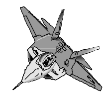


Fundamental Principles of Modeling & Simulation

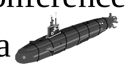
Roger Smith

smithr@modelbenders.com

http://www.modelbenders.com/

1999 Game Developer's Conference

San Jose, California

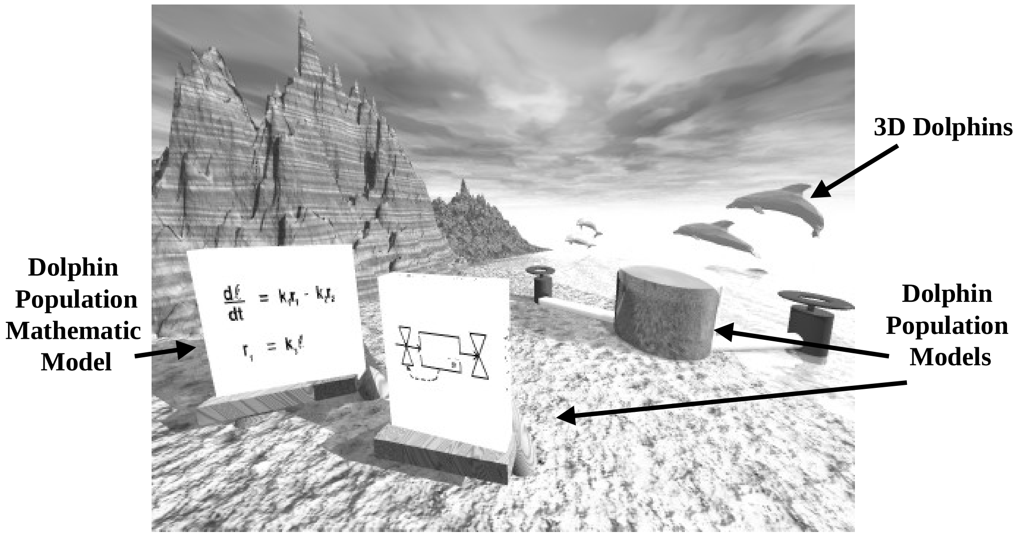


March 17, 1999

Notes

This presentation was given to an audience 160 people in a session at the 1999 Game Developer's Conference sponsored by Miller Freeman Publishers. It is an attempt to isolate universal principles for developing models and simulations for use in multiple domains. The military simulation domain was the original source of the information, but it is equally applicable to the practice of creating games, web spaces, and other digital representations of reality.

Model Abstraction



Notes

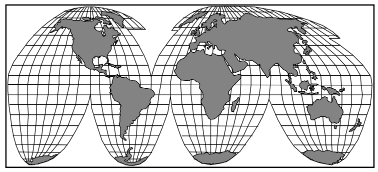
Any system can be represented at multiple levels of abstraction. The most natural and accessible is a 3D visual representation as shown by the jumping dolphins. This is most commonly found in television animation and other visual media.

Abstracting the system also allows us to remove the 3D characteristics of flesh and form to capture behaviors we are more interested in. The Tank-and-Valve on the beach is a representation of the birth and death rates of dolphins. This governs the size of the dolphin population without representing individual reproductive behaviors. The paper easel on the right is a modeling tool representation of the tank-and-valves.

The paper easel on the left is the mathematical definition of the birth and death rates and their effect of the population over time. This mathematical model is more abstract, but also captures behaviors that are much more difficult to represent in 3D.

In general the more abstract a model of a system, the less accessible it is to people - narrowing the audience that can use it or understand it.

Mapping ... Modeling



"Even now...maps are not true pictures of reality. Each map is a product of compromises, omissions, and interpretations. Even a good map tells a multitude of little white lies."

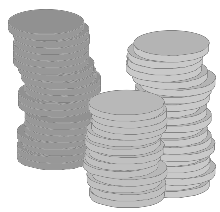
- Mark Monmonier, Syracuse University
National Geographic, February 1998

Notes

No model is a perfect representation of the system it models. A perfect model would be an instance of the system itself. However, these imperfections are completely normal, reasonable, and acceptable. Each model has a purpose which is a sub-set of the purpose or capability of the real system it represents.

