Code Conventions and Standards

1. Naming Conventions

1. Use camel case for variables.
2. Use Pascal case for method names.
3. Use Hungarian notation for variables.
4. Always specify access level modifiers consistently.
5. Prefix private variables and methods with an underscore.
6. Suffix variables and methods with system-unique declared names, followed by an underscore.
7. Enums should use Pascal case, and if possible, have a singular noun name.
8. Every class should use setters and getters.
9. Namespaces should use Pascal case, and sub-namespaces should be separated with a middle dot.

2. Code Readability

1. Strive for brevity.
2. Utilize appropriate naming conventions.
3. Segment blocks of code within the same section into paragraphs.
4. Avoid lengthy functions; ideally, a single function should perform a single task.
5. Follow the DRY (Don't Repeat Yourself) principle and automate repetitive tasks when necessary. Avoid duplicating code.
6. Avoid excessive nesting, as too many levels can make code harder to read. It is mandatory to maintain a complexity level below 120 percent using the Cognitive Complexity plugin. link.
7. Avoid long lines to enhance human readability; shorter horizontal lines are preferable.
8. Use meaningful variable names that clearly describe their purpose, and do not reuse variables.
9. Always include spaces after commas, pointers, and both before and after mathematical operators.
10. One-line or two-line if-else statements or for loops (without braces) and methods (with braces) are permissible if clarity is maintained.
11. Place event functions in execution order according to the Unity User Manual, always after method declarations, which should follow variable declarations. The order is: Awake, OnEnable, Start, FixedUpdate, OnTrigger, OnCollision, OnMouse, Update, StartCoroutine, LateUpdate, OnDrawGizmo, OnApplicationQuit, OnDisable, OnDestroy. link.

3. Headers

1. Include the module name.
2. Provide a summary of the module's functionality.
3. List the functions within the module.
4. Document variables accessed by the module.

4. Good Practices

1. Maintain a daily backup of your work, including versions.
2. Place comments on separate lines, and if necessary, use regions for organizing code. Prioritize documentation.

5. Exception Handling

1. Wrap the code in a try-catch block, with the catch block left empty.
2. If an issue is not easily solvable, leave it and provide comments for other team members to address promptly.

6. Indentation Style

1. Use the Allman brace placement style, and place inline comments after the closing brace in large statements.
2. Set the tab size to equal 4 spaces.
3. Follow the BSD style in general.

7. Project Structure

1. Always use Pascal case for naming, and avoid using spaces.
2. Avoid using Unicode characters or other symbols in file and folder names.
3. Do not leave empty folders; use ".keeper" files inside instead.
4. Limit folder nesting to a maximum of three levels.

8. Scene

1. Always use Pascal case for naming scenes and avoid using spaces, except for naming clones or instances.
2. When naming assets and other elements, use underscores to separate implicit definitions.