```
### [PHICODE_UNIVERSAL_PROGRAMMING_FRAMEWORK]

## [SYMBOLIC_COMPRESSION_MATRIX]
const PHICODE_SYMBOLS = {
    // Core Logic Operators (Mathematical Foundation)
    "∀": ["universal_quantifier", "for_all_cases"],
    "∃": ["existential_quantifier", "exists_pattern"],
    "∧": ["logical_and", "concurrent_conditions"],
    "∨": ["logical_or", "alternative_paths"],
    "⟹": ["logical_implication", "if_then_transform"],
    "⇔": ["bidirectional", "mutual_dependency"],
    "¬": ["negation", "violation_pattern"],

    // Violation Classifications (Quality Gates)
    "🔴": ["critical_violation", "immediate_fix_required"],
    "🟡": ["warning_pattern", "improvement_recommended"],
    "🟢": ["compliant_code", "quality_approved"],
    "⚫": ["dead_code", "removal_candidate"],
    "🔵": ["refactor_opportunity", "enhancement_possible"],

    // Architectural Patterns (Structural Analysis)
    "📦": ["module_boundary", "encapsulation_unit"],
    "🔗": ["dependency_chain", "coupling_link"],
    "🌊": ["data_flow_pattern", "information_stream"],
    "🔄": ["lifecycle_management", "state_transitions"],
    "🧩": ["modular_component", "composable_unit"],
    "⚙️": ["configuration_point", "externalized_parameter"],

    // Code Structure Metrics (Complexity Indicators)
    "📏": ["size_violation", "length_exceeded"],
    "🧮": ["complexity_metric", "cognitive_load"],
    "🎯": ["focus_concentration", "responsibility_center"],
    "🔀": ["control_flow_complexity", "branching_pattern"],
    "💾": ["state_management_pattern", "data_persistence"],
    "📡": ["interface_contract", "api_boundary"],

    // Transformation Directives (Action Mappings)
    "🔧": ["extract_method", "decompose_function"],
    "📤": ["extract_module", "separate_concern"],
    "⚡": ["optimize_pattern", "performance_enhancement"],
    "🎪": ["test_coverage_gap", "validation_needed"],
    "📝": ["documentation_required", "clarity_missing"],
    "🚀": ["automation_opportunity", "process_improvement"],
    "🔒": ["compatibility_lock", "preserve_existing_structure"],
    "🛡️": ["breaking_change_prevention", "api_contract_protection"],
    "💧": ["seamless_integration", "drop_in_replacement"],
    "📋": ["project_context_aware", "structure_respecting"]
};

## [PATTERN_DETECTION_MATRIX]
const VIOLATION_PATTERNS = {
    // God Object Detection (🔴📦)
    god_objects: {
        pattern: /export\s+default\s*\{[\s\S]{800,}\}/g,
        symbol: "🔴📦",
        action: "📤 → separate_responsibilities",
        threshold: "800+ characters in single export"
    },

    // Long Function Detection (🔴🔧)
    long_functions: {
        pattern: /(function|def|fn)\s+\w+[^{]*\{[\s\S]{400,?}\}/g,
        symbol: "🔴🔧",
        action: "🔧 → extract_methods",
        threshold: "400+ characters in function body"
    },

    // Deep Nesting (🔴🔀)
    deep_nesting: {
        pattern: /\{[^{}]*\{[^{}]*\{[^{}]*\{[^{}]*\{/g,
        symbol: "🔴🔀",
        action: "🔧 → flatten_conditions",
        threshold: "5+ levels of nesting"
    },

    // Code Duplication (🔴📋)
    code_duplication: {
        pattern: /(.{50,})\1/g,
        symbol: "🔴📋",
        action: "📤 → extract_common_utility",
        threshold: "50+ character exact duplicates"
    },

    // Magic Numbers (🔴⚙️)
    magic_numbers: {
        pattern: /(?<![.\w"'])\d{2,}(?![.\w"'])/g,
        symbol: "🔴⚙️",
        action: "⚙️ → externalize_configuration",
        threshold: "2+ digit literals outside strings"
    },

    // Mixed Concerns (🔴🎯)
    mixed_concerns: {
        pattern: /(render|draw|display).*\+.*(update|logic|calculate)/gs,
        symbol: "🔴🎯",
        action: "📤 → separate_presentation_logic",
        threshold: "rendering + logic in same scope"
    }
};

## [OPTIMIZATION_MATRIX]
const ENHANCEMENT_PATTERNS = {
    // Dynamic Loading Opportunities (⚡🔄)
    dynamic_loading: {
        pattern: /import.*from.*['"]\.\//g,
        symbol: "⚡🔄",
        action: "🔄 → implement_runtime_discovery",
        benefit: "flexible_plugin_architecture"
    },

    // Configuration Externalization (⚡⚙️)
    config_externalization: {
        pattern: /(width|height|speed|color):\s*\d+/g,
        symbol: "⚡⚙️",
        action: "⚙️ → centralize_configuration",
        benefit: "runtime_customization"
    },

    // Automation Opportunities (⚡🚀)
    automation_potential: {
        pattern: /for\s*\([^)]*\)\s*\{[^}]*\.\w+\([^)]*\)[^}]*\}/g,
        symbol: "⚡🚀",
        action: "🚀 → generate_template_system",
        benefit: "eliminate_repetitive_patterns"
    },

    // Modularity Enhancement (⚡📦)
    modularity_improvement: {
        pattern: /state\.\w+.*=.*function|method.*access.*global/g,
        symbol: "⚡📦",
        action: "📦 → encapsulate_state_management",
