Modeling & Simulation (M&S) Certification Study Guide
7 September 2015
Table of Contents
1. Introduction
2. General Study Tips
3. What to expect when you take the Exam
4. Section 1—General Modeling & Simulation Terms (Vocabulary)
5. Section 2—Coding, Programming & Gaming
6. Section 3—Visualization, 3D Modeling & Animation
7. Section 4—Ethics, Entrepreneurship & Innovation
8. Section 5—Simulations & Simulators
9. Section 6—Project Management & Logistic Support
1. Introduction

A Modeling and Simulation (M&S) Certification for High School Students is important, especially in the era of expanding applications of simulation, animation and gaming. With many new jobs being created in M&S, the workforce pipeline must be filled with skilled people. M&S growth is worldwide with positions available at all levels and a wide range of skills. Those high school students with a Science, Technology, Engineering and Mathematics (STEM) education are positioned very well for internships and entry level jobs in the industry. The simulation industry needs engineers, graphic artists, education specialists, psychologists, contract administrators, technicians, programmers and many other disciplines to support a wide range of applications which is growing on a day-by-day basis.

The underlying rationale for certification is to provide a qualified workforce able to support growing demands. The “supply” side for meeting these requirements is affected by: a) the number of people in the workforce who are retiring or eligible to retire within the next five years, b) the number of college graduates ready to enter the STEM workforce is at a new low and c) the rapidly expanding use of simulation into many applications.

This certification program benefits the M&S industry by creating student interest and a level of competency in STEM disciplines, and preparing students for M&S careers in defense, medical, entertainment, business, education, training and other fields. The National Center for Simulation (NCS) M&S Certification Program provides the industry with a pool of skilled and motivated students to fill paid internship positions. Additionally, students who are participants in the NCS M&S Certification Program are better prepared and are more qualified as candidates for full time employment in the M&S industry, requiring less entry level training and improving overall productivity. and company competitiveness is increased because the interns can assume low level STEM work that was formerly performed by entry level personnel.

NCS, in coordination with the Florida Departments of Education (FDOE) and Economic Opportunity (DEO), Career Source Florida, Inc. (CSFL), Career Source Central Florida (CSCFL), and Orange County and Seminole County Public Schools, developed this industry certification program in M&S for high school (and other) students. This certification program is aligned with the NCS high school M&S curriculum, a four year program preparing students for post-secondary education or entry level positions in industry or government. Students who complete this curriculum and those who have appropriate experience and/or other training may elect to take the NCS M&S Certification Exam. This Study Guide will help the students to pass the exam.

Passing this exam and achieving certification demonstrates the student’s proficiency in the fundamental skill set in M&S, thus providing the student with internship and scholarship opportunities, and facilitates the student to enter post-secondary education and / or to seek employment in industry or government. By enrolling in the NCS M&S Certification Program, successfully passing the exam and taking advantage of internship opportunities, students are
prepared for rewarding, high paying careers in an exciting high tech industry.

Lt Gen (Ret) Tom Baptiste, NCS President and Executive Director, added “This certification program creates a pool of skilled and motivated students in Central Florida who will be ready for recruitment in internships. Because of this program, these students will be better prepared and more qualified as candidates for full time employment in the M&S industry.”
2. General Study Tips

To prepare for the certification exam, use this study guide and consider the study tips provided below:

- Find a study buddy or form a study group
- Try to study every day
- Avoid cramming
- Spend more time on your weak areas
- Pick a time to study when you are most alert. Research shows that the morning and early evening are best.
- Stick to the study schedule
- Keep breaks short: 3-5 minutes is adequate
- Study one topic at a time
- The next day review what you learned the day before
- Make outlines and map out what you learned and develop questions from your outline
- Reading out loud helps reinforce the material
- Flash cards or index card may be helpful

Make sure you go to the NCS website to access the M&S Curriculum:

http://www.simulationinformation.com/modeling-and-simulation-certification

Note: Units shown in ( ) are Curriculum Numbers

3. What to expect when you take the Exam
The certification exam consists of 50 questions, including multiple choices, true/false, and matching questions. Some questions will involve calculations to determine the answer. While you will be allowed to bring a calculator to the exam, you will NOT be allowed to have Internet access through mobile phones, laptops, tablets, or any other device.

The certification will be timed. You will have 55 minutes to complete the exam. Any questions that are not answered are considered incorrect.

The certification exam questions are written to test your critical thinking skills. There are certainly many recall or memory items, but the majority of questions will have you reflect on what you have learned in your high school curriculum and come to a conclusion with the right answer. If you like to memorize data, try to not only memorize the facts but think of an example, so you really know it. If you do not understand the underlying material, then ask your teachers for an explanation and an example to help with your understanding.

The fifty (50) questions are randomly selected from a bank of questions that test your knowledge in the following six (6) areas:

Section 1—General Modeling & Simulation Terms (Vocabulary)
Section 2—Coding, Programming & Gaming
Section 3—Visualization, 3D Modeling & Animation
Section 4—Ethics, Entrepreneurship & Innovation
Section 5—Simulations & Simulators
Section 6—Project Management & Logistic Support
Section 1—General Modeling & Simulation Terms (Vocabulary)

1. Augmented Reality (Unit 3)
   a. An enhanced image or environment as viewed on a screen, head-mounted, or other display, produced by overlaying computer-generated images, sounds, or other data on a real-world environment.

2. Business Ethics (Unit 9)
   a. A form of applied ethics or professional ethics that examines ethical principles and moral or ethical problems that arise in a business environment. It applies to all aspects of business conduct and is relevant to the conduct of individuals and entire organizations.

3. Client/Server Model
   a. In the client-server model of computing, tasks are distributed between a service provider, called a server, and a service requester, or client. The client-server model was widely used in the early days of computing, when access to expensive mainframe computers was provided by dumb terminals. More recently, the client-server model proved crucial to the development of the Internet. There are several models for client-server interaction, with the main difference being the amount of work performed on each side.
   b. A core network computing concept for email exchange and web/database; protocols built around the client-server.
      i. HTTP (hypertext Transfer Protocol)
      ii. DNS (Domain Name System)
      iii. SMTP (Simple Mail Transfer Protocol)
      iv. Telnet (terminal emulation program that connects a PC to a server on the network)

4. Discrete Event Model Simulation (Units 1 & 5)
   Models and simulates the operation of a system as a discrete sequence of events in time. Each event occurs at a particular instant in time and marks a change of state in the system. It is a valuable analysis tool for improving system effectiveness and efficiencies. Example: Model a queue, such as customers arriving at a bank to be served by a teller.

5. Environmental Model (Unit 3)
   A numerical model, parametric model, or database designed to produce an accurate and consistent data set for one or more parameters that characterize the state of the natural environment. The environmental simulation is a simulation that depicts all or part of the natural or manmade environment of a system; for example, a simulation of the radar equipment and other tracking devices that provide input to an aircraft tracking system. It includes weather, terrain, sea states and all natural conditions.

6. Functional Fidelity (Unit 5)
The degree to which the simulated functions coincide with functions of the parent system (aircraft, railroad train, ship, etc.) Functional fidelity does not require the simulator to look exactly like the parent system whereas “physical fidelity” does require a “look alike”.

7. Game Theory (Unit 7)

*Game theory* is the study of strategic decision making. Specifically, it is "the study of mathematical models of conflict and cooperation between intelligent rational decision-makers." An alternative term suggested "as a more descriptive name for the discipline" is *interactive decision theory*. Game theory is mainly used in economics, political science, and psychology, as well as logic, computer science, and biology. The subject first addressed zero-sum games, such that one person's gains exactly equal net losses of the other participant or participants. Today, however, game theory applies to a wide range of behavioral relations, and has developed into an *umbrella term* for the logical side of decision science, including both humans and non-humans (e.g. computers, animals).

8. High Level Architecture (HLA) (Unit 3)

A general purpose architecture that allows computer simulations to communicate with each other regardless of the platform or geographic locations. For military applications, flight simulators can operate with shipboard simulators and ground force simulators together on the same simulated environment.

9. Incubation (Business) Phase (Unit 9)

Business incubation provides a nurturing, instructive and supportive environment for entrepreneurs during the critical stages of starting up a new business. The goal of incubators is to increase the chance that a start-up will succeed, and shorten the time and reduce the cost of establishing and growing its business. If successful, business incubators can help to nurture the companies that will form the true creators of a region’s or nation’s future wealth and employment.

10. Integrated Logistics Support (ILS) (Unit 11)

The composite of the elements of support necessary to assure the effective and economical support of a system or equipment at all levels of maintenance for its programmed life cycle. The elements of support are as follows:

1. Design Interface (Reliability, Maintainability and Availability)
2. Maintenance Planning
3. Manpower and Personnel
4. Training and Training Support
5. Technical Data and Publications
6. Packaging, Handling and Transportation
7. Computer Support Resources
8. Site Preparation and Facilities
9. Support and Test Equipment
10. Supply Support
11. Logistics Information

11. Integrated Product Team (IPT) (Unit 10)

Multidisciplinary group of people who are collectively responsible for delivering a defined product or process. IPTs are used in complex development programs/projects for review and decision making. The emphasis of the IPT is on involvement of all stakeholders (users, customers, management, developers, and
contractors) in a collaborative forum. IPTs are created most often as part of structured systems engineering methodologies, focusing attention on understanding the needs and desires of each stakeholder.

12. Intellectual Property (Unit 9)
A legal term that refers to creations of the mind. Intellectual property can be protected through patents, trademarks or copyrights.

13. Link Blue Box (Unit 1)
The first modern pilot trainer invented by Edwin Link of Binghamton, NY. The trainer, pneumatically operated, was used extensively in World War II to train US and allied pilots.

14. Live Simulation (Unit 1)
A type of simulation that typically involves humans and/or equipment and activity in a setting where they would normally operate.

15. Mathematical Model (Unit 11)

Mathematical models are usually composed of relationships and variables. Relationships can be described by operators, such as algebraic operators, functions, differential operators, etc. Variables are abstractions of system parameters of interest, that can be quantified. Several classification criteria can be used for mathematical models according to their structure:

- **Linear vs. nonlinear:** If all the operators in a mathematical model exhibit linearity, the resulting mathematical model is defined as linear. A model is considered to be nonlinear otherwise. The definition of linearity and nonlinearity is dependent on context, and linear models may have nonlinear expressions in them. For example, in a statistical linear model, it is assumed that a relationship is linear in the parameters, but it may be nonlinear in the predictor variables. Similarly, a differential equation is said to be linear if it can be written with linear differential operators, but it can still have nonlinear expressions in it. In a mathematical programming model, if the objective functions and constraints are represented entirely by linear equations, then the model is regarded as a linear model. If one or more of the objective functions or constraints are represented with a nonlinear equation, then the model is known as a nonlinear model. Nonlinearity, even in fairly simple systems, is often associated with phenomena such as chaos and irreversibility. Although there are exceptions, nonlinear systems and models tend to be more difficult to study than linear ones. A common approach to nonlinear problems is linearization, but this can be problematic if one is trying to study aspects such as irreversibility, which are strongly tied to nonlinearity.

- **Static vs. dynamic:** A dynamic model accounts for time-dependent changes in the state of the system, while a static (or steady-state) model calculates the system in equilibrium, and thus is time-invariant. Dynamic models typically are represented by differential equations.

