Example Documentation: Style Guide for Unreal Engine 4
Extra Example Documentation: Tips for Cleaning Blueprints
Documentation for Steering Wheel: LogitechGamingSteeringWheelSDK.pdf
Primary Contact: dhogue@umich.edu
Previous Year's Documentation: Data Dictionary
Screenshots of Code: Screenshots of code

UNREAL ENGINE 4 Blueprint Documentation Guidelines

1. Naming Conventions

1.1 Variable Naming

1.1.1 Forbidden Characters

- White space of any kind
- Backward slashes \
- Symbols i.e. #!@$%
- Any Unicode character

1.1.2 Non-Boolean Variables

- All non-boolean variable names must be clear and descriptive nouns
- All non-boolean variables should be in the form of PascalCase
    - *PascalCase: The first letter of every word is capitalized*

1.1.3 Boolean Variables

- Booleans should be descriptive adjectives
- Booleans should follow PascalCase but have a lowercase letter "b" in front of it
- Boolean variables should not be phrased in the form of a question i.e. bIsDead, rather should be bDead
    - *"Is" is reserved for functions*

1.1.4 Other Variables

- No explicitly naming the type in the identifier
- Arrays should be plural nouns

1.1.5 Example Variable Naming

- Brake value
    - Variable type: Non-boolean
    - Variable naming: BrakeVal
- Whether brake sound is played or not
    - Variable type: Boolean
    - Variable naming: bBrakeSoundPlayed

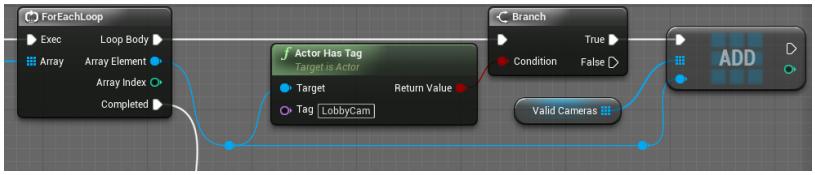
2. Blueprint flow

2.1 Graphs

2.1.1 Guidelines

- No Spaghetti!
    - Wires must have a clear start and end. Tangling of wires will lead to bad debugging and reduce the quality of work.
- Align wires, not nodes
    - When aligning nodes together, it is preferable to align the wires themselves rather than the nodes, which helps reduce the wire overlap.

Example Image 2.1.1a



- Tips + Shortcuts
    - You can straighten wires by using the Straighten Connections command with BP nodes selected. Hotkey: Q
    - White Exec lines (the ones at the top) take precedence over any other line when considering straightening lines.

2.2 Functions

2.2.1 Return Nodes

- All functions should have return nodes without any exception
- Example image

2.2.1 Total Nodes

- Node count should never be over 50.
- If it is over, consider splitting the function into two
- Following nodes do not count towards this “Node Count”
    - Comment
    - Route
    - Getters
    - Cast
    - Breaking a Struct
    - Function Entry
    - Self

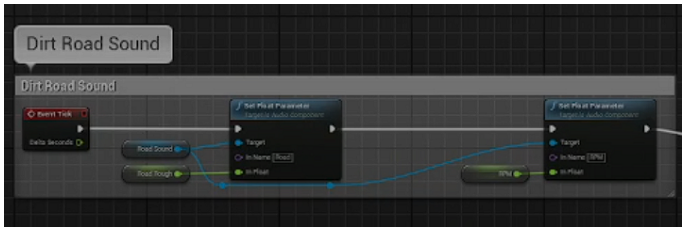
3. Comments

3.1 On-block Comments

3.1.1 Creating on-block comments (Equivalent to block comments)

- Usage: [Drag over code blocks then right-click]

Example Image 3.1.1a



3.1.2 On-block color organization

- Scale from traffic lights to determine the level of urgency to fix/to complete
    - USE ONLY a scale between red to green. Level of urgency can also be expressed in the prefix, this just helps with visuals.

- Determines if the code is temporary or permanent or needs to be fixed

3.1.3 On Block Comment Formatting

- Format of the block code should be as follows
    - [**PREFIX**] [description] [**TEAM**]

- Examples of each section are listed below:
    - FIX ME
    - TODO
    - %done
- [description] Examples
    - Features (anything that changes/moves something for an end goal go together)
- [TEAM] Examples
    - Platform: PLT
    - Occlusion: OCC
    - Data: DATA
    - Experiments: EXP
    - Maps: MAP
    - Roads: ROAD

3.2 Node Comments

3.2.1 Creating Node Comments

- To add a comment directly to a node in a Blueprint graph: Right-click on the node.
- Type a comment into the Node Comment text box in the menu that appears, then press Enter.

3.2.2 Node Comments Formatting

- Formatting is as follows:

  [Description] + [Identifier for documentation]
- Documenting Node Comments Method
    1. If the description on the node comment itself isn't enough for someone to understand on its own, label it with an identifier such as:

        - “Branch 1 + Documentation”
    2. Then take a screenshot and explain it on the respective document tab to provide further explanation.
    3. Use Occlusion/Platform’s documentation tab as a reference and document the feature shown in the screenshot.

4. Documenting in this document

Sections required for each screenshot part of blueprint:

1. *Main goal*
2. *Variables*
3. *Branches (this can be multiple parts)*
4. *Functions*
5. *Block Comments (if applicable)*
6. *Flows/Logic*

4. Other Organizational Methods (OPTIONAL)

4.1 Collapsible nodes

4.1.1 Creating Collapsible nodes

4.1.2 Collapsible nodes formatting

4.2 Bypasses

Bypasses

- Use Sequence nodes to create multiple branches of coding that can run independently.
    - (be careful bc too many then braces will causes delays in running/input)

Style:

- Can collapse nodes to make code easier to see.
    - But could be a problem as certain logic won't run.
    - Could also use functions, but that introduces the problem or benefit of local variables.
    -