# ersatz – a Software Platform for Making Art

Sheldon BROWN
*CRCA UCSD, 9500 Gilman Drive, La Jolla CA 92093 USA*

**Abstract**. The developments of digital technology provide opportunities for artists to radically rethink the notion of mediums as entities that are containers to be filled with the content of their artwork. Digital technologies provide the capability for artists to create their own functional mediums along with a particular artwork. However, this technological creativity can create daunting development tasks that obscure the types of creative processes that are inherently a part of many artmaking strategies. Over the past decade, the author has developed a number of large-scale, complex, public art projects that utilize such capabilities as high resolution interactive 3D graphics, multi-site networking, multi-user engagement, computer vision and haptic interfaces in novel architectural frameworks. In creating these works, the author has strived to take advantage of the best in available technological approaches, synthesized into a toolset that allows for cultural and aesthetic innovation facilitated by technological innovation. In doing so, a continuous balance between disciplined software development methodologies and the emergent processes of aesthetic discovery is negotiated. The *ersatz* framework of tools is a continually extensible approach that balances the opportunities created through discovery based processes and the need to create robust software frameworks to maximize resource capabilities.

## 1. Introduction

In our current cultural condition, an artist has more approaches to creating art then ever before. This is due to at least two situations: 1) the continued expansion of activity taking place under the rubric of artmaking that is the legacy of post-Duchampian art practice; and 2) the inventive possibilities for developing new forms of expression that digital technologies provide. Each of these attributes on their own provide an expansive horizon for creative inquiry, when combined they present a situation to artists that warrants a considered approach to how one can most effectively participate in working with our evolving techno-cultural dynamic.

Over the past 12 years, I have developed several large-scale computer mediated public art environments that utilize a variety of leading edge technological and aesthetic methodologies. Through this process I have strived to develop a set of approaches that balance the dynamic need to build robust technology, with the need to work in an environment that encourages, rather then inhibits, aesthetic and conceptual discovery. In some ways, these requirements run counter to each other and one can readily find approaches from other technologically dependent, creative industries which create firewalls (in the older sense of the term) between content creation and technology development[1]. In my studio practice, I try and create tools that facilitate my artmaking approaches through extensible technology frameworks, constructed via sound software architecture approaches. The technological underpinnings coming from various areas of recent computer science research (computational geometry, computer vision, graphics and HCI) and the efficiencies of tools from the technologically demanding computer game industry. I call this latest version of these tools *ersatz* and it includes an application environment, scene authoring toolkit and a set of API's which allow for the creation of high resolution, interactive, 3D graphic, multi-user, computer controlled environments.

## 2. Art Forms and Cultural Forms

First of all, the type of work I do is not easily categorized by the typical incantations of forms such as multi-media, computer games or virtual reality. It uses aspects of both, but departs from them in significant ways.[2]

In the 90's my work would often be referred to as virtual reality. The description was useful to indicate aspects of form, it was also useful as a cultural idea that the work would call into question.[1] One of the ongoing concerns with my art practice is the transgression of the boundaries of experiences as either mediated or physical. Virtual reality *was* a concept for carefully orchestrating computer mediated experiences to substitute for physical experiences. My work actively promotes inversions between these realms, investigating the ways in which our physical experiences are culturally mediated and the dependency on bodily engagements that effective mediation requires. Thus, I don't utilize devices such as stereography or orifice plugs to deliver my projects within[3] , preferring to create an environment that overtly calls into question the social and cultural relationships that move in and out of the interface to computer mediated realms. However, these works have always utilized all of the advanced methods of virtual environment creation, working with the most developed technologies of virtual imagery.. In the 90's this meant working with technologies that were migrating from the defence industry to the field of scientific visualization. Today it means combining continued advances in those fields with technologies that emerge from the computer gaming industry.

The works differ from video games in that they generally have a unique physical form that makes sculptural and architectural relationships with the computer mediated forms. They also differ from video games significantly in the structure of their content, which may have some referential aspect of games - the maze and navigation are recurring structures along with a self-conscious engagement of first person perspective. To this date, they have not been structured around competition between users, or against a clock, or against a set of obstacles that are overcome to rack up a score. The works do celebrate and encourage play as a strategy of exploration, and require participants to enact them (the works are meant to be viewed and/or participated in and I don't think either approach provides a more important, or meaningful experience, however to be viewed –someone has to be participating in them!).

