Silver Tear

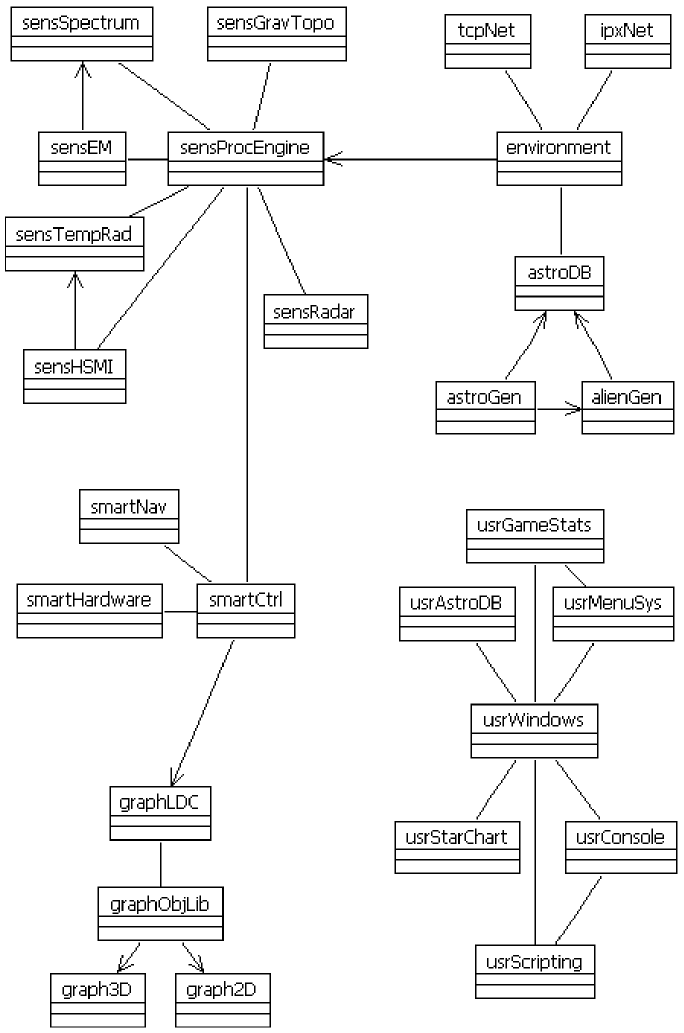
Game Development Quote

Andrew C. Moore

Overview

Unitel has created detailed design specifications for a PC video game that is sure to be a blockbuster. The most important aspect of any video game is how involving it is. There are two classical methods of involving players: The action game method and the role-playing method. The action game method is exemplified by ID Software's Quake, which provides a realistic environment and natural game play. The role-playing game method is exemplified by the popular PC game Diablo II, which provides detailed control over character development, allowing gamers to create highly unique characters. Silver Tear will combine both of these methods seamlessly. Silver Tear will feature real time interactive game play, intricate profile development, and an environment of unprecedented scope and detail.

Throughout this proposal document, you may refer to the following object model at your convenience:



Since Silver Tear is first of all an ultra realistic simulator, the graphics engine will not be the most sophisticated aspect of the game. Silver Tear will be a unique kind of simulator in that some of the simulated data will not be exposed to the user. The environment will be filtered through a special encapsulated software layer that will imitate data input from real sensor equipment. The sensor software will provide the game with a simple application interface that will not change when it is replaced with hardware. This will make it extremely easy to substitute the software with actual hardware. In fact, the entire game will be designed like a real hardware control interface.

The Silver Tear data file will be created with real astrometric data available on the Internet from various government (.gov) and educational (.edu) sites. Since there are not enough details to accurately render planets in other solar systems, not all of the game can be based on empirical data, thus the game's designers will have to supplement the real data with imaginary data. For example, by analyzing properties known about other stars (such as mass, size, and temperature), we can decide if it is even possible for a particular star to have a solar system or, more specifically, provide the proper conditions for one of its planets to support life. Silver Tear will calculate the range of possibilities as accurately as possible and all imaginary data will be constrained to a scientifically calculated range of possibility, thus no impossible or even unlikely imaginary data will be used.

In order to make game play as much fun as possible, Silver Tear will feature space faring extraterrestrial civilizations. A lot of reports of encounters with extraterrestrial beings have been recorded over the last 75 years or so, much of it available on the Internet. The game will use this information to randomly generate various encounters based on supposedly *real* alien species, such as the “Grays” and the “t’Zintli”.

Silver Tear will support SPX/IPX and local and Internet TCP/IP network play. Games will be saved and restored with “profiles”, which are like user accounts. Profiles will record various game statistics, such as “Class-M planets discovered”, “Planetary star systems discovered”, “Light years traveled by tunneling and/or sub light propulsion”, and “Extraterrestrial diplomatic stats”, which would provide a overview of total encounters and details on each, such as which ones became your friends and which ones were hostile or had to be destroyed. Profiles will also save any generated imaginary data. When a network game is created, only one player can host. However, the only job of the host will be to share the generated data from its profile so that everyone in the game reads the same details for each region of the galaxy. Hosts will only be necessary for local network games, since Internet games will use the same generated data all over the world. Network protocols will be primarily peer-to-peer.

