Abstract

The domain-specific language (DSL) of Minecraft, known as *mcfunction* enables the creation of modular content bundles, or "packs", capable of altering gameplay. Despite its widespread use, this language presents significant structural and syntactical limitations: each function must be defined within a separate file, and lacks programming constructs such as variables, conditional statements, and iteration mechanisms. These constraints result in verbose and repetitive code, which hinders scalability and maintainability in complex projects.

To overcome these issues, this thesis introduces a Java library developed during an academic internship, starting from an in-depth analysis of *mcfunction*'s design flaws and leading to the formulation of an abstraction that represents the pack structure as a tree of typed objects. By leveraging standard Java syntax and factory methods, the library enables the programmatic generation of packs, offering syntactic sugar and utilities that simplify access to core resource files. The proposed approach provides compile time validation, supports the definition of multiple resources within a single source file, and automates the generation of boilerplate structures, thus eliminating the need for external preprocessors or hybrid syntaxes adopted by alternative solutions.

A case study validates the approach: the example pack required 40% less code while consolidating 31 files into 3 source files, demonstrating substantial improvements in both code density and project maintainability.