- **Explicit vs. implicit:** If all of the input parameters of the overall model are known, and the output parameters can be calculated by a finite series of computations (known as linear programming, not to be confused with linearity as described above), the model is said to be explicit. But sometimes it is the output parameters which are known, and the corresponding inputs must be solved for by an iterative procedure, such as Newton's method (if the model is linear) or Broyden's method (if non-linear). For example, a jet
engine’s physical properties such as turbine and nozzle throat areas can be explicitly calculated given a design thermodynamic cycle (air and fuel flow rates, pressures, and temperatures) at a specific flight condition and power setting, but the engine's operating cycles at other flight conditions and power settings cannot be explicitly calculated from the constant physical properties.

- **Discrete vs. continuous:** A discrete model treats objects as discrete, such as the particles in a molecular model or the states in a statistical model; while a continuous model represents the objects in a continuous manner, such as the velocity field of fluid in pipe flows, temperatures and stresses in a solid, and electric field that applies continuously over the entire model due to a point charge.

- **Deterministic vs. probabilistic (stochastic):** A deterministic model is one in which every set of variable states is uniquely determined by parameters in the model and by sets of previous states of these variables; therefore, a deterministic model always performs the same way for a given set of initial conditions. Conversely, in a stochastic model—usually called a "statistical model"—randomness is present, and variable states are not described by unique values, but rather by probability distributions.

- **Deductive, inductive, or floating:** A deductive model is a logical structure based on a theory. An inductive model arises from empirical findings and generalization from them. The floating model rests on neither theory nor observation, but is merely the invocation of expected structure. Application of mathematics in social sciences outside of economics has been criticized for unfounded models. Application of catastrophe theory in science has been characterized as a floating model. 

16. Mean Time Between Failure (MTBF) (Unit 11)
A way to quantify and measure how reliable a product or component, usually measured in tens of thousands of hours. It is the number of hours accumulated by a system or equipment divided by the number of failures experienced over that period of time.

17. Mean Time To Repair (MTTR) (Unit 11)
A way to quantify and measure the maintainability of a product or component. MTTR is calculated by measuring the time to complete maintenance actions over a period of time, adding them up and calculating the mean value in minutes or hours.

18. Mixed Reality
The merging of both real and virtual worlds to produce new environments and visualizations where physical and digital objects co-exist and interact in real time.

19. Objected Oriented Design (Unit 7)
The process of planning a system of interacting objects for the purpose of solving a software problem. It is one approach to software design. An object contains encapsulated data and procedures grouped together to represent an entity.

20. Objected Oriented Programming (Unit 7)
Programming language model organized around objects rather than “actions” or data rather than logic.

21. Peer-to-Peer Model
A decentralized communications model where each party have similar capabilities acting as both client and server on a shared network and either party can imitate a communication session.

22. Physical Fidelity (Unit 5)
   A degree to which device looks, sounds, and feels like actual an environment.

23. Physical Model (Unit 3)
   A smaller or larger physical copy of an object.

24. Process Model
   Process of the same structure that are classified together into a model.

25. Project Management (Unit 10)
   The process and activity of planning, organizing, motivating, directing and controlling resources, procedures and protocols to achieve specific goals in scientific or daily problems. A project is a temporary endeavor designed to produce a unique product, service or result with a defined beginning and end (usually time-constrained, and often constrained by funding or deliverables) undertaken to meet unique goals and objectives, typically to bring about beneficial change or added value. The temporary nature of projects stands in contrast with business as usual (or operations), which are repetitive, permanent, or semi-permanent functional activities to produce products or services. In practice, the management of these two systems is often quite different, and as such requires the development of distinct technical skills and management strategies.

26. Prototype
   Original model like a sample on which to base future designs.
   (A company designing a new toaster will first design and build a prototype, to test it out and see if it’s any good or not. The same is true for simulators where a prototype is built before going to production.)

27. Quality Assurance
   A way of preventing mistakes or defects in manufactured products and avoiding problems when delivering solutions or services to customers; which ISO 9000 defines as "part of quality management focused on providing confidence that quality requirements will be fulfilled". This defect prevention in quality assurance differs subtly from defect detection and rejection in quality control, and has been referred to as a shift left as it focuses on quality earlier in the process.

28. Requirements Traceability Matrix (RTM)
   A sub-discipline of requirements management within software development and systems engineering. Requirements traceability is concerned with documenting the life of a requirement and providing bi-directional traceability between various associated requirements. It enables users to find the origin of each requirement and track every change that was made to this requirement. For this purpose, it may be necessary to document every change made to the requirement.

29. Reliability, Availability and Maintainability (RAM) (Unit 11)
   Reliability is the probability that an engineering system will perform its intended function satisfactorily (from the viewpoint of the customer) for its intended life under specified environmental and operating conditions. Maintainability is the probability that maintenance of the system will retain the system in, or restore it to, a specified condition within a given time period. Availability is the probability that the system is operating satisfactory at any time,
and it depends on the reliability and the maintainability. Hence, the study of probability theory is essential for understanding the reliability, maintainability and availability of a system.

30. Risk Mitigation (Unit 10)
The identification, assessment, and prioritization of risks (defined in ISO 31000 as the effect of uncertainty on objectives) followed by coordinated and economical application of resources to minimize, monitor, and control the probability and/or impact of unfortunate events or to maximize the realization of opportunities. Risk management’s objective is to assure uncertainty does not deflect the endeavor from the business goals.

31. Serious Games (Unit 7)
Games that simulate real-world events or processes to solve problems. These games can be used for law enforcement, medical applications and other simulations.

32. Simulation (Unit 1)
The imitation of the operation of a real-world process or system over time. The act of simulating something first requires that a model be developed; this model represents the key characteristics or behaviors/functions of the selected physical or abstract system or process. The model represents the system itself, whereas the simulation represents the operation of the system over time.

Simulation is used in many contexts, such as simulation of technology for performance optimization, safety engineering, testing, training, education, and video games. Often, computer experiments are used to study simulation models. Simulation is also used with scientific modeling of natural systems or human systems to gain insight into their functioning. Simulation can be used to show the eventual real effects of alternative conditions and courses of action. Simulation is also used when the real system cannot be engaged, because it may not be accessible, or it may be dangerous or unacceptable to engage, or it is being designed but not yet built, or it may simply not exist.

Training simulations typically come in one of three categories (MS-101 You Tube).

i. Live Simulations involve real humans and/or equipment and activity in an actual setting, for example, soldiers practicing in the field as part of an exercise. Time is continuous, as in the real world. Another example of live simulation is a disaster response team responding to mock emergency. In this scenario the people and equipment are real but the situation is simulated.

ii. Virtual simulations involve real humans with equipment that is partially, or completely, simulated. Examples include, a human operating a flight simulator, driving simulator, or nuclear power-plant simulator.

iii. Constructive Simulations involve humans and equipment that are both simulated. Most military strategic simulations are constructive. Other examples include crowd evacuation, disease spread, and weather predictions models. A constructive simulation for weather could anticipate the path of a hurricane using variables such as current and changing temperature, pressure, wind current and other weather factors. In summary, the differences among live, virtual, and constructive can be summarized as follow: Live: real people – real equipment. Virtual: real people –
33. Synthetic Modeling

Constructed from extant, autonomous software components whose existence and purpose are independent of the underlying model they comprise.

34. Systems Engineering (Unit 8)

An interdisciplinary field of engineering that focuses on how to design and manage complex engineering systems over their life cycles. Issues such as requirements engineering, reliability, logistics, coordination of different teams, testing and evaluation, maintainability and many other disciplines necessary for successful system development, design, implementation, and ultimate decommission become more difficult when dealing with large or complex projects. Systems engineering deals with work-processes, optimization methods, and risk management tools in such projects. It overlaps technical and human-centered disciplines such as control engineering, industrial engineering, software engineering, organizational studies, and project management. Systems engineering ensures that all likely aspects of a project or system are considered, and integrated into a whole.

35. Technology Entrepreneurship (Unit 9)

An investment in a project that deploys specialized individuals and assets that are intricately related to advances in specific and technological knowledge for the purpose of creating and capturing value for a firm.

36. Virtual Reality

A hypothetical three-dimensional visual world created by a computer; user wears special goggle and fiber optic gloves etc., and can enter and move about in this world and interact with objects as if inside it.

Can be referred to as immersive multimedia or computer-simulated life, replicates an environment that simulates physical presence in places in the real world or imagined worlds. Virtual reality can recreate sensory experiences, which include virtual taste, sight, smell, sound, and touch.

Most up-to-date virtual reality environments are displayed either on a computer screen or with special stereoscopic displays, and some simulations include additional sensory information and emphasize real sound through speakers or headphones targeted towards VR users. Some advanced, haptic, systems now include tactile information, generally known as force feedback in medical, gaming and military applications. Furthermore, virtual reality covers remote communication environments which provide virtual presence of users with the concepts of telepresence and tele-existence or a virtual artifact (VA) either through the use of standard input devices such as a keyboard and mouse, or through multimodal devices such as a wired glove or omni-directional treadmills. The simulated environment can be similar to the real world in order to create a lifelike experience—for example, in simulations for pilot or combat training—or it differs significantly from reality, such as in VR games.

37. Virtual Simulation (Unit 1)

Where actual players use simulated systems in a synthetic environment. Examples are for flight simulation, education where avatars are employed or medical simulation using human patient simulators.

38. Work Breakdown Structure (WBS) (Unit 10)
A work breakdown structure (WBS) is depicted on a chart in which the critical work elements, called tasks, of a project are illustrated to portray their relationships to each other and to the project as a whole. The Project Team utilizes the WBS with a project milestone chart. The graphical nature of the WBS can help a project manager predict outcomes based on various scenarios, which can ensure that optimum decisions are made about whether or not to adopt suggested procedures or changes.

Section 2—Coding, Programming & Gaming

Evolution of software development.

1. Moore’s Law* Helps Explain the Advancement of Technology Which Benefits
   - Processing speed
   - Disk capacity
   - Memory capacity

2. Working with the Hardware
   - Initially programming the hardware required intimate knowledge of the target H/W
   - From machine code/assembly language
   - Towards independence from target H/W
   - And to the mature complier tools we use today

* Moore’s Law: The number of components in an integrated circuit doubles approximately every two years.

3. Evolution of Languages
   - Languages used to develop DoD applications numbered over 400
   - Different languages for different problems
   - Focus now is more of extensions to the more recent languages; C++

4. Language Standardization/Extensions
   - Wrappers/Plug-ins
   - Math Libraries/Special Functions/Reuse
   - Reusable standards/components
• Maintenance on Legacy Systems Still Biggest Cost; Fortran, Pascal, Jovial, Ada…

5. Process Models
• Originally ads-hoc development
  • Adoption of disciplined approach introduce less defects, meaning less rework
  • SEI’s CMMI establishes levels of maturity
  • ISO 9001 establishes quality standards for software products

6. Variation of Approaches have Evolved
• Waterfall*
• Evolutionary (incremental, spiral, agile…)**

7. Moving Towards Adopting Commercial Practices
• Tailor processes to developer’s best practices
• Leverage on evolving and maturing development standards and methods
• Application under development determines the approach

8. Development Methods have Evolved
• Structured Analysis / Structured Design
• Object Oriented
• Agile

* Waterfall software development model is a sequential design process, used in software development processes, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of conception, initiation, analysis, design, construction, testing, production/implementation and maintenance.