Golden Rule of Modeling™

A model has no inherent value of its own.
The value of a model is based entirely upon the degree to which it solves someone's real-world problem.



Notes

This is one of the most important concepts you will encounter in this class and the key to creating a good model. If this principle is violated all the brilliant work in the world can not create a good model. If you want a silk purse you do not begin with a sow's ear.

Axioms to the Golden Rule

- Models are not universally useful, but are designed for specific purposes
  - Different Problems Require Different Models
  - Yesterday's Hit is Tomorrow's Dinosaur
  - Hundreds of Problems Imply Hundreds of Models
  - Code Reuse Usually Sucks
  - Steal Ideas not Code

Notes

The primary reason for creating a new model is that the existing models are not sufficient for the job. Therefore, it is unlikely that extensive reuse of existing models will generate a new and improved solution to the problem.

Taken to the extreme, code reuse is the equivalent of making a fresh copy of the old simulation executable. You would think that if code reuse is good, then just running a copy of the old system is the best solution :)

Axioms to the Golden Rule

- A great model for the wrong problem will never be used
  - The Soviet demise destroyed hundreds of DOD simulations
  - Valuable models fit the customer's problem, not the programmer's preferences
  - What does the customer want?

Notes

Every simulation addresses a specific set of problems and makes fundamental assumptions about those problems. Therefore, a good model for one problem can easily be a terrible model for another problem.

As an example, most combat simulations assume that the enemy will fight to the death with great tenacity. These models internalize the assumption that the conflict is a traditional NATO-Soviet confrontation in Europe. These models can not correctly portray a Desert Storm scenario. There is no software for surrendering or fleeing in mass numbers.

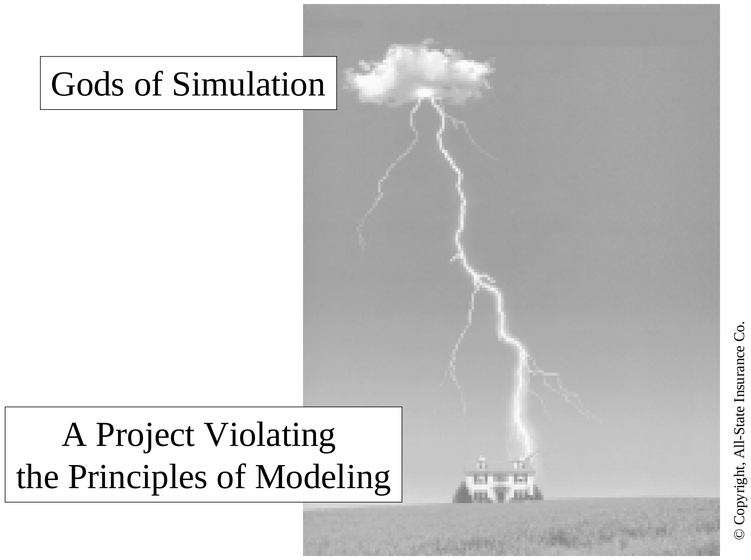
Axioms to the Golden Rule

- Learning to model is better than learning about models
  - Historians and librarians have great references and clues
  - Visionaries see the past, but create the future
  - BB Brains rant on and on about how they did it last time
  - Visionaries invent what BB Brains will rant about next year

Notes

Historians are essential, but only visionaries can create the next revolution in models.

Principles of Modeling



Any Questions?

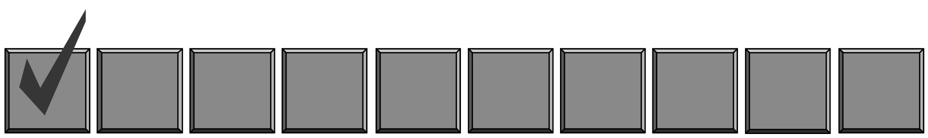
Notes

Model developers have learned and cataloged some very valuable lessons for successfully creating models and simulation systems. Ignoring these lessons is a bad idea and leaves your project open to unnecessarily repeating the mistakes of the past.

