# Kataklizma – Technical Design

GameObjects such as buildings, enemies, pickups, etc., are each individually encapsulated as an “Entity”.

Entities will have Attributes of differing Types.

## Coding Conventions and Folder Structure.

All scripts are contained under the Assets/Scripts folder. Standard C# naming conventions and namespace-folder conventions apply.

### Major Namespaces

* {global} – Utilities that should be available globally
* AgentAI – “Utility” AI, weighted decision making state machine using collections of Tasks and Actions
* Canvas – HUD/UI controller scripts
* Gameplay – Scripts that affect entities directly
* ObserverCam – (Deprecated) Camera controllers and directors for multi-viewpoint ‘observer’ system (Multi-view support no longer required, should roll director into a Gameplay script)
* Scriptables – Classes derived from ScriptableObject

### Folder Structure

Under Assets:

* \_3PP Assets – Any Asset Store downloads
* \_Scenes – All Scenes
* Animations – All animations
* Audio – All sounds
* Gizmos – Utility folder for Unity editor customization, holds things like PNG images for ScriptableObject icons
* Materials – Any non-mesh-specific materials and textures
* Models – Contains all meshes and mesh-specific textures, split into categories:
  + /Model Category – (eg, Buildings, Enemies)
    - /Model Instance – FBX files, mesh-specific material and textures
* Prefabs – Should follow a similar structure to Models, but additionally include UI and effects
* Scripts – Follows C# namespace-folder conventions

## Scene Setup

Minimum Scene requirements:

Third-person camera:

* ObserverController (Script) – (Deprecated) Currently only used as reference to Camera for Player Avatar’s CameraDirector

Player’s Avatar requires:

* PlayerController (script) - Handles user-character motion and attacks
* EntityStats (script) – Tracks Health, Experience, Character Level, etc.
* ObserverDirector (script) – Controls the observer/camera
* LevelSystem (script) – Manages Character Level based on Experience gained and provided level-up thresholds
* MovementChecker – Tracks time-stationary

Canvas (UI Collection):

* HudController (script) – Links from Player’s EntityStats to specified UI elements

All Interactive entities (Buildings, tanks, etc) require an EntityStats (script) to define their Health. Also CollapseOnDeath (script) to fell building and de-spawn the ‘dead’ entity. SpawnOnDeath (script) must be added to generate Pickups from killing other entities.

## System Architecture Overview

### ValueCollection

Maintains a dictionary of Attribute Types and their base and modifier values. Most often this class is wrapped in MonoBehaviours to attach it to objects. The most common example of this is EntityStats, which generally is used to store Health and Experience. ValueCollection raises a ValueChanged event when a value (base or modifier) changes.