My interest in creating these works, with all of their iconoclastic yet affine characteristics, has always kept me on the hunt for the best approaches for their creation. Each tool that one uses brings with it a history of conceptions about how it is used and what it is used for. One should understand enough about that history to make interesting choices, and perhaps have that history bleed into the work itself – although being too self-conscious can lead one to a state of paralysis. Part of the artists job is to use this expressive power, and its articulation through its engagement with form, leading viewers to some understanding of intent. If I want some aspect of work to be about hammering, that will have to be evident in the end in some way (excessiveness, oddness and virtuosity are three common approaches).

## 3. Authorship Levels

When I began making works that engaged this sort-of virtual reality/gaming stuff – the primary uses of the technology were with simulation systems, pioneered by military flight training. Computers that could manipulate real-time texture mapped worlds were very expensive and generally used in institutions that would have computer system administrators and computer programmers as operators. I had developed a number of projects that had used smaller scale computer systems to do much simpler graphic environments coupled with

control of environmental sensors and actuators. These previous works had engaged such areas as 3D graphics programming for rendered works, video switching, environmental sensing and control with motion sensors, tilt switches, lights and motors. Moving the approaches that I had developed for the more discreetly separate activities of an installation to a more synthesized version in the virtual reality-like environments – necessitated a careful consideration of toolset approaches.

One sensibility that I bring to any work that I do – is that there is an ongoing conversation between my initial interests and the development of those interests as it becomes articulated through the act of making. My artmaking process is a discovery of truths that come from the manifestation of conceptual structures. In order to achieve this process, there has to be an ability to work fluidly, allowing the final form of the work to reveal itself, rather then strictly imposing a pre-determined vision upon it. I find that this approach leaves behind artworks that continue to generate new meanings through repeated encounters, accepting and demanding the user to intellectually participate in their final articulation – and these work will continue to provide even their maker with discovery long after they are complete.

### 3.1 Low Level

With computer graphic systems we have at our behest a range of tools available. Starting with the physical structure of the chips (CPU, GPU, various memories) these are controlled by some type of operating system through which we can access low-level structures of these devices through programming languages. Above the primitive and direct controls of the registers of these chips are higher level descriptions of their functions which have been collected into Application Programmer Interfaces (API's) such as Open GL or DirectX. These abstract certain ideas such as "load a register with value that will be interpreted as the level of blue in a pixel", to include such things as how to fake drawing a straight line between two points that don't have a simple line of pixels between them[4], or the transformation of a set of polygons with bit mapped images pasted on them in perspective space. Above these graphics API's we have tools that can then be used to codify some more typical uses of these capabilities – such as we might have a whole heck of a lot of these texture mapped polygons that we have to efficiently organize – these graphic objects might also have sound associated with them – or they might represent characteristics such as taking up physical space and have material properties such as impenetrability, mass or jiggle. API's have been developed that provide this level of functionality and don't necessitate engaging in descriptions as basic as describing the structure of a set of polygons as being tessellated or a quadratic mesh. These higher level API's employ clever tricks to optimize graphics performance by utilizing assumptions on how content developers want their programs to operate – such as we make a representation of the world based on the metaphor of a camera that looks at what is in front of it in described by Cartesian rules. In this way – the API can often choose to process and draw sub-segments of an environment, rather then all of the entities described – providing significant increases in performance. The API can also provide interpretive descriptions of surface characteristics that have been developed to simulate lighting, or the animation of these geometric entities in time, triggered or not by user input. These entities and their characteristics (form, texture and motion) are generally described by a combination of procedural rules or via modelling programs (Maya, 3D Studio Max, etc). Other software programs will be used to develop assets such as imaging for surfaces (Photoshop) or sounds (Soundforge). Of course, many more pieces of software are often used to develop aspects of these environments – and for each example given above there are several equally valid substitutes.

