# Coding style and practices guide

## Accepted practiced

### Software

For consistency and stability, the following software versions will be used…

1. Godot 4.4.1
   1. Gdunit4 – for unit testing
2. Blender 4.2.1 LTS
3. Audacity 3.6.1

### Accepted file types

* **Combined objects**: .blend
* **3D models**: FBX, gltf 2.0 (glb/gltf)
* **Materials**: png, webp, jpg (for fully filled only)
* **Audio**: mp4, wav, ogg

## Coding style guide

### Naming Conventions

* **Variables**: Use snake\_case for variables. Start with a lowercase letter, e.g., player\_health.
* **Functions**: Also use snake\_case. Names should be verbs if they perform actions, e.g., update\_score().
* **Constants**: Use all uppercase with underscores, e.g., MAX\_SPEED.
* **Classes**: Use PascalCase, starting each word with a capital letter, e.g., EnemyCharacter.
* **Signals**: Start with a verb, followed by the condition, using snake\_case, e.g., hit\_by\_enemy.

### Indentation and Whitespace

* Use tabs for indentation.
* Add blank lines to separate logical blocks of code.
* Use a single space around operators (=, +, -, <, >, etc.), except when passing keyword arguments in function calls.
* Use words for logical operators where applicable, e.g., or instead of ||, and instead of &&, and not instead of !.

### Commenting and Documentation

* Use comments to explain "why" something is done, not "what" is done—unless it’s particularly complex.
* Comments should be full sentences, starting with a capital letter and ending with a period.
* Use ## for comments and function documentation. This makes documentation searchable

### Function Definitions

* Keep functions short and focused. Each function should do one thing well.
* Parameters should have descriptive names. Use default parameter values where practical.
* Document every function with comments above the function definition, explaining purpose, parameters, and return values.

### Error Checking

* Check for errors or invalid conditions early in functions, and handle them gracefully.
* Use assertions sparingly to check for conditions that should never occur.

### File Structure

* Place requires and imports at the top of the file.
* Organize global variables as follows, with comments defining each group:
  + **Constants**: Constants used throughout the script.
  + **Export Variables**: Variables exported to the inspector for easy adjustment.
  + **Object References**: References to other nodes or objects.
  + **Runtime Variables**: Variables that change during runtime.
* Define classes next. If a file contains multiple classes, consider splitting them into separate files.
* Main code (if any) should go at the bottom of the file.

### Use of Godot Features

* Prefer using built-in types like Vector2, Rect2, etc., for math operations.
* Use preload for loading resources like scenes and textures, which are known before runtime and won't change.
* Utilize signals for communication between nodes and scripts effectively.

### Version Control Integration

* Include a .gitignore file in your project root to avoid committing unnecessary files like \*.import or the user:// directory.
* Commit changes frequently with clear, concise commit messages.