system flow

**🔴 Slide 9: Comparison: Same Game in Both Engines (3 min)**

**Example Game: [your chosen game, e.g., 2D shooter]**

| **Feature** | **Godot** | **ECS (C++)** |
| --- | --- | --- |
| Architecture | Scene Tree (OOP) | ECS (Data-oriented) |
| Player Design | Sprite + script node | Entity with Sprite/MovementCmp |
| Events | Signals | Custom Event System |
| Resource Mgmt | Automatic (Resource files) | Manual (ResourceManager.cpp) |

🎥 Optionally show a **side-by-side demo or screenshots**

🧠 **Say**: “The gameplay is the same, but the structure and developer experience differ a lot.”

**🟣 Slide 10: Pros & Cons of Godot (2 min)**

**✅ Pros:**

* Fast prototyping with built-in editor
* Great for 2D and small 3D games
* Signals and node system reduce boilerplate
* Active open-source community

**❌ Cons:**

* Performance not on par with custom C++ ECS
* Less control over low-level systems
* Scene tree abstraction may hide what's really happening

🧠 **Say**: “It’s ideal for rapid dev, but if you want full control or learn internals, custom ECS is better.”

**🟢 Slide 11: Takeaways (1 min)**

* Godot offers a clean high-level workflow
* Our ECS gives fine-grained control and deep learning
* Choosing a game engine depends on **your project’s needs and goals**

**🟡 Slide 12: Q&A or Discussion Prompt (2–3 min)**

* “What engine would you use for your next game?”
* “How would you extend the ECS engine to include a scene concept like Godot?”