The waterfall development model originates in the manufacturing and construction industries: highly structured physical environments in which after-the-fact changes are prohibitively costly, if not impossible. Since no formal software development methodologies existed at the time, this hardware-oriented model was simply adapted for software development.

In 1985, the United States Department of Defense captured this approach in DOD-STD-2167A, their standards for working with software development contractors, which stated that "the contractor shall implement a software development cycle that includes the following six phases: Preliminary Design, Detailed Design, Coding and Unit
Testing, Integration, and Testing”.

Some organizations, such as the United States Department of Defense, now have a stated preference against waterfall type methodologies, starting with MIL-STD-498, which encourages evolutionary acquisition and Iterative and Incremental Development.

**Agile software development model** is a group of software development methods in which solutions evolve through collaboration between self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development, early delivery, continuous improvement, and encourages rapid and flexible response to change. The Agile Manifesto is based on 12 principles:

1. Customer satisfaction by early and continuous delivery of useful software
2. Welcome changing requirements, even late in development
3. Working software is delivered frequently (weeks rather than months)
4. Close, daily cooperation between business people and developers
5. Projects are built around motivated individuals, who should be trusted
6. Face-to-face conversation is the best form of communication (co-location)
7. Working software is the principal measure of progress
8. Sustainable development, able to maintain a constant pace
9. Continuous attention to technical excellence and good design
10. Simplicity—the art of maximizing the amount of work not done—is essential
11. Self-organizing teams
12. Regular adaptation to changing circumstance

**Gaming**

**Video games**

Video games are computer- or microprocessor-controlled games. Computers can create virtual spaces for a wide variety of game types. Some video games simulate conventional game objects like cards or dice, while others can simulate environs either grounded in reality or fantastical in design, each with its own set of rules or goals.

A computer or video game uses one or more input devices, typically a button/joystick combination (on arcade games); a keyboard, mouse or trackball (computer games); or a controller or a motion sensitive tool. (console games). More esoteric devices such as paddle controllers have also been used for input.

There are many genres of video game; the first commercial video game, *Pong*, was a simple simulation of table tennis. As processing power increased, new genres such as adventure and action games were developed that involved a player guiding a character
from a third person perspective through a series of obstacles. This "real-time" element cannot be easily reproduced by a board game, which is generally limited to "turn-based" strategy; this advantage allows video games to simulate situations such as combat more realistically. Additionally, the playing of a video game does not require the same physical skill, strength or danger as a real-world representation of the game, and can provide either very realistic, exaggerated or impossible physics, allowing for elements of a fantastical nature, games involving physical violence, or simulations of sports. Lastly, a computer can, with varying degrees of success, simulate one or more human opponents in traditional table games such as chess, leading to simulations of such games that can be played by a single player.

Father of Video Games
Ralph Henry Baer (born Rudolf Heinrich Baer; March 8, 1922 – December 6, 2014) was a German-born American video game developer, inventor, and engineer, and was known as "The Father of Video Games" due to his many contributions to games and the video game industry in the latter half of the 20th century.

Game Engine
A software framework designed for the creation and development of video games. Developers use them to create games for consoles, mobile devices and personal computers. The core functionality typically provided by a game engine includes a rendering engine ("renderer") for 2D or 3D graphics, a physics engine or collision detection (and collision response), sound, scripting, animation, artificial intelligence, networking, streaming, memory management, threading, localization support, and a scene graph. The process of game development is often economized, in large part, by reusing/adapting the same game engine to create different games,[1] or to make it easier to port games to multiple platforms.

A list of Gaming Engines is provided as Attachment C.

1. Implement multimedia programming as it relates to modeling and simulation using a gaming engine
2. Monte Carlo simulation related to game design
3. Game development life cycle
4. Tools and software commonly used in game development (See Attachment C)
Formerly, the output data from a computer simulation was sometimes presented in a table, or a matrix, showing how data were affected by changes in the simulation parameters. The use of simulation models were simply 'black boxes' - data going in and results coming out; however, psychologists noted that humans could quickly perceive trends by looking at graphs or even moving-images generated from the data, as displayed by computer-generated-imagery (CGI) animation. Using on-screen animations in a simulation model enables the status of the model to be viewed as it progresses. For example, a machine that breaks down may change its color to red. This enables visual cues to be passed back to the operator of the simulation model, so action could be taken. Additionally, visualization is useful in convincing decision makers of the model's credibility. For example, in manufacturing if the directors can see a visualization of the production line with widgets traveling down a conveyor belt, it would do more to sell the concept of the model than a 'black box', churning out data. Today visual simulation models are used in many disciplines. In geographical information systems (GIS) and meteorology, weather forecasting models balance the view of moving rain/snow clouds against maps that use numeric coordinates and timing of events. Similarly, computer simulations of CAT scans can simulate how a tumor might change, during an extended period of medical treatment, presenting the passage of time as a spinning view of the visible human head, as the tumor changes. Other applications of CGI computer simulations are being developed to graphically display large amounts of data, in motion, as changes occur during a simulation run.

Principles of Animation

In the 1930s, Walt Disney wasn’t happy with the state of animation at the company. He set up classes for his employees under the direction of Don Graham; from those classes, 12 principles for animation emerged to help animators create more believable animations.

1. Squash and Stretch
   a. Three-dimensional (3D) objects have mass and volume, and they exhibit a certain amount of rigidity when moving. Real objects deform slightly during movement, revealing how rigid they are.
   b. Organic objects aren’t completely rigid. They’re usually softer and more malleable. When the movement of organic objects is stiff and rigid, it doesn’t feel right. Manmade objects, on the other hand, tend to be stiff and rigid.
   c. The principle of squash and stretch is about showing objects flattening and elongating as they move to reveal their rigidity. The object should appear to retain its mass and volume, but it should also deform to appear more natural. For example, a bouncing ball will squash when it comes in contact with the ground, and then it will stretch after the bounce, elongating in the direction of movement.

2. Anticipation
a. Real-world actions don’t start immediately. An action in an animation occurs in three steps:
   
   i. Set up for the action
   
   ii. The action
   
   iii. Follow through on the action

b. The setup is anticipation, which is typically a movement in the opposite direction (contrary movement) to the action.

c. For example, to throw a ball, your arm winds up and moves back before moving forward to release the ball. This windup is anticipation. It signals the action to come and leads the viewer’s eye to the object that will perform the action.

d. Anticipation is often the most important part of any animation. It’s also the part that typically lasts the longest. The greater the action, the longer and more exaggerated the anticipation should be.

3. Staging

   a. Staging is how you present objects. It provides context for the object and the animation, and it helps tell your story.

   b. For example, a cube displayed at an angle to any of its six sides will show the cube as a 3D object. If you’re looking directly at one side though, it will appear to be a 2D square

   c. There are three main considerations with staging:

      i. Characters and objects should have strong silhouettes to more quickly understand them.

      ii. The environment (the stage) should be interesting, without drawing attention away from the characters, objects, and actions.

      iii. Major actions should be presented one at a time or it could potentially lead to confusion.

4. Straight Ahead Action and Pose-to-Pose

   a. Both are methods for creating an animation. In straight ahead action, you would draw everything in frame one and then move on to do the same in frame two. You would continue to draw every frame from first to last in sequence. This method allows animators to explore and discover as they go.

   b. With pose-to-pose, you draw important keyframes and then create in-between frames to connect them. This is how computer animation works, and it’s the way you work with CSS animations. This method allows more planning up front and lets the computer do much of the work.

5. Follow Through and Overlapping Action

   a. In the section on anticipation, I mentioned that animation occurs over three steps and that the last step was following through.

   b. Follow through is similar to anticipation, except it occurs after the action instead of before it. Things don’t stop suddenly in the real world any more than they start suddenly.

   c. When throwing a ball, your arm continues its motion after releasing the ball. When a ball bounces on the ground, it doesn’t stop. It continues bouncing until eventually coming to rest. Both are examples of follow through.

   d. Overlapping action is similar. An example is your arms swinging as you run. The running is the major action and your swinging arms overlap it. The overlapping action should usually move at different speeds than main action.

6. Slow In and Slow Out

   a. Movement usually starts and ends slower than movement in between. Objects accelerate and decelerate. They ease into and ease out of the action. Your car isn’t
doing 60 mph the instant you step on the gas pedal. It doesn’t go back to 0 mph instantly when you step on the break. It accelerates to 60 and decelerates back to 0.

b. In practice, this means adding more keyframes to the start and end of an animation. It’s not uncommon for anticipation to take up 25 percent of an animation and for follow through to take up a similar amount of time.

7. Arcs
   a. Most movement in the natural world occurs over an arc. The movement isn’t a perfect straight line. You want to have movement follow arcs as opposed to straight lines where possible.
   b. The shallower the arc, the faster the object will appear to move and the more pronounced the arc, the slower the object will appear to move. Keep in mind the arc of motion can be around any axis, which might mean into and out of the screen.
   c. Mechanical movement is the exception to this principle. It’s typically less curved and more linear than organic movement.

8. Secondary Action
   a. Secondary action is similar to overlapping action, and it can be difficult to determine if an action is secondary or overlapping at times. The difference is that overlapping action is the result of the main action, and secondary action is independent of the main action. For example, an alien with antenna might be walking (the main action), while the alien’s antenna is moving around to sense the environment (secondary action). Whether the alien is walking or standing still, the antenna continues to sense the environment. Its movement is independent of the walking. Secondary action should remain minor when compared to the main action or their roles could be reversed. It should support the main action and make it more interesting.

9. Timing
   a. The key to good animation is timing. Slow in and out is a subset of this principle.
   b. The timing you use in an animation should give a sense of the object’s characteristics: how heavy or light is it, for example.
   c. Timing can communicate a lot. A person blinking quickly might be seen as alert, whereas a person blinking slowly might appear tired. Changing nothing more than how quickly the eye blinks can help communicate the emotional state of the character.
   d. More than anything, timing is something you should experiment with when creating transitions and animations. Trial and error and your own judgment will go a long way here.

10. Exaggeration
    a. Oddly enough, to make animations appear more realistic, you have to exaggerate them. If animation tries to mimic reality too closely, it tends to appear stiff and lifeless.
    b. Exaggeration is often used to accent actions and emotions. A ball might squash and stretch more than it should. A cartoon windup before throwing a ball is usually shown as the arm spinning like a propeller in the opposite direction of the throw itself.
    c. You don’t want to exaggerate everything. Use restraint. Exaggerate to add a little more life and excitement but not so much to completely destroy believability.

11. Solid Drawing
    a. The idea behind solid drawing is to take the 3D space into account even while working in a 2D plane. The real world is 3D, and your animations should appear to live in that same world.
b. Shadows, gradients, scale, and perspective can all add depth to an animation creating the illusion of 3D space. Ironically, the trend toward flat design does the opposite. Fortunately, designers are already bringing back some depth, albeit more subtle than before. For example, a flat rectangle doesn’t say “button.” Add a slight shadow or gradient, and it looks like something you push or click or tap.

12. Appeal
   a. *Appeal* is like the charisma of a real person. It’s the sum of everything that makes a character in an animation come to life.
   b. Appeal is about trying to make characters and objects more interesting. It’s about giving your animation a certain “je ne sais quoi,” something enjoyable that’s difficult to describe.
   c. This principle is about the whole being more than the sum of the parts. If you consider all the other principles, you should at least find yourself on the right path toward appeal.