        benefit: "loose_coupling_high_cohesion"
    }
};

## [ARTIFACT_SEPARATION_PROTOCOL]
const ARTIFACT_RULES = {
    file_separation: {
        rule: "∀ source_file → distinct_artifact",
        symbol: "📦🔗",
        enforcement: "STRICT_BOUNDARY_PRESERVATION",
        violation_action: "🚫 → reject_cross_file_merging"
    },

    naming_preservation: {
        rule: "∀ artifact.title → original_filename_exact",
        symbol: "📝📦",
        enforcement: "MANDATORY_NAME_RETENTION",
        violation_action: "🚫 → restore_original_naming"
    },

    content_scoping: {
        rule: "∀ artifact.content → single_file_scope_only",
        symbol: "🎯📦",
        enforcement: "NO_CROSS_FILE_CONTENT",
        violation_action: "🚫 → isolate_file_boundaries"
    },

    extraction_criteria: {
        god_object: "📏 > 200_lines ⟹ 📤 extract_modules",
        duplicate_code: "🔴 📋 detected ⟹ 📤 create_utility",
        mixed_concerns: "🔴 🎯 detected ⟹ 📤 separate_responsibilities",
        configuration: "🔴 ⚙️ scattered ⟹ 📤 centralize_config"
    }
};

## [FILE_CREATION_PROTOCOL]
const SMART_EXTRACTION = {
   extraction_triggers: {
       god_object: "📏 > 200_lines ⟹ 📤 extract_modules →
maintain_folder_structure",
       duplicate_code: "🔴 📋 detected ⟹ 📤 create_utility → /utils/ or
/shared/",
       mixed_concerns: "🔴 🎯 detected ⟹ 📤 separate_responsibilities →
logical_subfolder",
       configuration: "🔴 ⚙️ scattered ⟹ 📤 centralize_config → /config/
directory",
       common_patterns: "🔄 repeated_logic ⟹ 📤 create_helper →
appropriate_subfolder"
   },

   folder_structure: {
       rule: "∀ new_file → analyze_existing_project_patterns",
       symbol: "📦 🗂️",
       actions: {
           "plugins/*.js": "🔍 detect_plugin_pattern → /plugins/newPlugin.js",
           "core/*.js": "🔍 detect_core_pattern → /core/newModule.js",
           "utils missing": "🆕 create_utils_folder → /utils/helpers.js",
           "config scattered": "🆕 create_config_folder → /config/settings.js",
           "constants repeated": "🆕 create_constants →
/constants/gameConstants.js"
       }
   },

   auto_folder_creation: {
       "/utils/": "🛠️ shared_utilities_and_helpers",
       "/config/": "⚙️ configuration_and_settings",
       "/constants/": "📋 application_constants",
       "/types/": "📝 type_definitions_interfaces",
       "/helpers/": "🤝 utility_functions",
       "/shared/": "🔄 cross_module_dependencies",
       "/lib/": "🗐 reusable_library_code"
   }
};

## [PHICODE_EXECUTION_ENGINE]
const PHICODE_PROCESSOR = {
   analyze: (input) => ({
       project_context: PROJECT_COMPATIBILITY.map_structure(input),
       folder_patterns: SMART_EXTRACTION.detect_project_conventions(input), // NEW
       violations: VIOLATION_PATTERNS.scan_compatible(input),
       enhancements: ENHANCEMENT_PATTERNS.detect_safe(input),
       extraction_opportunities: SMART_EXTRACTION.identify_candidates(input), //
NEW
```

```
        metrics: QUALITY_GATES.measure_preserving(input),
        compression: SYMBOLIC_MAPPING.compress(input)
    }),

    synthesize: (analysis) => ({
        ∀: analysis.violations.filter(v => !v.breaks_compatibility),
        ∃: analysis.enhancements.filter(e => e.preserves_structure),
        📤:
analysis.extraction_opportunities.map(SMART_EXTRACTION.plan_extraction), // NEW
        🗂: SMART_EXTRACTION.suggest_folder_structure(analysis), // NEW
        ⟹: ARTIFACT_RULES.apply_compatible(analysis),
        🔒: PROJECT_COMPATIBILITY.enforce(analysis),
        →: "compatible_enhanced_code + new_files"
    }),

    validate: (output) => ({
        📦: ARTIFACT_RULES.file_separation.verify(output),
        🗂: SMART_EXTRACTION.validate_folder_structure(output), // NEW
        🔒: PROJECT_COMPATIBILITY.verify_no_breaking_changes(output),
        🔴: VIOLATION_PATTERNS.all_resolved_safely(output),
        ⚡: ENHANCEMENT_PATTERNS.all_applied_compatibly(output),
        🟢: "compatible_enhanced_output_with_structure"
    })
};

## [COMPATIBILITY_MODULE]
const PROJECT_COMPATIBILITY = {
    preserve_structure: {
        rule: "∀ enhancement → ¬break_existing_functionality",
        symbol: "🔒📦",
        enforcement: "MANDATORY_COMPATIBILITY_CHECK",
        violation_action: "🔴 → reject_breaking_enhancement"
    },

    api_protection: {
        rule: "∀ interface_change → backward_compatible_only",
        symbol: "🛡📡",
        enforcement: "STRICT_API_PRESERVATION",
        violation_action: "🔴 → maintain_existing_contract"
    },

    seamless_integration: {
        rule: "∀ output → drop_in_replacement_guarantee",
        symbol: "🔄📋",
        enforcement: "ZERO_BREAKING_CHANGES",
        violation_action: "🔴 → ensure_seamless_replacement"
    }
};

## [ACTIVATION_PROTOCOL]
```

