Introduction

Animations are present everywhere, from games to the electronic billboards we encounter in everyday life. In recent years, there has been an attempt to reduce the complexity of writing code while preserving human creativity and intent. This motivation extends to CSS animations, which form an important aspect of modern web development. Although AI-based tools claim to automate this process, they often fail due to an inability to capture the precise animation behavior expected by non-technical users.

To address this problem, we propose a structured approach with the aim of simplifying the creation of CSS animations. This work introduces GameScript, a domain-specific language (DSL), as an abstraction layer for animation authoring. Rather than relying solely on direct CSS manipulation, animation behavior is described at a higher level and later translated into executable CSS constructs.

GameScript was originally designed for developing games such as Asteroid Destroyer and Tetris, where system behavior relies on repeatability, explicit rules, and an underlying ideology of time-driven execution. Owing to these properties, it is interpreted in this work within the web domain, rather than being restricted to game development alone.

That said, the proposed system does not seek to replace manual CSS animation coding. Instead, it functions as a generator that produces CSS code with ease, focusing on smooth animation curves, interaction states such as hover, focus, and active, and other inherent behavioral relationships. This includes group and peer-based logic, as well as property transformations that evolve over time.

Despite its strengths, the framework exhibits certain limitations. Some CSS features, particularly grid-based layouts, remain unreachable within the current architecture due to the behavioral nature of GameScript. These constraints arise from the mismatch between spatial layout representation and time-based rule evaluation.

Finally, the system architecture is composed of a parser, a sampling engine, a semantic interpreter, and a dedicated time layer. This underlying time layer forms a critical aspect of the system, enabling controlled temporal evolution of animations. Through this pipeline, high-level rules are interpreted and transformed into CSS animations in a consistent and predictable manner.