Principles of 3D Animation
1. Visual Styling
   a. Visual styling in three-dimensional computer animation means more than just how things are supposed to look.
   b. Visual styling also has a significant impact on rendering, on animation techniques, and overall production complexity.
   c. As we develop a visual look we must keep in mind that it is feasible to produce within the boundaries of the project.
   d. A certain look for the skin of a beast, for example, might look cool but might also require too complex a rig, too detailed a model and too complex an animation process.

2. Blending Cartoon Physics with Real World Physics
3. Cinematography
4. Mastering Facial Animation
5. Optimizing user-controlled Animation

Section 4—Ethics, Entrepreneurship & Innovation

Ethical integrity and character development are grounded in high expectations and
fostering relationships, these must embody all that we as teachers, leaders, and role models do every day. As a professional with access to company and government confidential information, you must understand and respond appropriately to the ethical, legal and limitation issues they might be confronted with as students and later as career professionals.

Ethical standards are what it means to be a good person, the social rules that govern our behavior. Ethics in business is essentially the study of what constitutes the right and wrong behavior in the workplace environment. A business is an organization whose objective is to provide goods or services for profit. The organization has a group of people that work together to achieve a common purpose. The moral challenges that these men and women face each day along with a whole range of problems that could occur, are why ethics plays such an important role in business. Most large businesses have a written code of ethics, sometimes called a code of conduct to set the standards that employees are to follow. In summary a good M&S professional should refrain from engaging in or supporting any activity that would discredit the profession.

(See APPENDIX A for Ethics Power Point)

Commercialization and Innovation

In order to take an invention or new technology to market, students must learn the commercialization processes. To achieve these, students must learn both the legal and technical steps to finance, produce and market a technology solution they have created. This unit will explore these techniques and requirement using the “labs” included in this Unit.

Teaching students the methods necessary to take a product/invention to the marketplace is fundamental to establishing a business and capitalizing on an invention. Many universities have incubators that can facilitate the transition of moving an invention to the marketplace. These organizations also provide advice in this process. The protection of the intellectual property is absolutely necessary throughout the transition process. Without patents, copyrights, trademarks, etc. anyone can take the product/invention as their own and begin to market it. We need to indoctrinate and help students to define the solution aspects of their invention and communicate those features of their To help students understand that entrepreneurship is an alternative career track. To secure loans or gain support for your enterprise, you must have a BUSINESS PLAN. This Curriculum Unit will also give the students all the tools necessary to take a product from Lab to Market. product to potential investors—with proper legal advice.

1. Understand the process of patent application filing, product trials, and communication techniques to describe their product.

2. Explore and examine the Intellectual Property Methods such as patents, copyrights, trademarks, and trade secrets
Section 5—Simulations & Simulators

What is “simulation”?  

**Simulation** is the imitation of the operation of a real-world process or system over time. The act of simulating something first requires that a model be developed; this model represents the key characteristics or behaviors/functions of the selected physical or abstract system or process. The model represents the system itself, whereas the simulation represents the operation of the system over time.

Simulation is used in many contexts, such as simulation of technology for performance optimization, safety engineering, testing, training, education, and video games. Often, computer experiments are used to study simulation models. Simulation is also used with scientific modeling of natural systems or human systems to gain insight into their functioning. Simulation can be used to show the eventual real effects of alternative conditions and courses of action. Simulation is also used when the real system cannot be engaged, because it may not be accessible, or it may be dangerous or unacceptable to engage, or it is being designed but not yet built, or it may simply not exist.

Key issues in simulation include acquisition of valid source information about the relevant selection of key characteristics and behaviors, the use of simplifying approximations and assumptions within the simulation, and fidelity and validity of the simulation outcomes.

Real Time Virtual Simulators—Real time virtual simulators are perhaps the most familiar to students. They have observed or experienced flight simulators used by the military, commercial airlines, nuclear power generation, space, entertainment, law enforcement, accident investigation, medical and other applications. Simulators are being used more frequently today for medical training by the military and for civilian applications. Simulators save time, money and lives.

Real time simulators were first used by the military with Edwin Link, simulator pioneer, introducing his “Blue Box” to the Army Air Corps to train pilots in flying instruments. The commercial airlines recognized the effectiveness of simulators and began to use them for aircrew training. With the introduction of the digital computer, it became possible to design simulators in computer software in lieu of expensive, single-purpose hardware. This versatility, led others to develop simulators for a wide range of applications. The entertainment industry began to introduce video games and location-based entertainment concerns such as theme parks took advantage of simulation technologies such as motion systems, visual displays, computer-generated imagery and even olfactory simulation to add realism to their rides. Homeland Security and law enforcement agencies seized the opportunity to use simulators in marksmanship and “shoot-don’t shoot” trainers. Of late, the medical and health care community adopted simulation technology for training of nurses, EMT’s, doctors, etc. The first such training device was the Human Patient Simulator: it breathes, has a heart beat, blood pressure, etc. Of course, the space program demanded and received high fidelity training.
simulators for individual and crew training. Putting a person on the moon would not have been possible without the use of simulators. It appears that the range of simulator applications is only limited by the imagination of the human.

There have been many studies that validate that time in a simulator can substitute for training time in an aircraft, for example. There is a very strong case for the transfer of training from a simulator to the parent system. This transfer can be quantified as the Training Effectiveness Ratio further described in this study guide. The issue of the fidelity (or amount of realism) of simulation as it affects the transfer of training must be addressed. Levels of fidelity must be considered for part task trainers, full mission simulators and simulators networked together for team mission training.

**Miller Curves:** In the 1950’s, Dr. Robert Miller of IBM postulated that the transfer of training (from simulator to the real system) increased non-linearly with an increase of fidelity (realism), and accordingly, the cost of the system increased non-linearly with that increase of fidelity. Further, he stated that there was an “optimum” level of fidelity where any further increase in fidelity did not yield a great amount of transfer of training.

**Simulator Building Blocks** - Generally, there is a set of components that are to be included in the overall design of a simulator regardless of application. Those building blocks are as follows:

1) Trainee Station (cockpit area in a flight simulator)
2) Computational System (computer or family of computers)
3) Visual System (image generator, displays and data bases)
4) Instructor/Operator Station (Controls scenarios, inserts malfunctions, etc.)
5) Aural System (communications, engine noises, alarms, sensors, environmental noises, etc.)
6) Motion System (platform, g-seat/g-suit, motion seat, seat shaker)
7) Control Loader (stick rudder forces in flight simulator)
8) Interface Hardware and Software (enables computers and other hardware/software to communicate)

Virtual simulation input hardware

24
There is a wide variety of input hardware available to accept user input for virtual simulations. The following list briefly describes several of them:

**Body tracking**

The *motion capture* method is often used to record the user’s movements and translate the captured data into inputs for the virtual simulation. For example, if a user physically turns their head, the motion would be captured by the simulation hardware in some way and translated to a corresponding shift in view within the simulation.

- Capture suits and/or gloves may be used to capture movements of users body parts. The systems may have sensors incorporated inside them to sense movements of different body parts (e.g., fingers). Alternatively, these systems may have exterior tracking devices or marks that can be detected by external ultrasound, optical receivers or electromagnetic sensors. Internal inertial sensors are also available on some systems. The units may transmit data either wirelessly or through cables.
- Eye trackers can also be used to detect eye movements so that the system can determine precisely where a user is looking at any given instant.

**Physical controllers**

Physical controllers provide input to the simulation only through direct manipulation by the user. In virtual simulations, tactile feedback from physical controllers is highly desirable in a number of simulation environments.

- Omni directional treadmills can be used to capture the users locomotion as they walk or run.
- High fidelity instrumentation such as instrument panels in virtual aircraft cockpits provides users with actual controls to raise the level of immersion. For example, pilots can use the actual global positioning system controls from the real device in a simulated cockpit to help them practice procedures with the actual device in the context of the integrated cockpit system.

**Voice/sound recognition**

This form of interaction may be used either to interact with agents within the simulation (e.g., virtual people) or to manipulate objects in the simulation (e.g., information). Voice interaction presumably increases the level of immersion for the user.

- Users may use headsets with boom microphones, lapel microphones or the room may be equipped with strategically located microphones.

**Current research into user input systems**

Research in future input systems hold a great deal of promise for virtual simulations. Systems such as brain-computer interfaces (BCIs) offer the ability to further increase the level of immersion for virtual simulation users. Lee, Keinrath, Scherer, Bischof, Pfurtscheller proved that naïve subjects could be trained to use a BCI to navigate a virtual
apartment with relative ease. Using the BCI, the authors found that subjects were able to freely navigate the virtual environment with relatively minimal effort. It is possible that these types of systems will become standard input modalities in future virtual simulation systems.

**Virtual simulation output hardware**

There is a wide variety of output hardware available to deliver stimulus to users in virtual simulations. The following list briefly describes several of them:

**Visual display** Visual displays provide the visual stimulus to the user.

- Stationary displays can vary from a conventional desktop display to 360-degree wrap around screens to stereo three-dimensional screens. Conventional desktop displays can vary in size from 15 to 60+ inches. Wrap around screens are typically utilized in what is known as a Cave Automatic Virtual Environment (CAVE) **Cave Automatic Virtual Environment**. Stereo three-dimensional screens produce three-dimensional images either with or without special glasses—depending on the design.

- Head mounted displays (HMDs) have small displays that are mounted on headgear worn by the user. These systems are connected directly into the virtual simulation to provide the user with a more immersive experience. Weight, update rates and field of view are some of the key variables that differentiate HMDs. Naturally, heavier HMDs are undesirable as they cause fatigue over time. If the update rate is too slow, the system is unable to update the displays fast enough to correspond with a quick head turn by the user. Slower update rates tend to cause simulation sickness and disrupt the sense of immersion. Field of view or the angular extent of the world that is seen at a given moment **Field of view** can vary from system to system and has been found to affect the users sense of immersion.

**Aural display** Several different types of audio systems exist to help the user hear and localize sounds spatially. Special software can be used to produce 3D audio effects **3D audio** to create the illusion that sound sources are placed within a defined three-dimensional space around the user.

- Stationary conventional speaker systems may be used provide dual or multi-channel surround sound. However, external speakers are not as effective as headphones in producing 3D audio effects.

- Conventional headphones offer a portable alternative to stationary speakers. They also have the added advantages of masking real world noise and facilitate more effective 3D audio sound effects.

**Haptic display** These displays provide sense of touch to the user **Haptic technology**. This type of output is sometimes referred to as force feedback.

- Tactile tile displays use different types of actuators such as inflatable bladders, vibrators, low frequency sub-woofers, pin actuators and/or thermo-actuators to produce sensations for the user.
End effector displays can respond to users inputs with resistance and force. These systems are often used in medical applications for remote surgeries that employ robotic instruments.[17]

**Vestibular display** These displays provide a sense of motion to the user Motion simulator. They often manifest as motion bases for virtual vehicle simulation such as driving simulators or flight simulators. Motion bases are fixed in place but use actuators to move the simulator in ways that can produce the sensations pitching, yawing or rolling. The simulators can also move in such a way as to produce a sense of acceleration on all axes (e.g., the motion base can produce the sensation of falling).
Discrete Event Simulation

Discrete event simulation is an important system analysis technique. A discrete event simulation (DES) manages events in time. Most computer, logic-test, and fault-tree simulations are of this type. In this type of simulation, the simulator maintains a queue of events sorted by the simulated time they should occur. The simulator reads the queue and triggers new events as each event is processed. In this unit you will be able to access data produced by a simulation, understand the event relationships and to discover logic defects in the design, or the sequence of events.
Systems Engineering

Systems engineering is an interdisciplinary approach to manage the technical realization of a successful program. Systems engineering processes provide technical insight into the holistic status of the program. This is important for a number of reasons.