As a player explores the universe, recording data on uncharted areas, they will collect “stat points” in their user profile. Once a certain number of stat points have been accumulated, the player will be able to enhance their profile, such as increasing sensor sensativity, implementing new “smart” functions, and even including more spacecraft, which can be controlled simultaneously. A user interface will be designed to control multiple spacecraft with ease and efficiency. Players will have the ability to perform composite sensor analysis, for example, with four ships, one on each side of a planet collecting unique data on the same object. Advanced players can create their own composite sensor analysis and split up tasks according to custom scripts.

The Game Engine

There will be three layers to the graphics engine: The scene creator, the object creator, and the graphics rendering system, which will consist of a 2D and 3D module. The scene creation component (graphLCD; “graphics local data compiler”) will render the scene background and import objects from the object creation component (graphObjLib; “graphics object library”), creating a scene object. The scene object can then be passed to either graphics rendering components to be displayed in 2D, 3D, or combinations of both.

Rendering graphics to the screen is accomplished with three steps: Create a game window, assign a graphics object to the window, and render it. The window object’s graphics object is an abstract class that is implemented in any class that needs to render graphics. By using an abstract class, the window object’s graphics object can be any kind of object that renders graphics, while enabling early binding. (Early binding is a method of increasing performance by defining an object variable so the program knows what kind of object the variable contains before it is used.) Additionally, by allowing objects to define their own graphics routines, the graphics engine does not need to be modified whenever a new graphics function is created.

The graphObjLib component will be optimized for reading and rendering 3D objects. Objects will be stored in a data file in a VRML compatible format. Using VRML will enable game designers to work will a wide variety of 3D software so that a proprietary program will not be required in order to create additional game objects. However, due to possible performance issues, the game may not use the VRML directly. Instead, only the objects included in the local environment during game play will be loaded into a cache in a unique game format. When this occurs, if an object has not been cached before, a conversion template will be stored to disk so the next time it is used, it can be loaded faster. Several other parts of the game will function similarly by loading raw data, processing it, caching it, and then saving processing templates that will allow the raw data to be loaded much faster the second time. By using the game console (a text-based command interface), game developers can tell the program to create all templates at once or specify template data individually.

The Game Environment

Real astronomic data will be used to generate Silver Tear's game environment in order to provide gamers with a universe as realistic as possible. Even though there is an enormous amount of star data available from a number of sources, obviously there will not be enough information to recreate planets in other solar systems or dark matter. These gaps in the real data will be filled in with imaginary data. The software components that manage the environmental data will consist of three components: AstroDB, astroGen, and astroAlienGen. The astroDB component will use the other two components and provide the interface to the rest of the application. By using the real astronomic data component's interface exclusively throughout the rest of the program, all data will be treated as actual data, acceptable to any scientific organization.

The astroGen component will be very different from astroDB, as it will not actually store any data. Instead, generated imaginary data will be stored and retrieved with astroDB. The Environment module's *region* property will be set by astroDB's Data method, which will require the coordinates of the region and the current user profile. The astroDB module will determine if the region is uncharted and retrieve the data from its own GetEnvData method or the astroGen module's Data method, which will either generate new data or retrieve data from the user profile.

The Environment module one of only three modules, which will be removed from the program when it is interfaced to actual hardware (the other two being astroGen and alienGen (the individual sensor modules will be replaced with actual hardware, but the software interface will remain the same)) since the Environment module will provide the sensor modules with seed data necessary to generate the simulated sensor input data.

The Developers

According to the current Silver Tear development plan, Unitel will hire one project manager on a twelve-month contract, three to five programmers and two to three graphic artists on three-month contracts. The project manager will be the author of this proposal document. Dividing the entire one-year project into four stages with four three-month contracts will allow necessary flexibility during the creation of different parts of the game. Some components will require the aid of specialists in a variety of areas of computer programming and science, especially in the design of the sensor systems and the "smart" components.

One or two of the programmers' contracts hired in the first phase will be renewed through the rest of the project. These programmmers will be very important throughout the development of Silver Tear, functioning as lead programmers. They will ensure that each new module created is integrated appropriately into the rest of the program architecture.