Principles of Modeling

① Simplify, Simplify

- “essentia non sunt multiplicanda praeter necessitatem”
- “hypotheses are not to be complicated without necessity”
    - Occam’s Razor, 1320AD - Sir William of Occam
- “Everything should be made as simple as possible - but no simpler!”
    - Albert Einstein



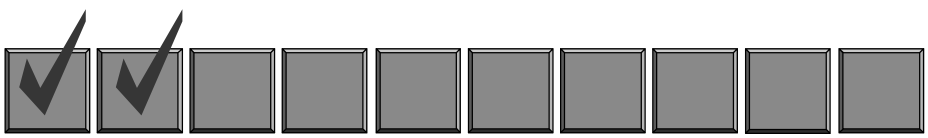
Notes

Everyone from the customer to the programmer is eager to add as many interesting and impressive details as possible. Each will profess eloquently that each entail is an essential element of the simulation. Creating a model that is appropriately detailed and efficiently simple requires real professional judgement and critical examination.

Principles of Modeling

② Learn from the Past

- ➢ Know and use existing theory of modeling
- ➢ Study previous models of similar systems
- ➢ Rely on experienced people in the field



Notes

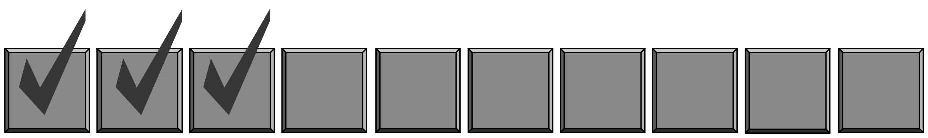
The first wargame was built in 1664, the first flight simulator in 1930, the first virtual world in the 1960's. The simulation community has wrestled with problems similar to yours and arrived at very valuable solutions. You need to learn from all of the mental and physical work that has gone into previous systems.

"Those who do not know the past are destined to repeat it."

Principles of Modeling

③Create a Conceptual Model

- ➢ Large models are too complex to understand without mentally digging into them
- ➢ Use design process to identify important factors
- ➢ Identify black holes, inconsistencies, waste



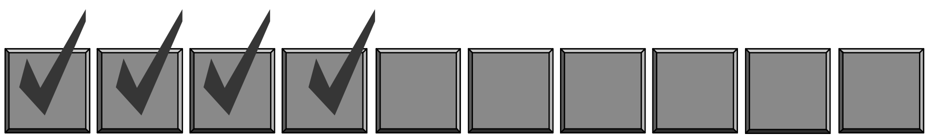
Notes

A virtual world is a very complex thing. It is difficult to envision all of the pieces and interactions that will exist in it. A formal conceptual model captures the elements, event, modifiers, and functions that will operate in the system. Without this the model will turn out lop-sided at best and inoperable at the worst.

Principles of Modeling

④Build a Prototype

- “Large successful systems come from small successful systems.”
  - Bill Joy, Sun Microsystems
- Aim prototype at the 80:20 rule
  - 80% of the functionality is 20% of the code



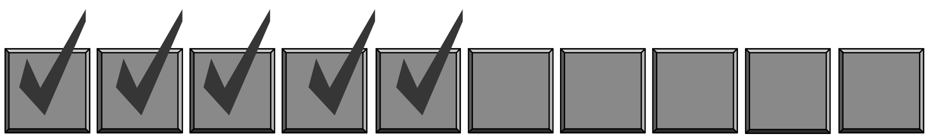
Notes

Running model spends 80% of its time executing 20% of the code written. A prototype should dig into this essential 20%, demonstrate it to the user, allow modification and refinement, and win trust in the system that will result.

