Prototype Design Pattern - A Way to Clone an Object

Khang P. Nguyen [1], Nghia T. Hoang [1], Cao C. Phan [1] and Hoang N. Nguyen [1]

[1]University of Science - Vietnam National University

December 1, 2025

Outline

Outline

What is Prototype Pattern

Definition

Prototype Pattern is a **creational** design pattern that enables object duplication through **cloning** rather than **instantiation** (Chan, 2025)

What is Prototype Pattern

Definition

Prototype Pattern is a **creational** design pattern that enables object duplication through **cloning** rather than **instantiation**

Why should we use it?

This approach is particularly **useful** when object creation is **costly**, objects have **numerous** configurations, or you want to **decouple** object creation from its representation.

Problem

Description

You instantly need to create **1.000** objects `Solid` that has complicated *attributes, classes, and methods* such as **(Texture, 3D Model, Audio, Database, .etc)**

Problem

Description

You instantly need to create **1.000** objects `Solid` that has complicated *attributes, classes, and methods* such as **(Texture, 3D Model, Audio, Database, .etc)**

Naive Solution

Use a `for` loop `1000 times` to execute the command `new Soldier()`.

Problem

Description

You instantly need to create **1.000** objects `Solid` that has complicated *attributes, classes, and methods* such as **(Texture, 3D Model, Audio, Database, .etc)**

Naive Solution

Use a `for` loop `1000 times` to execute the command `new Soldier()`.

Problem

But for each time you initialize an object, which **MUST** load all of the data from disk (I/O), analyze configurations, and connect to the Database to get some attributes.

Problem

Description

You instantly need to create **1.000** objects `Solid` that has complicated *attributes, classes, and methods* such as **(Texture, 3D Model, Audio, Database, .etc)**

Naive Solution

Use a `for` loop `1000 times` to execute the command `new Soldier()`.

The Consequence

- Spend a lot of `CPU/RAM` resources, **lag**, or **”not responding”** error.

Optimized Approach

Prototype

Create a single **prototype** object with all heavy assets **already loaded**. Then, simply `clone` it when needed. (GeeksforGeeks,)
This approach saves costly resources and time, especially when object creation is a **heavy** process.

Optimized Approach

Prototype

Create a single **prototype** object with all heavy assets **already loaded**. Then, simply `clone` it when needed. (GeeksforGeeks,)
This approach saves costly resources and time, especially when object creation is a **heavy** process.

Suppose a user creates a document with a specific layout, fonts, and styling, and wishes to create similar documents with slight modifications.

Optimized Approach

Document and Content Management Systems *can use the prototype pattern to manage* `document templates`*. Users can* `clone` *an existing template and then make specific modifications.*

Optimized Approach

Document and Content Management Systems *can use the prototype pattern to manage* `document templates`*. Users can* `clone` *an existing template and then make specific modifications.*

Game engines *can use them to frequently* `clone` *complex characters or terrain objects. The Prototype approach allows efficient duplication without repeating costly initialization.*

Analogy Example

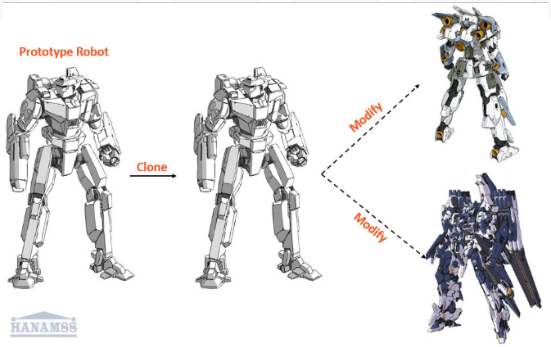

Figure 1 – Analogy Example for Prototype Pattern.

Outline

Conclusion

Achievements

- We proposed **LTDAD-Talker**, a **landmark-guided** and **2-stages** talking face framework that ensures accurate lip-sync, temporal consistency, and high visual quality.
- By combining **Attention Mechanism** and a **Detail-Aware Discriminator**, our model generates realistic and smooth videos while preserving speaker identity.

Future Directions

- Enhance the Audio2Lmk module to produce landmarks with higher audio–lip synchronization accuracy.
- Explore Diffusion-based approaches for video synthesis.
- Investigate advanced enhancement techniques to further improve output video quality in terms of sharpness, realism, and temporal smoothness.

Thank You for Your Attention!

If you have any *questions*, please keep them in your *mind*.

Reference

CHAN, M. M. **Understanding the Prototype Design Pattern in C#**. 2025. ⟨https://chanmingman.wordpress.com/2025/11/30/understanding-the-prototype-design-pattern-in-c/⟩. Accessed: Nov. 30, 2025.

GeeksforGeeks. **Prototype Design Pattern**. ⟨https://www.geeksforgeeks.org/system-design/prototype-design-pattern/⟩. Accessed: Nov. 30, 2025.