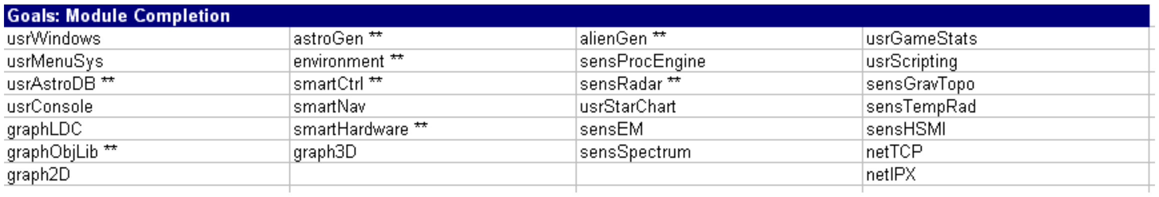
The graphic artist hired in the first phase will also likely have their contracts renewed over the course of the one-year development project in order to maintain style integrity throughout the game. At their discretion, one additional graphic artist may be hired on three-month contracts, for example, in order to help with the research and creation of the alien species.

An astronomy specialist may be required during the construction of the smartNav module to ensure scientifically accurate navigation. However, it will be during the last two phases of development that the scientific specialists will come into play, specifically for the implementation of the sensor modules. These components will function as well as real sensor interpretation programs. The scientist will ensure that every aspect of Silver Tear is completely realistic as a simulator.

The Development Timeline

Each component in the object model will be completed within one of the four stages of development. The first stage will be the most important in that it must lay a firm foundation for the rest of the program. However, it will be easy to adjust these basic functions throughout the rest of the project, the team will be focusing on other important components and will need to utilize other specialist.

The following diagram shows which software components will be completed during each of the four three-month development phases. Components followed by a double asterisk (**) will require continued development in other stages. These components have a larger scope, such as usrAstroDB, which will contain all of the interstellar data used by the game, thus they will require more than three months time to complete.



Goals: Module Completion			
usrWindows	astroGen **	alienGen **	usrGameStats
usrMenuSys	environment **	sensProcEngine	usrScripting
usrAstroDB **	smartCtrl **	sensRadar **	sensGravTopo
usrConsole	smartNav	usrStarChart	sensTempRad
graphLDC	smartHardware **	sensEM	sensHSMI
graphObjLib **	graph3D	sensSpectrum	netTCP
graph2D			netIPX

The first phase of the project will focus on the overall program architecture and the user interface. This is likely to be the most intense of all four stages, since the later stages will require a firm foundation on which to implement the more specialized functions. All of the specialists hired during this phase will be programmers specializing in interface design and object modeling.

The second phase will also focus on programming, although the programmers will need to specialize in artificial intelligence and realtime 3D graphics. During this phase all of the 3D graphics routines will be written and the ship's *smart* components will be created and implemented. It will be important to keep the smart components' interface simple, as the sensor systems will rely on the A.I. to perform composite senor analysis.

The third phase will focus on sensor control and analysis. For this phase, scientific consultants specializing in radar and spectography will work with the programmers to design those sensor modules. The graphic artist will assemble the alienGen graphics and work with the programmers to integrate realistic alien enounters into the game play.

Next to the first, the final phase will be the most intense. The networking protocals will be implemented and the remaining sensor modules will be created. Also, the game statists tracking will be integrated and the console will be enhanced with an extensible module designed to allow scripting.

Conclusion

Every game has a story associated with it, providing a background for the game's setting. The story to Silver Tear will be exactly what the game is ultimately being designed for. Game play is supposed to make a player feel as if they are actually in direct control of an interstellar spacecraft. Gamers will imagine that Unitel has received sufficient funding to build the first "droid cube". This is a structure designed to house seven or eight thousand Unitel droid units, 75cm diameter Unitel spacecraft armed to the teeth with sensors and equipped with a superluminal data port, or “ansible”*[*].

Although the technical depth of Silver Tear will be unrivalled, so will the ease of use. A player will be able to immerse them self in exploring a realistic universe right away and acquaint them self with the technical details at their leisure. The smartCtrl module will handle as little or as much of the complex functions of the game as the player specifies. Integrated help systems will make learning low-level control of the sensors and diagnostic systems as easy as possible.

The original Silver Tear simulator game will be released with the plan in mind that there will be a sequel. Similar to *Diablo II*, *Silver Tear Release 2* will basically a much-improved version of the original. However, unlike Blizzard Software, Unitel Inc. will be planning to make a second game. Blizzard didn’t quite expect the first Diablo to be such a hit, nor were they completely prepared for the volume of suggested improvements. So when they realized they were onto a really great game and they came up with so many improvements they just threw up their hands and announced; “Let there be Diablo II!” Unitel knows that it might as well be impossible to develop a perfect game of any magnitude right off the drawing board. With this in mind, we will release parts of the game that we feel could benefit the most from massive enhancements (such as the astroGen module, the smartCtrl module and the sensProcEngine) as open source. The open source license will require users to post code enhancements to the Unitel web site. The game will also provide integrated online bug reporting and suggestion submissions. Other areas of the game that contain more sensative code that should not be made open source, but nonetheless would benefit from that type of improvement, will employ integrated development tools and scripts, similar to ID Software’s “Quake C”. By planning from the start to create a second release, Unitel will be able to make the second game exponentially enhanced and do so much quicker than comparable video game sequels.

[*] Borrowed from ***Ender’s Game*** by Orson Scott Card