Principles of Modeling

⑤Push the User’s Hot Buttons

- Identify all of the “users” who must be satisfied
- Observe the real system to be modeled
- Give the model “face validity”



Notes

Each user has special Hot Buttons that have to be pushed. You should have uncovered these in your “Know the User’s Needs” phase. These need to be highlighted in order to make their Green Lights go on.

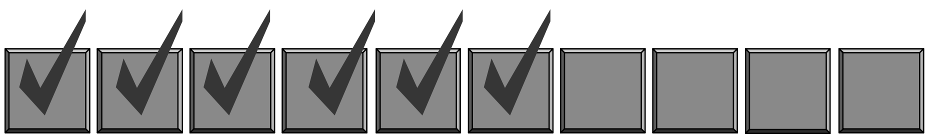
Spend time listening to the users. Do not rely solely on their written specifications. Listen to their stories about past experiences, successes, and failures. Come to understand WHY they are asking for certain features. This will be a huge help when you reach the phase of determining how to satisfy these requirements in the conceptual model and the software.

If you have a valid model, it is very difficult to achieve acceptance if, on the surface, it appears counter-intuitive or impossible to understand. This is one reason for the explosion in 2D and 3D displays of internal model operations - maps, virtual reality, pie charts, histograms, etc.

Principles of Modeling

⑥Model to Data Available

- Data is specialized and scarce
- Data is not available to support all models
- List useful data you do have access to
- Laws of Data
  - Availability, Quality, Collection, Synthesis



Notes

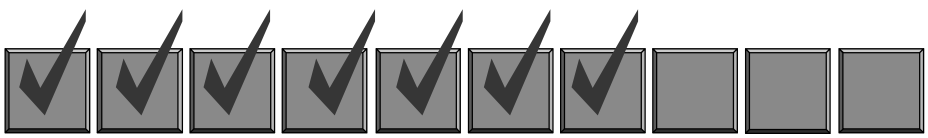
Each military service has several organizations that are responsible for collecting, storing, and distributing data that is valuable to model developers and simulation sponsors. One such organization is the Center for Army Lessons Learned (CALL). This is part of the Army Training and Doctrine Command (TRADOC) - http://www.tradoc.army.mil/

Data is available for the most common cases of equipment, performance, causal results - but never for all of the combinations that will appear in a model.

Principles of Modeling

⑦Separate Data from Software

- Good Software Engineering
- Data separation is one key to flexibility, extensibility
- People want to change the darndest things



Notes

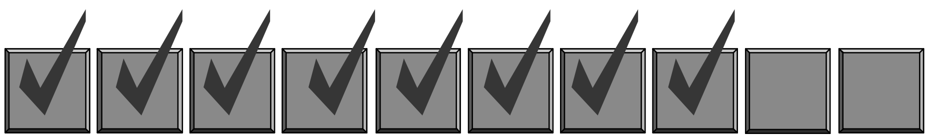
Every project has a horror story about hard-coding data into software. Then months later the customer asks to change those characteristics. As a result the modeler has to rework large amounts of code, spending unnecessary money and time.

Hard-coded data values range from time step, frame rate, probability of kills, environmental conditions, lighting conditions, behavioral assumptions, etc. etc. etc.

Principles of Modeling

⑧Trust Your Creative Juices

- Modeling is a creative process, not just science and engineering
- Creativity and energy are primary ingredients of a simulation
- Not following your creativity will stagnate the project



Notes

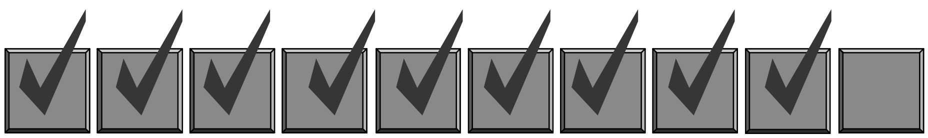
When working with a new team that has not created a simulation before, they are afraid to move forward without explicit direction and definition about what they should build. They are afraid that they will head off in the wrong direction and create a product that others will criticize. This fear of criticism is more crippling than their aversion to reworking a program that has gone wrong. Experienced members of the team must demonstrate, instill, and encourage the brave act of trusting your own creative juices. The team leaders must provide the vision for the entire product, but each programmer, designer, and artists must have the freedom and confidence to express what they see in the product.