### 3.2 Middleware

The API's described above provide powerful capabilities that abstract certain complicated programming operations, but each stage of tools that are used bring with them assumptions about intent and gain some measure of their power by their limiting the type of expressions that they are capable of. API's of this level of capability have been developed in the computer game and scientific visualization industries and are commonly referred to as "middleware". There are middleware API's for such tasks as rendering, physics, networking, sound, user interface control and media streaming. In order to create applications (games, virtual environments or hybrid artworks) the application developer still has to write complex computer programs – although without the middleware, the same end result would possibly require the author to write 10 times more code. Nevertheless the task of creating art with middleware software tools is still onerous. Large software projects often do not benefit from the rather messy process of associative discovery that I described earlier as the basis for my artistic process. Computer programming produces more efficient results with a top-down approach to specifying goals that can be broken down into smaller and smaller tasks.

### 3.3 Modding

With that in mind, it makes good sense to consider alternatives to writing complex computer programs as a basis for artmaking. Within this modal realm, another approach to creating interactive 3D graphic artworks is to hack existing computer games. We use this approach within our undergraduate program at UCSD (The Interdisciplinary Computing in the Arts Program, or ICAM) to teach students what is involved in creating things like games (the development and specification of large numbers of constituent elements) as well as to develop projects that have a cultural criticality, which is often used to comment upon genres of video games. The most often used environment for this is Unreal Tournament. Unreal provides a visually adequate platform and an extensive set of attributes that can be controlled via a scripting interface. It imports elements well from 3D Studio Max and students can create complete projects from scratch in 10 weeks of determined effort. However, every project made will have to work within the narrow narrative structure of a "first-person shooter". The elements of a first person shooter can be changed or sometimes removed, but underlying any work will be this basic structure of narrativity, subjectivity and otherness. Any act upon the external world will be done through the sniper like gaze that enacts the most caricatured example of the oft-critiqued, objectifying, violent, western male gaze.

### 3.4 Taking the Middle Path

In my own case, this trade-off between facility of expression and range of expression has not been of interest in making my artworks. While there are a number of works done with game level modifications that I admire, it provides too much of a pre-authored context about the issues that I'm interested in investigating, to be of any use to me. With that in mind, I have evolved an approach over this past decade to creating these works that provides for the type of top-down programming that is needed to create robust code, and creates a platform for associatively derived expressions, where it is most useful.

## 4. Hacking Supercomputers

This approach is now done under a framework of tools called "*ersatz*". These tools have evolved over the past decade through several projects – some of the key moments of which I'll describe here. Its first incarnation came in 1992 when I began developing a framework for creating a virtual reality project at the San Diego Supercomputer Center at UCSD. At that time, computers capable of transforming texture mapped polygons in real-time through 3-

dimensional space were just moving from being 5 million dollar machines, to 1 millions dollar machines. SDSC had one of these, and it had a middleware API called SGI Performer, which had been developed to facilitate the creation of flight simulators. Performer allowed for data to be imported in a few obscure forms, and at some point a way was found to get data from a common modelling environment – Alias – into a format that Performer could render. However there still wasn't a method to specify what happened to that data once under control of the code, accept though the act of coding. At that point I realized that if one thought about the data and its behaviour in anobject oriented way (you encapsulated data and process together) significant aspects of object behaviour could be specified during the modelling process. However there was no field to specify object behaviour in a modelling environment. In fact the modelling environment that I used (Alias) was a poor one for creating the types of limited capacity virtual environments at that time. It used splines to describe objects, and these were in turn converted into polygons at the time of rendering, which was an operation that took place in un-real time – generating still frames for use in film and video. The only way this environment could be used for behaviour specification was through naming conventions. So a set of name tags was developed that used conventions to specify basic behaviour. Second order behaviours were then specified by phantom structures – simply tree lists of name tags. This could control a higher level of interaction between object nodes. The top layer of control was accomplished through a scripting interface that determined basic things such as what files were loaded, overall tempo, user-interface elements, and other general descriptions.