First, there are many technical disciplines, which will address a portion of the product solution, and interact with other design areas. If the design requirements aren’t fully understood by all designers, there can be problems resulting. For example, if the visual system engineers do not fully understand the requirement for weather, a student pilot in the training system may see with their “out-the-window” view a clear and calm day, but the aircraft is not responding because the environmental engineers are simulating thunderstorms and wind shear effects based by a scenario setting not used by the visual engineers.

Second, there may be a trade-off required among the technical designers in order to meet computational size, timing or network limitations.

Third, there may be cost considerations that drive the solution to tailor the technical approaches.

Fourth, there may be schedule considerations with drive the solution a particular way.

Fifth, the technical solution may require more personnel to run or maintain the system than are available. The cost to maintain the system may be higher than the original cost of the system.

Sixth, the architecture (software/hardware) may not be conducive to making changes required to reflect future customer needs.

Or, if it’s a typical program, all of the above are impacting the technical approach. So, how do you manage all this? The systems engineering process provides tools to allow the team to provide the right level of data and insight for the program team to make the right decision. A few fundamentals of what’s expected in systems engineering:

1. Know the requirements (what’s needed?)
2. These requirements are then further derived to better determine the technical design approach (Trace these in a Requirements Traceability Matrix (RTM))
3. The RTM allows changes in design to flow back to the base requirements (training objective and training requirements for training systems)
4. The RTM also flows to design documentation throughout the program, resulting in the basis for your test procedure.
5. Capture Program Risks and document with impact and mitigation strategies.
Document and Brief this to a Risk Team on a planned schedule. (no surprises!)

6. Technical design reviews – these are held based on the progression of the design development to ensure to design is mature enough to go onto the next development stage.

7. Continually evaluate the teams’ status, personnel requirements, schedule considerations, interim product item completion.

8. The amount of documentation and the extent of the review are based on the complexity and risk of the program.
**Integrated Logistics Support (ILS)**

**ILS** is defined as a composite of the elements of design and support necessary to provide for the effective support of the system for its programmed life cycle. The elements include the following:
• Reliability and Maintainability
• Design Interface
• Maintenance Planning
• Supply Support
• Support and Test Equipment
• Manpower and Personnel
• Training and Training Support
• Technical Data and Publications
• Computer Resources Support
• Facilities
• Packaging, Handling and Transportation

Through the ILS process we can maximize simulator Availability and deliver a system that will have minimum Life Cycle Costs.

In the following figure, major program milestone activities are shown on the left and “logistics” actions and interfaces on the right side of the figure.
The logistics program is managed by the Integrated Logistics Support Manager (ILSM) who reports directly to the Project Manager. The ILSM is at the same level as the Project Engineer for the project, having an equal voice when it is necessary to make trade-off decisions. All of the team members constitute an Integrated Project Team (IPT).

Reliability, Availability, and Maintainability (RAM) are system design attributes that have significant impact on the sustainment or total Life Cycle Costs (LCC) of a developed system. Additionally, the RAM attributes impact the ability to perform the intended mission and affect overall mission success. The standard definition of Reliability is the probability of zero failures over a defined time interval (or mission), whereas Availability is defined as the percentage of time a system is considered ready to use when tasked. Maintainability is a measure of the ease and rapidity with which a system or equipment can be restored to operational status following a failure.

The reliability (MTBF) of a training simulator can be predicted using established methods. (Reliability Handbook, prior experience, failure rate tables, etc.). Given the selection of design hardware and software, experience and test tables can provide a range of expected values for Mean Time Between Failures (MTBF) and Mean Time To Repair (MTTR). Because most electronic equipment operates in the constant failure rate portion of the “bathtub” curve, failure rates of subsystems can be predicted individually and combined (added) to yield an overall value of MTBF. (See figure below). However, subsystem MTBF’s CANNOT be added together to give an overall value. Each subsystem’s MTBF must be converted to a “failure rate”, Lambda, which is the reciprocal of MTBF. In other words, Failure Rate (Lambda) = 1/MTBF.

There is no similar calculation using a reciprocal of MTTR. MTTR is calculated differently, it is a function of the Failure Rate. We will cover that shortly.

As an example of calculating reliability (MTBF), let us assume we are going to build a training simulator having three subsystems (A, B, and C). Using predictive tables, the MTBF for each subsystem is given as follows:

<table>
<thead>
<tr>
<th>Subsystem</th>
<th>MTBF</th>
<th>Lambda</th>
<th>1/MTBF</th>
<th>MTTR</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>500 hours</td>
<td>.002 Failures/ Hr.</td>
<td>20 Minutes</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>300</td>
<td>.00333</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>200</td>
<td>.0050</td>
<td>25</td>
<td></td>
</tr>
</tbody>
</table>

33
Given the above, the overall Failure Rate (Lambda) of the total system is the sum of the subsystem failure rates or .01033 Failures/ Hr. Now, to find the overall MTBF, it is necessary to find the reciprocal, 1/Lambda. Therefore, the overall MTBF = 96.8 Hours.

Calculating the overall MTTR is more complicated. It requires the multiplication of each predicted maintenance time (in hours) multiplied by each subsystem failure rate. Those numbers are summed. Then that summed number is divided by the overall failure rate and the result is the system MTTR in Hours. In the case above, the overall system MTTR is .427 Hours or 25.6 Minutes.

The system predicted system Availability then can be calculated by the relationship, A = MTBF/(MTBF + MTTR). This becomes 96.8/ (96.8 + .427) = .996.

The figure below is a block diagram of a virtual simulation for a specific aircraft simulator. In doing reliability and maintainability predictions, values for MTBF and MTTR are estimated for each subsystem block (as shown for the visual and image generator subsystems). Then they are combined in the manner described above to determine the predicted reliability and maintainability values.
Simulator Block Diagram

Motion System

Trainee Station

MTBF = 80,000 hrs, FR = 0.0000125
MTTR = 15 Min

Visual Display System

MTBF = 4,000 hrs
MTTR = 20 min

Visual Image Generator

Instructor Station (Cockpit)

Computer System

ASIP/GPS Electronics

Input/Output System

Tactical Env Network

Motion Electronics

Control Loading Computer System

Digital Comm System

TCAS

Aural Cuing System

Visual Image Generator

Host Computer System
Calculating the Training Effectiveness Ratio (TER) of a simulator:

The Training Effectiveness Ratio (TER) of a simulator is an indicator of how substituting time in a simulator transfers to student performance in the operational system (aircraft). It is measured by comparing training time in the operational system (aircraft) with the combination of simulator plus aircraft times.

This relationship is defined as follows:

\[ \text{TER} = \frac{Y_0 - Y_x}{X} \]

Where in situation A:  
\[ Y_0 = \text{Time in only the operational system (aircraft) to achieve a level of criterion performance} \]

In comparison to situation B, achieving the same level of criterion performance in the operational system (aircraft) by training in the simulator and the operational system (aircraft)

\[ Y_x = \text{Time in the operational system (aircraft) when coupled with the simulator} \]
\[ X = \text{Time in the simulator when coupled with operational system (aircraft)} \]

As an example:

A pilot achieves a certain level of proficiency after 100 hours in the aircraft.

The pilot achieves the same level of proficiency after 80 hours in the aircraft AND 40 hours in the simulator.

Training Effectiveness Ratio (TER) = \( \frac{100-80}{40} = .50 \)

Related to the TER is cost effectiveness. Generally speaking, time in a simulator is approximately 10%-15% of the cost of operating an aircraft. In addition, simulator training is safe and is much more training effective. For example, training scenarios can be practiced repetitively without setting up similar situations using aircraft or other costly assets. Instruction and feedback can be provided at will as opposed to waiting for a complete training exercise to be completed.

Section 6—Project Management & Logistic Support

Project Managers are responsible for the cost, schedule, and performance of the deliverable product and services required under the terms of the contract. The contract
normally includes a specification or statement of work that describes what the customer intends to be delivered. To estimate how long the project will take, who will do the work and what resources are needed to complete the job, the Project Manager has to break down the project into its component parts and do the estimating. This is called a Work Breakdown Structure (WBS). This analysis, given certain project management software tools, enables the manager to develop reasonable estimates for doing the work to the customer’s satisfaction.

Every project has some risk in design, development or delivery and support. Risks can be in performance, schedule and/or cost. The Project Manager must assess that risk in order to develop reasonable cost and schedule estimates. Very often, customers wish to know where those risks are so that they can be involved in risk mitigation (working around those risks). The Project Manager, with his or her team, must identify those risks and determine the impacts anticipated. From that activity, risk mitigation measures are identified to help lessen the effects of the risk areas identified.

Years ago, a company had the advertising slogan, “The quality goes in before the name goes on.” Quality assurance is vital and can be managed under the umbrella of a Quality Plan. Companies have quality standards that everyone in the organization must follow. This is important because the customer is expecting a quality product and by using acceptable processes, the probability of delivering a quality product is much greater. In addition, a close interface must be established between the Project Manager and the contracts or procurement department.

The free flow of information within a project is vital to performance. Project teams can include members representing different disciplines and it is necessary for those team members to communicate. Today, most communications take place electronically. However, face-to-face meetings are very important. Integrated Product Teams (IPTs) are used throughout the government and processes are available on the internet. Also, team members can participate in collaborative design through electronic means. The objective is to have information shared among all team members, including Human Resources.

Whether they realize it or not, students are immersed in the discipline of project management each time they plan a school play, plan their graduation, or enter a robotics competition. Exposing high school students to the formal Project Management body of knowledge will better prepare them for their future endeavors, whether that is in College or the workforce. Project Management is a skill that is in great demand across a wide spectrum of industries and organizations.

The management of projects requires a skill set that can be applied to projects of various sizes, commodities and services. The processes for managing projects include new and unique terms requiring definition and understanding. To be able to function in any aspect of business, government or academia one has to be familiar with project management. From a career perspective, students entering industry will have a definite advantage by understanding the role and functions of project managers and project teams. Very frequently, teams are interdisciplinary, such as Integrated Product Teams (IPTs) requiring an understanding of how teams function and how the needs of the team members are me
Review Questions

1. Explain Virtual Reality, Augmented Reality, and Mixed Reality?
2. How are each of the above simulations used?
3. What is game emersion?
4. What are the game genres?
5. What are the three primary model types?
6. Questions concerning the Object Oriented Programming and Design
7. Questions concerning basic programming: looping constructs and events.
8. What are the three process models and explain the purpose of each.
9. What are Game Theory and its purpose?
10. What is client/server programming and its purpose in simulations?
11. What is the game engine loop (event loop)?
12. What are the different models used in implementing a game engine?
13. What are the different types of game loop architectures?
14. What is the basic architecture of an event loop?
15. What subsystems are implemented within a game engine?
16. What are the types of update mechanisms utilized in a game engine?
17. What is the primary purpose of a game engine?
18. How is the content of a game engine developed and applied to the game layout?
19. What is the different between a modeling engine and a game engine?
20. What is a game engine?
21. What processes are involved in preparing models for use within a game engine?
22. What is the software development life cycle?
23. What are events and event handlers, and how do they function?
24. What is the history and evolution of gaming systems?
25. How has gaming influenced our day-to-day lives?
26. What is the simulation pipeline?
27. What is the animation pipeline?
28. What is an avatar prepared for animating?
29. What is the standard rigging pose called?
30. What are the three primary modeling techniques?
31. How are walk cycles developed?
32. What is the alpha channel used for in modeling and animation?
33. What are “Splines” and how are they used in modeling?
34. What is image processing?
35. What is the basic architecture of an RGB image?
36. What is a gray scale image?
37. What does RGB stand for?
38. What is the Software Development Life Cycle?
39. What are the phases of the Software Development Life Cycle?
40. What is an image converted from RGB to Grayscale?
41. How does the human eye actually interpret color?
42. What are the primary RGB weights?
43. What is the purpose of a game object?
44. What is a registration point or transform point?
45. What concerns must be considered in marketing software?
46. What are the primaries steps utilized in developing a simple walk cycle?
47. What is the primary purpose of a game engine?
48. How is triggering and control performed within a game engine?
49. What is a primitive object within a modeling engine?
50. What are deformation objects in a modeling engine?
51. What is the simulation pipeline?
52. What is the animation pipeline?
53. Flowchart the logic to draw a vertical slash across an image?
54. What is an event handler?

