Semantic Design for Nu Game Engine (using Sedela)

let World = (Game : Game)

let Game = (Screens : List<Screen>; Simulant)

let Screen = (Layers : List<Layer>; Simulant)

let Layer = (Entities : List<Entity>; Simulant)

let Entity = (Facets : List<Facet>; Simulant)

let Simulant = (Name : String, Dispatcher : Dispatcher, Properties : Map<String, Any>)

let PropertyDefinition =

(Name : String,

Type : Axiom "A value type.",

Default : Any)

let Event<S :> Simulant> =

(Publisher : Simulant,

Subscriber : S,

Data : Any)

let Dispatcher =

(PropertyDefinitions : List<PropertyDefinition>,

Behaviors : List<Behavior>)

let Facet =

(PropertyDefinitions : List<PropertyDefinition>,

Behaviors : List<Behavior>)

let Behavior<S :> Subscriber> = Event<S> -> World -> World

Semantic Design for Observable Property Bag Simulations (now implemented by Nu)

let PropertyChangeHandler<Key> = Simulation<Key> -> Simulation<Key> -> Simulation<Key>

and PropertyChangeUnhandler<Key> = Simulation<Key> -> Simulation<Key>

and Simulation<Key> = Axiom "A simulation in terms of an observable property bag."

let getPropertyOpt<Key, A> : Key -> Simulation<Key> -> Maybe<A> =

Axiom "Obtain a simulation property associated with the given key if it exists."

let setPropertyOpt<Key, A> : Key -> Maybe<A> -> Simulation<Key> -> Simulation<Key> =

Axiom "Set a simulation property associated with the given key if it exists."

let handlePropertyChange<Key> : Key -> PropertyChangeHandler<Key> -> (PropertyChangeUnhandler<Key>, Simulation<Key>) =

Axiom "Invoke the given handler when a property with the given key is changed."