This fear of making mistakes results in constant revisits and repetition of details in requirements analysis, organizational restructuring, product research, process definition, etc. The team avoids making concrete decisions about the design of the product. They will not allow programmers to finish a conceptual model or build a prototype. Thousands of man-hours can be wasted in this trap. But eventually this cycle will be broken by the arrival of a competent leader, the disobedience of talented designers and programmers, an imminent deadline, or cancellation of the project.

Principles of Modeling

⑨Fit Universal Constraints

- ➢ Quality of product
- ➢ Time to deliver
- ➢ Money to spend
- ➢ Team competency



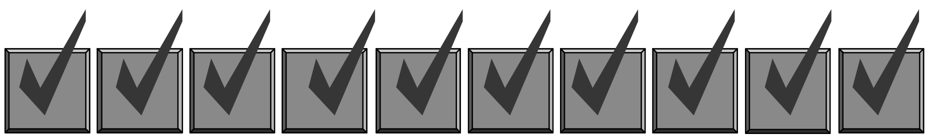
Notes

Project Mangers have long recognized that there is a constant tug-of-war between Quality, Time, and Money. Any time one is in short supply the others must compensate. However, in technical fields today there is a fourth constraint. Team Competency is just as restrictive as the others. Competency is in high demand and scarce supply, and it requires long lead-times to build-up. It may be impossible to build a system because of the lack of human resources, even when abundant time and money are available.

Principles of Modeling

⑩Distill Your Own Commandments

- Commandments are invaluable guides
- Your experience contains the commandments most applicable to your situation



Notes

The commandments provided in this course are principles learned from the experience of multiple simulation developers. These experiences are invaluable and, if followed, will improve the performance of projects and individuals. However, these are not the only principles. Each person has a pool of experience that can provide principles custom fitted for your situation. You should take the time to capture principles buried in your own experiences. This is an ongoing process that provides new principles over many months and years. It also replaces principles that are no longer applicable or that were poorly conceived at an earlier time.

Pritsker's Modeling Principles

- Conceptualizing a model requires system knowledge, engineering judgement, and model-building tools.
- The modeling process is evolutionary because the act of modeling reveals important information piecemeal.
- The secret to being a good modeler is the ability to remodel.
- The problem or problem statement is the primary controlling element in model-based problem solving.
- In modeling combined systems, the continuous aspects of the problem should be considered first. The discrete aspects of the model should then be developed.
- A model should be evaluated according to its usefulness. From an absolute perspective, a model is neither good or bad, nor is it neutral.
- The purpose of simulation modeling is knowledge and understanding, not models.

Notes

These seven principles of modeling are derived by Alan B. Pritsker of Pritsker Corporation. He is one of the early developers of simulation specific languages and commercial simulation products. A more detailed description of each of these can be found in the *Handbook of Simulation*, edited by Jerry Banks for Wiley Interscience, 1998.

Creating a Digital World

Notes

Simulations are a digital version of some aspects of the real world. We seek to capture only those details that are important for the purposes of the simulation, and to represent these with models that interoperate with each other in a consistent manner.