On the actual code side, we created the general types of relationships that could be specified in this modelling/scripting procedure. In a way, the act of modelling and scripting were akin to the activities of game level modding, only it was my own game levels that I was modding, set up with the primary relationships that I wanted between subjectivity, the other and the media environment. The final project that came out of this approach was a project, called *Apparitions*, which I developed with a group of graduate and undergraduate students over a year and a half of courses that explored the ideas and technologies of virtual reality and explored a variety of approaches for creating tools.[5]

### 4.2 Evolving Complexity

The next evolutionary step in this approach came in the creation of a much more complex project *Mi Casa es Tu Casa/My House is Your House*, which was a bi-national, networked virtual reality playhouse between San Diego and Mexico City. It ran on graphics supercomputers that were two generations more powerful then the Apparitions project utilized, and created a multi-user, shared virtual environment between the two cities.[6]

I use the term evolution to describe the way in which these tools progress - invoking it in the sense that evolution is often prompted by catastrophic events such as a meteor hitting the earth which prompts dinosaurs to become birds. Computer graphics has been evolving in the last decade with that type of evolutionary track. For Mi Casa, graphic tools were available that provided more capability for creating data assets. A modelling tool – Multi-Gen provided both the polygonal modelling needed to create the highly efficient (no wasted polygons) database, along with the behaviour tags that we had laboured before to create. This allowed for much of the expressive areas of the work to be located in the development of this database and scripting environment to coordinate the numerous combinatorial relationships between the multiple users and their actions.

### 4.3 PC gets Virtual Reality and call it Games

The next set of projects that were developed came after a massive meteor strike brought with it a large amount of alien DNA, mutating the form of virtual reality into the form of computer games. The graphics capability of a $250,000 computer in 1995 was available for $250 in 2001. Driving this development was the advent of 3D computer games, and in

particular the first person shooter computer games pioneered by Quake and Doom. Early versions of those games utilized very low level assembler language coding to create primitive 3D graphics environments without the use of specialized hardware processing. They proved the broad demand for this type of capability and companies soon began producing graphic processors, with dedicated vertex processing, on a mass level. This had enormous cost benefits to the approaches that makers of graphic supercomputers had taken – which used multiple general purpose processors in custom configurations to do parallel processing of these operations. The mass market chip in the ubiquitous pc platform provided massive cost/performance improvements.

This change in hardware economics has provided for a rapid development of complex software approaches for increasingly sophisticated interactive graphic experiences. More complex ways of simulating surface characteristics (such as reflections, translucency, texture), shadows, vertex animations (not just objects moving, but the sub-components of objects under parameterized control).

Modelling environments have provided more sophisticated tools for specifying these attributes, but have yet to provide the real-time engines for enacting them. The capabilities for turning this all into an expressive toolkit have become more complex. Entering into this territory, I was commissioned to do a multi-user virtual environment by the California Arts Council. I designed a project that would create six networked kiosks, with large rear screen projections, filling each user's peripheral vision (my usual nod to sensory immersion) with a shared 3D graphic environment that all users would act upon and within- but each getting their own view of this environment depending upon their interactions.

## 5. Application Structure

Initially, I looked to simply develop this project using a middleware renderer and the asset creation software (3D Max, Photoshop, Soundforge) tied together through the necessary code-base and organized by a scripting environment. For much of the development cycle of the project, this proved to be an effective approach. Simply getting the major elements of the project to function (a client-server architecture for coordinating multiple users in a shared 3D graphic environment – turning data from 3D scanners into parts of each users avatar, etc.) drove the development of the underlying application framework. However, after working for about a year on this application framework, the type of discoveries that were occurring in the manifestation of the expressive form of the project were causing the application framework to become an indecipherable mess. It soon became clear that we needed a powerful way to make rapid changes to the environment while focusing on the development of a robust application engine.[7]

For this, we created a tool called the "SceneEditor" which would load all of the data assets of the scene, and allow an event structure to be built between all of the data assets. The key to this, is that extensions made in the application framework are available in the SceneEditor, although the SceneEditor itself is not a part of the final application environment. Thus we preserve the computational efficiency of the application environment – which can never be fast enough, and have a highly flexible environment for changing attributes of the scene.