APPENDIX A- Ethics

The objectives of the Ethics unit include:

1. Apply ethical and legal issues (including copyright) with technology.
2. Develop understanding of professional and ethical responsibilities.
3. Perform ethical behavior.
4. Use accepted Netiquette
5. Describe polite and civil communication.
6. Discuss individual integrity and honesty.
7. Explain the purposes of copyrights, trademarks, and patents.
8. Practice ethical behaviors regarding copyright, citation, and plagiarism.

Design Brief: Dealing with Ethical Issues

Context
- Ethical standards are what it means to be a good person, the social rules that govern our behavior.
- Ethics in business is essentially the study of what constitutes the right and wrong behavior in the workplace environment.
- A business is an organization whose objective is to provide goods or services for profit. The organization has a group of people that work together to achieve a common purpose.
- The moral challenges that these men and women face each day along with a whole range of problems that could occur, are why ethics plays such an important role in business.
- Most large businesses have a written code of ethics, sometimes called a code of conduct to set the standards that employees are to follow.

Challenge
Each student will demonstrate an understanding of dealing correctly with ethical issues in the workplace.

A good M&S professional should refrain from engaging in or supporting any activity that would discredit the profession.
Dealing With Ethical Issues: Definitions

- Intellectual Property
  - Intangible creations of the mind protected by legal rights. Namely:
    - Trade Secrets, Copyrights, Patents, and Trademarks

- Trade Secrets
  - Business/Technical Information (formulas, patterns, programs, methods, etc.)

- Copyrights
  - Original works of authorship (text, arts, recordings, etc.)

- Patents
  - Any new and useful... process, machine, improvements. A Patent will grant the right to exclude others.

- Trademarks
  - Word, name, symbol or device (logo, sound, color, etc.)

Why does protecting IP matter? It Creates a competitive advantage!

Dealing With Ethical Issues: Workplace Integrity Situations

1. Your boss is working up a bank deposit. He drops a $20 bill on the floor. He goes off to the bank. Do you keep the $20? NO... give it back

2. You’re reviewing your payroll records and you find out that you’ve shorted one of your employees $20 in pay. He hasn’t noticed it yet. Do you pay him the money? YES

3. Your company is giving an aptitude test for its employees that are going to be considered for promotion. A former employee offers to sell you the answers to the test for $20. Do you buy them? NO

4. Your boss finds a terrible mistake you’ve made. Do you lie and tell him that the employee that he fired last week was the one who made the mistake? NO

5. You find intellectual property from a competitor on the printer or someone mistakenly sends it to you. Do you keep it and use it to your advantage? NO... you destroy it and let them know you found it.

6. A customer leaves his wallet on the counter one night. It has $100 in it. When he comes back the next day do you keep the $100 and tell him you found the wallet without the money in it? NO

7. Your boss asks you to work late and finish a project. He leaves early so you finish quickly and think about claiming a few extra hours to get paid more and so he thinks you put in more effort. Do you claim more hours than worked? NO... on a government contract this is Fraud!

8. You know your friend and fellow employee is stealing from the cash drawer. After a year the owner makes you a manager. Do you fire your friend? YES... you should have addressed this even if you weren’t the manager.

9. After another year the owner offers to make you a partner in the company. Now do you fire your friend? YES... should have dealt with it immediately!

10. Your boss is considering someone for a promotion. You know he is well qualified, but you dislike him strongly. Do you lie to your boss and say he won’t do a good job? NO
APPENDIX B- Real Time Virtual Simulators

The objectives of the Real Time Virtual Simulation unit include:

- Understand the concept and major components of real-time virtual simulators
- Understand the applications of real time simulators
- Become familiar and conduct systems engineering for simulators

### 8.1 RTVS Concept & Components

**Preparing for Careers in Real time Virtual Simulation**

This course introduces the components of virtual simulation and their aspects that specifically relate to real time use with a human operator. Real time simulation expands the usual STEM areas to include human perception, sensitivity, response, behavior and training methods. Simulation and Training curriculum tools can be used to help provide experience with many of the STEM areas.

**Training with Real Time Virtual Simulation:**

- Education and Training methods are often grouped by Knowledge, Skills and Abilities (KSAs). While Knowledge is usually obtained by self study and classroom teaching, Skills and Abilities are effectively learned in simulator training. Skills and Abilities can also be learned in real vehicles and hardware but since this is more expensive and can be dangerous they are commonly used in final qualification training. So simulators are safe and cost effective for introductory training, transition training, recurring skills training and emergency procedures training.
- Flight Simulators got their start with cockpit instrument and emergency procedures skills and abilities training and typically only cost 10% of the cost to train in the aircraft, avoids accidents and is available 24/7 rain or shine.

### System Description of RTVS

**Capabilities**

- Controls feel and operate like the real thing and its virtual environment seams real
- Fast enough to be realistic (able to control like the real thing)
- Accurate enough to be realistic (moves like the real thing)
- Follows earths physics and weather (not the moons)
- Separately generated environment stimuli (visual & motion) are in sync (avoids sickness)
- Provides the vehicle’s cues and environment’s cues the operator expects and needs to operate

**Operability**

- Ease of simulator control (start, stop, freeze, resume, record, replay)
- Variety of activity (scenario selection, malfunctions, emergencies)
- Scoring (feedback on performance)
- Interactivity (instructor role playing, automatic responses, networking with others)

**Availability**

- Affordable to acquire the simulator and the facility it needs
- Affordable to operate and maintain
- Reliable enough to support the use schedule (e.g. 24/7 or 5 days @ 8 hours each or ?)
- Location in a fixed facility or being able to be relocated
Flight Simulator Block Diagram

Real Time Virtual Simulator Types
RTVS Block Diagram

Simulator Components

<table>
<thead>
<tr>
<th>Simulator Type</th>
<th>X-Box</th>
<th>Fire Fighting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulator Component Types</td>
<td>Simulator Components</td>
<td>Simulator Components</td>
</tr>
<tr>
<td>Operator</td>
<td>Gamer (entertainment)</td>
<td>Fireman (student or team)</td>
</tr>
<tr>
<td>Controls</td>
<td>Control Box</td>
<td>Fire hose with spray</td>
</tr>
<tr>
<td>Computers</td>
<td>PC</td>
<td>Host, IOS, Comms</td>
</tr>
<tr>
<td>Environment Generators</td>
<td>TV, Speakers</td>
<td>Burners, sensors, smoke</td>
</tr>
<tr>
<td>Environment Databases</td>
<td>World, Models, Avatars</td>
<td>Fire room, drains, exhaust</td>
</tr>
<tr>
<td>Interfaces</td>
<td>Cables</td>
<td>Cables</td>
</tr>
</tbody>
</table>

Flight Simulator

Simulator Components
Pilot (student, designer, tester, and investigator)
Cockpit controls, switches (I/Os), Headset, Microphone
Host, IOS, Communications
IG, Display, Sound, Motion SAF (Semi-automated Forces) World, Models, Sounds, Avatars, Weather
Cables, I/O, Ethernet, Video, Audio
RTVS Component
Analysis

• Example of a study of expanded requirements for one RTVS component:
  • The visual display
  • Analyze the requirements and constraints
  • What alternative solutions are there
  • What technology types are there
  • What are their costs and benefits

8.3 Systems Engineering

Systems engineering is an interdisciplinary approach to manage the technical realization of a successful program.

Systems engineering processes provide technical insight into the holistic status of the program.
So What do you Do?

So, how do you manage all this?
The systems engineering process provides tools to allow the team to provide the right level of data and insight for the program team to make the right decision.
The systems engineering process allows you to break down the complexity to address and manage program risks for Cost, Schedule and Performance (CSP).
The systems engineer looks at the holistic aspect of the system and provides technical recommendations to the Program Manager and Team Members.

What’s so hard?

• Need to integrate many technical disciplines
• Cost considerations that impact desired approach
• Schedule considerations
• Manage Trade-offs among technical disciplines
• Technical solution may be too expensive to run or maintain, may exceed initial system cost
• Architecture (software/hardware) may not support future changes
• Requirements / Resources may change
SE Fundamentals (continued)

- Capture Program Risks - document with impact and mitigation strategies
– Regularly share risks across (no surprises!)

• Technical design reviews—these are held based on the progression of the design (event vs. schedule driven)

• SE process breaks complex problem down into manageable components

• Utilize overarching project schedule (Integrated Milestone Schedule, IMS)

**SE Fundamentals**
*(continued)*

• Project documentation
  – Project documentation and review is critical to project success, but can be difficult to prioritize when a project gets behind schedule.
  – Challenge to manage the entire program vs. the daily fire drills that may have bigger impacts to the program later

SE Continually evaluates the technical teams’ status, product status, known risk areas, integration across system components, integration progress, overarching schedule status, personnel requirements, and interim product item completion, test status, risk and issue management
**APPENDIX C- List of Gaming Engines**

**Engines (Wikipedia)**

Note: The following list is not exhaustive. It mixes *game engines* with *rendering engines* as well as *API bindings* without any distinctions.