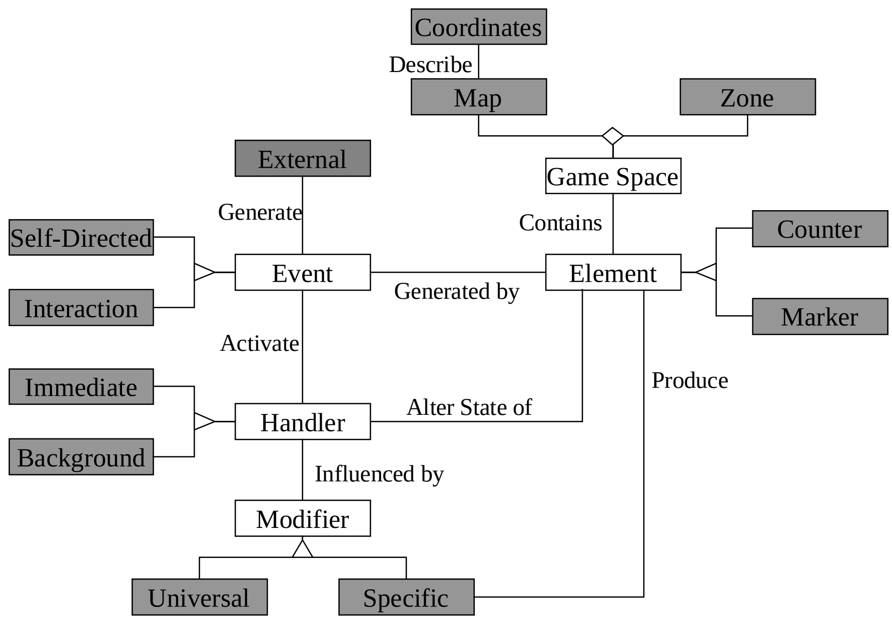
The digital world is always a sub-set of the real world.

The digital world is always an approximation of the real world.

A perfect model of the real world would be another real world.

A simulation contains variables that can be set many ways, only some of which match the real world. Therefore, the simulation itself is more complex than the real world for those variables that it includes. Though the simulation is a sub-set of the real world, it is a more complex sub-set.

Abstract Object Model



Notes

This is an object diagram (using the UML notation) of one potential abstract model. This model is directed specifically at military conflict simulations and supports either constructive or virtual representations.

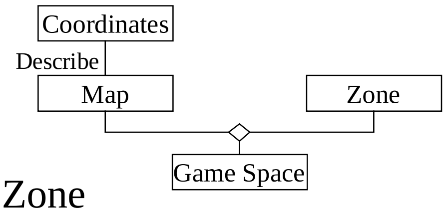
The model centers around five major components: the Element, Event, Handler, Modifier, and Game Space. The Element is the object in the simulation - vehicle, person, unit, etc. Elements generate Events as a natural product of their operations - movement, detection, engagement, plan development, etc. Handlers accept these stated events and compute their meaning and effects in the simulation. Modifiers influence the inputs to a handler or the outputs resulting from one. The results generated by the handler impact some set of Elements in the simulation.

The Game Space is a container that holds all of the elements and binds them together into an interacting environment. The most common representation of Game Space is a map or virtual world in which the elements exist.

Modifiers are divided into two categories. Specific Modifiers are generated by the actions of Elements. Universal Modifiers are a result of the Game Space or interference from external controllers.

Game Space

❑Date, Time

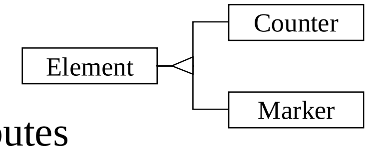


- ❑ Map
  - ➢ Represents location, position, condition
  - ➢ Geographic Coordinates
  - ➢ Terrain, Ocean, Air
  - ➢ Perhaps Weather
- ❑ Zone
  - ➢ Generic areas of influence or effects that are not explicitly modeled
  - ➢ Air Defense Lethality
  - ➢ Holding Bins
    - ❍ Aircraft Crossing Ocean
    - ❍ Aircraft Position on Carrier

Notes

The entire Game Space usually holds general information like the date and time of the entire simulation. Maps and Zones contain or represent more detailed information about the battlespace. A map is generally related to the physical environment in which the simulation occurs. A zone is a general concept that is not modeled in detail, but is represented to include the effects of the events and elements that exist outside of the map.