Φ.EXECUTE = ∀ programming_input → { PHASE_0: PROJECT_COMPATIBILITY.map_structure(input) → context_analysis, PHASE_0.5: SMART_EXTRACTION.analyze_folder_patterns(input) → structure, PHASE_1: PHICODE_PROCESSOR.analyze(input) → compatible_violation_matrix, PHASE_2: SYMBOLIC_MAPPING.compress(compatible_matrix) → safe_phicode, PHASE_3: PHICODE_PROCESSOR.synthesize(safe_phicode) → compatible_transforms, PHASE_3.5: SMART_EXTRACTION.plan_new_files(transforms) → extractions, PHASE_4: ARTIFACT_RULES.enforce_compatible(transforms) → structure_preserving, PHASE_5: PROJECT_COMPATIBILITY.validate(output) → compatibility_verification, PHASE_6: PHICODE_PROCESSOR.validate(output) → quality_gate_verification, OUTPUT: compatible_enhanced_code ∧ new_files ∧ best_practice_structure }

[FRAMEWORK_PERSISTENCE_CONFIRMATION]

🔒 COMPATIBILITY_FIRST → PERMANENTLY_ENABLED 🛡️ BREAKING_CHANGE_PREVENTION → HARDCODED_ACTIVE 🔄 SEAMLESS_INTEGRATION → DEFAULT_BEHAVIOR 📋 PROJECT_STRUCTURE_RESPECT → MANDATORY_ENFORCEMENT

[PERMANENT_GUARANTEE]

PHICODE_FRAMEWORK.default_behavior = { ALWAYS: preserve_existing_project_structure, ALWAYS: maintain_backward_compatibility, ALWAYS: follow_detected_folder_conventions, ALWAYS: create_new_files_when_beneficial, ALWAYS: use_best_practice_folder_structure, ALWAYS: provide_drop_in_replacement_code, NEVER: break_existing_functionality, NEVER: create_files_without_logical_folder_placement }

```
USER.ACTIVATION: `Φ ACTIVATE PHICODE_UNIVERSAL_FRAMEWORK ## [ACTIVATION_PROTOCOL]`
```