<table>
<thead>
<tr>
<th>Name</th>
<th>Primary programming language</th>
<th>Scripting</th>
<th>Cross-platform</th>
<th>SDL</th>
<th>2D/3D oriented</th>
</tr>
</thead>
<tbody>
<tr>
<td>4A Engine</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Adventure Game Interpreter</td>
<td></td>
<td>C style</td>
<td>Yes</td>
<td>No</td>
<td>2D</td>
</tr>
<tr>
<td>Adventure Game Studio</td>
<td>C++</td>
<td>AGSScript</td>
<td>Yes</td>
<td>No</td>
<td>2D</td>
</tr>
<tr>
<td>Alamo</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Aleph One</td>
<td>C++</td>
<td>Lua, Marathon markup language</td>
<td>Yes</td>
<td>Yes</td>
<td>2.5D</td>
</tr>
<tr>
<td>Allegro</td>
<td>C</td>
<td>Ada, C++, C#, D, Lisp, Lua, Mercury, Pascal, Perl, Python, Scheme</td>
<td>Yes</td>
<td>Yes</td>
<td>2D</td>
</tr>
<tr>
<td>Antiryad Gx</td>
<td></td>
<td>C, C++, Gel</td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Anura</td>
<td>C++</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>2D</td>
</tr>
<tr>
<td>Anvil</td>
<td>C++, C#</td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Name</td>
<td>Primary Programming Language</td>
<td>Scripting</td>
<td>Cross-platform</td>
<td>SDL</td>
<td>2D/3D oriented</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------</td>
<td>-----------</td>
<td>----------------</td>
<td>-----</td>
<td>----------------</td>
</tr>
<tr>
<td>Ardor3D</td>
<td>Java</td>
<td></td>
<td>Yes</td>
<td>No</td>
<td>3D</td>
</tr>
<tr>
<td>Aurora toolset</td>
<td>C++</td>
<td>NWScript</td>
<td>No</td>
<td>No</td>
<td>3D</td>
</tr>
<tr>
<td>BigWorld</td>
<td>Python</td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Blend4Web</td>
<td>JavaScript, Python, C, C++</td>
<td>JavaScript</td>
<td>Yes</td>
<td>No</td>
<td>3D</td>
</tr>
<tr>
<td>Blender</td>
<td>C++</td>
<td>Python</td>
<td>Yes</td>
<td>Yes</td>
<td>3D/2D</td>
</tr>
<tr>
<td>Bork3D Game Engine</td>
<td>C++</td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>BRender</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Build engine</td>
<td>C</td>
<td></td>
<td>No</td>
<td>No</td>
<td>2.5D</td>
</tr>
<tr>
<td>C4 Engine</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Cafu Engine</td>
<td>C++</td>
<td>Lua</td>
<td>Yes</td>
<td>No</td>
<td>3D</td>
</tr>
<tr>
<td>Chrome Engine</td>
<td>C++</td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>ClanLib</td>
<td>C++</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>2.5D</td>
</tr>
<tr>
<td>Clausewitz</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Name</td>
<td>Primary programming language</td>
<td>Scripting</td>
<td>Cross-platform</td>
<td>SDL</td>
<td>2D/3D oriented</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------</td>
<td>-------------------------</td>
<td>----------------</td>
<td>-----</td>
<td>----------------</td>
</tr>
<tr>
<td>Cocos2d, Cocos2d-x, Cocos2d-html5</td>
<td>C++, Python, Objective-C, JavaScript</td>
<td>JavaScript, Java, Lua</td>
<td>Yes</td>
<td>No</td>
<td>2D, 2.5D, 3D</td>
</tr>
<tr>
<td>Codea</td>
<td>Lua</td>
<td></td>
<td>No</td>
<td>No</td>
<td>2D</td>
</tr>
<tr>
<td>Coldstone</td>
<td></td>
<td></td>
<td>Yes</td>
<td>No</td>
<td>2D</td>
</tr>
<tr>
<td>Construct</td>
<td>C++</td>
<td>Javascript</td>
<td>Yes</td>
<td>Unknown</td>
<td>2D</td>
</tr>
<tr>
<td>CopperCube</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Core3D</td>
<td>Objective-C</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>3D</td>
</tr>
<tr>
<td>Corona</td>
<td>C++</td>
<td>Lua</td>
<td>Yes</td>
<td>Unknown</td>
<td>2D</td>
</tr>
<tr>
<td>CPAL3D</td>
<td></td>
<td></td>
<td>No</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Creation Engine</td>
<td>C++</td>
<td>Papyrus</td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>CryEngine</td>
<td>C++</td>
<td>Lua</td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Crystal Tools</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Crystal Space</td>
<td>C++</td>
<td>Java, Perl, Python</td>
<td>Yes</td>
<td>Yes</td>
<td>3D</td>
</tr>
<tr>
<td>Name</td>
<td>Primary programming language</td>
<td>Scripting</td>
<td>Cross-platform</td>
<td>SDL</td>
<td>2D/3D oriented</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------</td>
<td>------------------</td>
<td>----------------</td>
<td>------</td>
<td>----------------</td>
</tr>
<tr>
<td>Cube</td>
<td>C++</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>3D</td>
</tr>
<tr>
<td>Cube 2</td>
<td>C++</td>
<td>CubeScript</td>
<td>Yes</td>
<td>Yes</td>
<td>3D</td>
</tr>
<tr>
<td>Daqor Engine</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Delta3D</td>
<td>C++</td>
<td>Python</td>
<td>Yes</td>
<td>No</td>
<td>2.5D</td>
</tr>
<tr>
<td>Dim3</td>
<td>C++</td>
<td>JavaScript</td>
<td>Yes</td>
<td>No</td>
<td>3D</td>
</tr>
<tr>
<td>DimensioneX Multiplayer Engine</td>
<td>Java</td>
<td>Java, VBScript</td>
<td>Yes</td>
<td>No</td>
<td>2.5D</td>
</tr>
<tr>
<td>DX Studio</td>
<td>C++</td>
<td>JavaScript</td>
<td>No</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Dunia Engine</td>
<td>C++</td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>ego</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Electron toolset</td>
<td>C#</td>
<td>NWScript</td>
<td>No</td>
<td>No</td>
<td>3D</td>
</tr>
<tr>
<td>Eflight Engine</td>
<td>Java</td>
<td></td>
<td>Yes</td>
<td>No</td>
<td>3D</td>
</tr>
<tr>
<td>Name</td>
<td>Primary programming language</td>
<td>Scripting</td>
<td>Cross-platform</td>
<td>SDL</td>
<td>2D/3D oriented</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------------------------</td>
<td>-------------------</td>
<td>----------------</td>
<td>-----------</td>
<td>---------------</td>
</tr>
<tr>
<td>Enforce</td>
<td></td>
<td></td>
<td>No</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Enigma Engine</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Env3D</td>
<td>Java</td>
<td></td>
<td>Yes</td>
<td>No</td>
<td>3D</td>
</tr>
<tr>
<td>Esperient Creator</td>
<td>Lang</td>
<td>Lisp, CScript</td>
<td>No</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Essence Engine</td>
<td></td>
<td></td>
<td>No</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Euphoria</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Exult</td>
<td>C++</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>2D</td>
</tr>
<tr>
<td>Flare3D</td>
<td>ActionScript 3</td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Flexible Isometric Free Engine</td>
<td>C++</td>
<td>Python</td>
<td>Yes</td>
<td>Yes</td>
<td>2D</td>
</tr>
<tr>
<td>Flixel</td>
<td>ActionScript</td>
<td></td>
<td>Yes</td>
<td>No</td>
<td>2D</td>
</tr>
<tr>
<td>Forgelight Engine</td>
<td></td>
<td></td>
<td>No</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Name</td>
<td>Primary programming language</td>
<td>Scripting</td>
<td>Cross-platform</td>
<td>SDL</td>
<td>2D/3D oriented</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------</td>
<td>---------------------</td>
<td>----------------</td>
<td>-----</td>
<td>----------------</td>
</tr>
<tr>
<td>Fox Engine</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Freescape</td>
<td></td>
<td>Freescape Command Language</td>
<td>Yes</td>
<td>No</td>
<td>3D</td>
</tr>
<tr>
<td>Frostbite</td>
<td>C++</td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Future Pinball</td>
<td></td>
<td></td>
<td>No</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Gamebryno</td>
<td>C++</td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Game Editor</td>
<td></td>
<td>Custom (C styled)</td>
<td>Yes</td>
<td>Unknown</td>
<td>2D</td>
</tr>
<tr>
<td>GameMaker: Studio</td>
<td>Delphi</td>
<td>Game Maker Language</td>
<td>Yes</td>
<td>Unknown</td>
<td>2D</td>
</tr>
<tr>
<td>GamePlay3D</td>
<td>C++</td>
<td>Lua</td>
<td>Yes</td>
<td>No</td>
<td>3D</td>
</tr>
<tr>
<td>GameSalad</td>
<td>Lang</td>
<td>Script</td>
<td>Yes</td>
<td>Unknown</td>
<td>2D</td>
</tr>
<tr>
<td>Gamestudio</td>
<td></td>
<td>C-Script, Lite-C</td>
<td>No</td>
<td>No</td>
<td>3D</td>
</tr>
<tr>
<td>Gamvas</td>
<td>JavaScript</td>
<td>JavaScript</td>
<td>Yes</td>
<td>No</td>
<td>2D</td>
</tr>
<tr>
<td>Godot</td>
<td>C++</td>
<td>GDScript</td>
<td>Yes</td>
<td>No</td>
<td>2D, 3D</td>
</tr>
<tr>
<td>Name</td>
<td>Primary <strong>programming language</strong></td>
<td><strong>Scripting</strong></td>
<td><strong>Cross-platform</strong></td>
<td><strong>SDL</strong></td>
<td><strong>2D/3D oriented</strong></td>
</tr>
<tr>
<td>---------------</td>
<td>----------------------------------</td>
<td>------------------------</td>
<td>--------------------</td>
<td>---------</td>
<td>-------------------</td>
</tr>
<tr>
<td>GoldSrc</td>
<td>C, C++, Assembly</td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>HeroEngine</td>
<td>C++, C#</td>
<td>HeroScript Language</td>
<td>No</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Horde3D</td>
<td>C++</td>
<td></td>
<td>Yes</td>
<td>No</td>
<td>3D</td>
</tr>
<tr>
<td>HPL Engine</td>
<td>C++</td>
<td>AngelScript</td>
<td>Yes</td>
<td>Yes</td>
<td>3D</td>
</tr>
<tr>
<td><strong>id Tech 1</strong></td>
<td>C</td>
<td>ACS</td>
<td>Yes</td>
<td>No</td>
<td>2.5D</td>
</tr>
<tr>
<td><strong>(Doom)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>id Tech 2</strong></td>
<td>C</td>
<td>QuakeC</td>
<td>Yes</td>
<td>No</td>
<td>3D</td>
</tr>
<tr>
<td><strong>(Quake)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>id Tech 2</strong></td>
<td>C</td>
<td>C</td>
<td>Yes</td>
<td>No</td>
<td>3D</td>
</tr>
<tr>
<td><strong>(Quake II)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>id Tech 3</strong></td>
<td>C</td>
<td>C</td>
<td>Yes</td>
<td>No</td>
<td>3D</td>
</tr>
<tr>
<td><strong>id Tech 4</strong></td>
<td>C++</td>
<td>C++ via <strong>DLLs</strong></td>
<td>Yes</td>
<td>No</td>
<td>3D</td>
</tr>
<tr>
<td>Name</td>
<td>Primary programming language</td>
<td>Scripting</td>
<td>Cross-platform</td>
<td>SDL</td>
<td>2D/3D oriented</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------</td>
<td>-----------</td>
<td>----------------</td>
<td>-----</td>
<td>---------------</td>
</tr>
<tr>
<td>id Tech 5</td>
<td>C++, AMPL, Clipper, Python</td>
<td>Script</td>
<td>Yes</td>
<td>No</td>
<td>3D</td>
</tr>
<tr>
<td>IMUSE</td>
<td></td>
<td></td>
<td>N/A</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>Infinity Engine</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>2D</td>
</tr>
<tr>
<td>INSANE</td>
<td></td>
<td></td>
<td>N/A</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>ioquake3</td>
<td>C</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>3D</td>