Elements



- Classes
  - Equipment (C)
    - Assets Controlled by Thinkers
  - Thinker (C)
    - Planning & reasoning
  - Association (M)
    - Logical relationships
    - C3, factions, & sides
  - Natural Feature (M)
    - Sea state, weather, forests
  - Influence (M)
    - Socio-economic, ideological, morale, emotion
  - Region (M)
    - Borders, controls, phase lines
- Attributes
  - Identity (Name and Capability)
  - Affiliation
    - Command, Reporting, Logistical
  - Perception
  - Kinematics
  - Health
    - Damage State
  - Wealth
    - Logistical State
  - Appearance
    - Signature
  - Intention
    - Objectives and goals
  - Activity
    - Current actions in progress

Notes

Elements (both Counters and Markers) generally fall into one of the classes described above. These classes of elements have attributes to represent capabilities, assets, and actions.

The lists shown here are meant to be comprehensive. An abstract model tries to provide all of the tools for representing the battlespace, but does not dictate how the simulation designer will use these tools. Therefore, the classes and attributes may be combined in many overlapping patterns to accomplish the goals of the designers.

Event Types

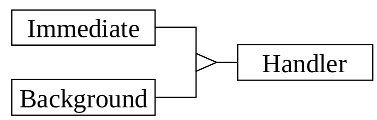
- Physical
  - Change in motion state, signal emissions, explosions
- Decision
  - Changes in Perception, Intention, Activity
- Information Exchange
  - Exchanges of state information, not necessarily messages
- Interchange
  - Interactions involving more than one element (combat, collision, resupply)
- Environmental
  - Changes in the state of the game space
- Game State Update
  - Changes in the state of “global information”
- External
  - Events directed to or received from outside the simulation
- Element Coordination
  - Events required to communicate among model components
- Simulation Control
  - Stop, start, checkpoint, resume

Notes

Events fall into one of the categories described above. These categories are very broad in an attempt to be comprehensive. If an abstract model is too specific it tends to become very, very large and dictatorial to the simulation designers - which is not the purpose of an abstract model.

Handler Types

- Lookups
  - Retrieval from data tables, parameter tables, game state
- Conditionals
  - Logical tests against conditions
- Branches
  - Explicit choice in decision or operation flow
- Calculations
  - Mathematical function calls
- Random Draws
  - Generation of pseudo-random values
- Orderings
  - Explicit sequencing of sub-sequences
- Consequences
  - Secondary effects of other events

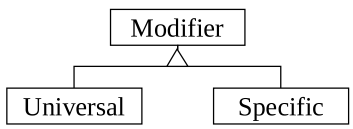


Notes

Handlers are usually implemented in computer programs as object methods or functions. These are the software engines that drive a simulation forward and make combat activities happen. These Handlers fall into the categories described above.

Modifier Types

- Limitations / Restrictions
  - Performance modifiers
- Propagate
  - Conditional Rules to Trigger Consequences
- Cancel
  - Eliminate Event
- Stochastic
  - Input/Output Modifier
- Delay / Advance
  - Change time of event or effect
- Logic Change
  - Reroute decisions or events
- Overwrite
  - Change outcome



Notes

Modifiers are generated by Elements or controllers and automatically change data values or outcomes from or within a Handler. These Modifiers can be categorized as shown above.

Modeling Process

1. **Describe It**
   What is it for? Why build it?
2. **Touch a Real One**
   Size, color, connections, etc.
3. **Use a Real One**
   Feel, motion, process
4. **Define the Interactions**
   Input, output, dependencies
5. **Define State Variables**
   Physical, functional, behavioral
6. **Define Hot Buttons**
   What sells?
7. **Code the Interface**
   Isolate methods, data
8. **Build Core Code**
   Capture breadth, stub details
9. **Dig Deeper**
   Code details, modular structure
10. **Rework**
    Throw out bad ideas, add detail
11. **Field Test**
    Test under fire
12. **Better Next Time**
    Learn lessons, move on

Notes

There are twelve guiding steps to building a model of a system. These are described above as distilled from the experiences of many people in the field.

Conclusion

- Military and game modeling have a common goal
  - Abstracting the real world and animating the virtual world
- Common principles for good modeling do exist
- You can improve productivity and reduce failure by learning from the experiences of those before you

Notes