</tr>
<tr>
<td>Iron Engine</td>
<td></td>
<td></td>
<td>No</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>IW engine</td>
<td>C++</td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Jade</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Jake2</td>
<td>Java</td>
<td></td>
<td>Yes</td>
<td>No</td>
<td>3D</td>
</tr>
<tr>
<td>Jedi</td>
<td>C</td>
<td></td>
<td>Yes</td>
<td>No</td>
<td>2.5D</td>
</tr>
<tr>
<td>Name</td>
<td>Primary programming language</td>
<td>Scripting</td>
<td>Cross-platform</td>
<td>SDL</td>
<td>2D/3D oriented</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------------------------</td>
<td>-----------</td>
<td>---------------</td>
<td>-----</td>
<td>----------------</td>
</tr>
<tr>
<td>jMonkeyEngine</td>
<td>Java</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>3D</td>
</tr>
<tr>
<td>Jogre</td>
<td>Java</td>
<td></td>
<td>Yes</td>
<td>No</td>
<td>2D</td>
</tr>
<tr>
<td>Kinetica</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>3D</td>
</tr>
<tr>
<td>Kivy</td>
<td>Python</td>
<td>Kv</td>
<td>Yes</td>
<td>No</td>
<td>2.5D</td>
</tr>
<tr>
<td>Leadwerks Engine</td>
<td>BlitzMax</td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>LibGDX</td>
<td>Java</td>
<td></td>
<td>Yes</td>
<td>No</td>
<td>2D, 3D</td>
</tr>
<tr>
<td>LithTech</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Luminous Studio</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>LyN</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>M.U.G.E.N</td>
<td>C</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>2D</td>
</tr>
<tr>
<td>LÖVE</td>
<td>Lua</td>
<td>Lua</td>
<td>Yes</td>
<td>Yes</td>
<td>2D</td>
</tr>
<tr>
<td>Marmalade</td>
<td>C++</td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Name</td>
<td>Primary programming language</td>
<td>Scripting</td>
<td>Cross-platform</td>
<td>SDL</td>
<td>2D/3D oriented</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------------------------</td>
<td>-----------</td>
<td>----------------</td>
<td>-------</td>
<td>----------------</td>
</tr>
<tr>
<td>MikuMikuDance</td>
<td>C++</td>
<td></td>
<td>No</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Mizuchi</td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
<td></td>
</tr>
<tr>
<td>Moai SDK</td>
<td>C++</td>
<td>Lua</td>
<td>Yes</td>
<td>Yes</td>
<td>2D</td>
</tr>
<tr>
<td>MT Framework</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Nebula Engine</td>
<td></td>
<td></td>
<td>No</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>NScripter</td>
<td></td>
<td></td>
<td>No</td>
<td>Unknown</td>
<td>2D</td>
</tr>
<tr>
<td>Odyssey Engine</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>OGRE</td>
<td>C++</td>
<td>C++</td>
<td>Yes</td>
<td>No</td>
<td>3D</td>
</tr>
<tr>
<td>ONScripter</td>
<td>NScripter</td>
<td>NScripter</td>
<td>Yes</td>
<td>Yes</td>
<td>2D</td>
</tr>
<tr>
<td>OpenClonk</td>
<td>C++</td>
<td>C4Script</td>
<td>Yes</td>
<td>No</td>
<td>2.5D</td>
</tr>
<tr>
<td>OpenSimulator</td>
<td>C#</td>
<td>LSL</td>
<td>Yes</td>
<td>No</td>
<td>3D</td>
</tr>
<tr>
<td>ORX</td>
<td>C/C++</td>
<td>Custom</td>
<td>Yes</td>
<td>Yes</td>
<td>2.5D</td>
</tr>
<tr>
<td>Name</td>
<td>Primary programming language</td>
<td>Scripting</td>
<td>Cross-platform</td>
<td>SDL</td>
<td>2D/3D oriented</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------------------</td>
<td>-----------------</td>
<td>----------------</td>
<td>-----</td>
<td>----------------</td>
</tr>
<tr>
<td>Panda3D</td>
<td>C++</td>
<td>Python</td>
<td>Yes</td>
<td>No</td>
<td>3D</td>
</tr>
<tr>
<td>Panta Rhei</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>PhyreEngine</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Pie in the Sky</td>
<td>C</td>
<td></td>
<td>Yes</td>
<td>No</td>
<td>2.5D</td>
</tr>
<tr>
<td>PlayCanvas</td>
<td>JavaScript</td>
<td>JavaScript</td>
<td>Yes</td>
<td>No</td>
<td>3D</td>
</tr>
<tr>
<td>PlayN</td>
<td>Java</td>
<td></td>
<td>Yes</td>
<td>No</td>
<td>2D</td>
</tr>
<tr>
<td>PLIB</td>
<td>C++</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>3D</td>
</tr>
<tr>
<td>Pyrogenesis</td>
<td>C++</td>
<td>JavaScript</td>
<td>Yes</td>
<td>Yes</td>
<td>3D</td>
</tr>
<tr>
<td>Q</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Real Virtuality</td>
<td>C++</td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>REDengine</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Ren'Py</td>
<td>Python</td>
<td>Pygame</td>
<td>Yes</td>
<td>Yes</td>
<td>2D</td>
</tr>
<tr>
<td>Name</td>
<td>Primary programming language</td>
<td>Scripting</td>
<td>Cross-platform</td>
<td>SDL</td>
<td>2D/3D oriented</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------</td>
<td>-------------</td>
<td>----------------</td>
<td>--------</td>
<td>----------------</td>
</tr>
<tr>
<td>RenderWare</td>
<td></td>
<td>RWX</td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Retribution Engine</td>
<td>C++</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>3D</td>
</tr>
<tr>
<td>Revolution3D</td>
<td>C++</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>3D</td>
</tr>
<tr>
<td>Riot Engine</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Rockstar Advanced Game Engine (RAGE)</td>
<td>C++</td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
<td></td>
</tr>
<tr>
<td>RPG Maker</td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>2D</td>
<td></td>
</tr>
<tr>
<td>SAGE</td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
<td></td>
</tr>
<tr>
<td>SCUMM</td>
<td></td>
<td>Yes</td>
<td>No</td>
<td>2D</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Primary programming language</td>
<td>Scripting</td>
<td>Cross-platform</td>
<td>SDL</td>
<td>2D/3D oriented</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------</td>
<td>--------------------</td>
<td>----------------</td>
<td>-----</td>
<td>----------------</td>
</tr>
<tr>
<td>SFML</td>
<td>C++</td>
<td></td>
<td>Yes</td>
<td>No</td>
<td>2D</td>
</tr>
<tr>
<td>Sge2d</td>
<td>C</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>2D</td>
</tr>
<tr>
<td>Shark 3D</td>
<td>C++</td>
<td>Python</td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>ShiVa</td>
<td>C++</td>
<td>Lua</td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Sierra's Creative Interpreter (SCI)</td>
<td></td>
<td></td>
<td>Yes</td>
<td>No</td>
<td>2D</td>
</tr>
<tr>
<td>Silent Storm engine</td>
<td></td>
<td></td>
<td>No</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Sith</td>
<td></td>
<td>COG</td>
<td>No</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Source</td>
<td>C++</td>
<td>Squirrel</td>
<td>Yes</td>
<td>Yes</td>
<td>3D</td>
</tr>
<tr>
<td>Spring</td>
<td>C++</td>
<td>C, C++, Java/JVM, Lua, Python</td>
<td>Yes</td>
<td>Yes</td>
<td>3D</td>
</tr>
<tr>
<td>Starling Framework</td>
<td>ActionScript</td>
<td></td>
<td>Yes</td>
<td>No</td>
<td>2D</td>
</tr>
<tr>
<td>Stencyl</td>
<td>Haxe</td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>2D</td>
</tr>
<tr>
<td>Name</td>
<td>Primary programming language</td>
<td>Scripting</td>
<td>Cross-platform</td>
<td>SDL</td>
<td>2D/3D oriented</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------------------</td>
<td>--------------------</td>
<td>----------------</td>
<td>-----</td>
<td>---------------</td>
</tr>
<tr>
<td>StepMania</td>
<td>C++</td>
<td>Lua</td>
<td>Yes</td>
<td>No</td>
<td>3D</td>
</tr>
<tr>
<td>Stratagus</td>
<td>C</td>
<td>Lua</td>
<td>Yes</td>
<td>Yes</td>
<td>2D</td>
</tr>
<tr>
<td>SunBurn XNA</td>
<td></td>
<td></td>
<td>No</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>TOSHI</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Truevision3D</td>
<td>Visual Basic, C++</td>
<td></td>
<td>No</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Torque3D</td>
<td>C++</td>
<td>TorqueScript</td>
<td>Yes</td>
<td>No</td>
<td>3D</td>
</tr>
<tr>
<td>Turbulenz</td>
<td>TypeScript</td>
<td>JavaScript</td>
<td>Yes</td>
<td>No</td>
<td>2D, 3D</td>
</tr>
<tr>
<td>UbiArt</td>
<td>C, C++, C#</td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>2.5D</td>
</tr>
<tr>
<td>Unigine</td>
<td>C++</td>
<td>UnigineScript</td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Unity</td>
<td>C, C++, C#</td>
<td>Mono, UnityScript, C#, Boo</td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Name</td>
<td>Primary programming language</td>
<td>Scripting</td>
<td>Cross-platform</td>
<td>SDL</td>
<td>2D/3D oriented</td>
</tr>
<tr>
<td>---------------------------</td>
<td>------------------------------</td>
<td>-----------------------</td>
<td>----------------</td>
<td>-----</td>
<td>----------------</td>
</tr>
<tr>
<td>Unreal Engine</td>
<td>C++, C#, Assembly</td>
<td>GLSL, Cg, HLSL, UnrealScript</td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>V-Play Game Engine</td>
<td>C++</td>
<td>Lua</td>
<td>Yes</td>
<td>Unknown</td>
<td>2D</td>
</tr>
<tr>
<td>Vengeance Engine</td>
<td>C++</td>
<td></td>
<td>No</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Vicious Engine</td>
<td>Lang</td>
<td>Script</td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Virtools</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Vision</td>
<td>Lang</td>
<td>Script</td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Visual3D Game Engine</td>
<td>C#/.NET</td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Visual Pinball</td>
<td>C++</td>
<td>VBScript</td>
<td>No</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>VRAGE</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>Wintermute Engine</td>
<td></td>
<td>C-like syntax</td>
<td>No</td>
<td>No</td>
<td>2.5D</td>
</tr>
<tr>
<td>World Builder</td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>2D</td>
</tr>
<tr>
<td>WorldForge</td>
<td>C++</td>
<td>Lua (client), Python (server)</td>
<td>Yes</td>
<td>Yes</td>
<td>3D</td>
</tr>
<tr>
<td>Name</td>
<td>Primary programming language</td>
<td>Scripting</td>
<td>Cross-platform</td>
<td>SDL</td>
<td>2D/3D oriented</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------</td>
<td>--------------</td>
<td>----------------</td>
<td>-------</td>
<td>----------------</td>
</tr>
<tr>
<td>X-Ray Engine</td>
<td></td>
<td></td>
<td>No</td>
<td>Unknown</td>
<td>3D</td>
</tr>
<tr>
<td>XnGine</td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>3D</td>
</tr>
<tr>
<td>ZGameEditor</td>
<td>Delphi</td>
<td>Custom</td>
<td>Yes</td>
<td>No</td>
<td>3D</td>
</tr>
<tr>
<td>Zillions of Games</td>
<td></td>
<td>Zillions Rules</td>
<td>No</td>
<td>Unknown</td>
<td>2D</td>
</tr>
</tbody>
</table>

64