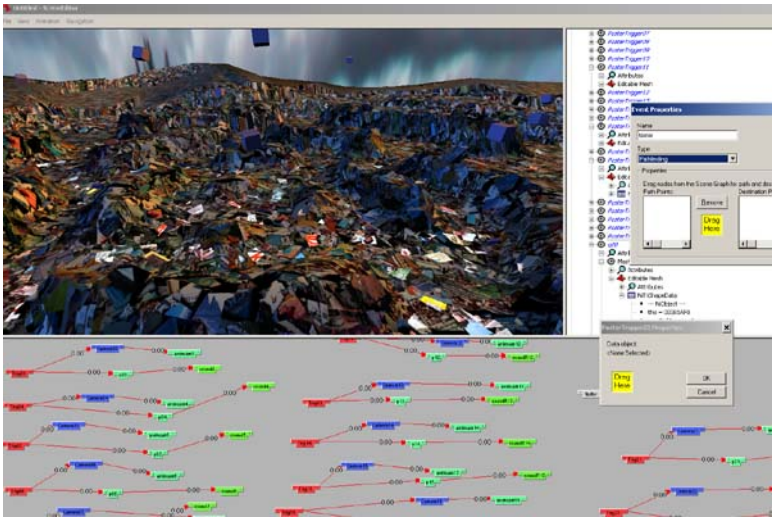

**Figure 1. SceneEditor Version 1 screenshot. This shows the 3 major sections of the application: 1) database (on right); 2) attribute type (in pop-up window); 3) event stream (colored boxes at bottom). The graphic window showing the scene is not important, as it does not interpret the scene events, it simply plays all of the animations in a scene simultaneously**

The SceneEditor has gone through two major versions and is currently in the midst of its third revision. The major elements consist of three areas – the scene database created in Maya or 3D Max, media assets (audio and video which is used in various ways), variables, and the area in which you make relationships between users and these elements. These relationships are made in a two dimensional, non-continuous and infinitely extensible space, where flow-chart like elements are drawn out. Event nodes are created that are of specific types, i.e. floor trigger, path following, play a video, play a sound, start an animation, conditional statements, scene linkages, and camera changes, and many others. These are connected with time values between them. Groups of these events can be encapsulated into black boxes and these can be controlled with variables passed in and out of them.

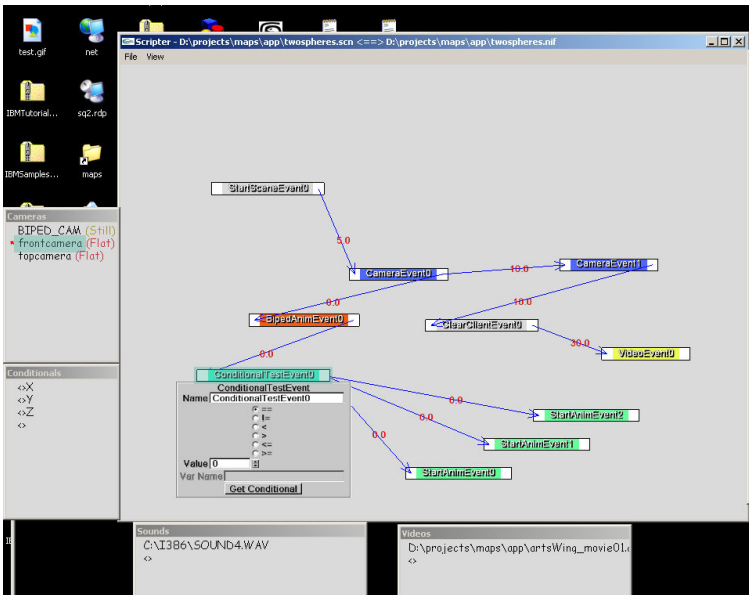


**Figure 2. SceneEditor Version 2 screenshot. Additional facility is added for better interface to procedural systems within the application framework. Also, a better organization of multi-media assets: including video, audio and camera types. The rendered graphic window has been dropped. The changes are only visible when running within the real-time application engine.**

The result of this has been the ability to rapidly apply changes to numerous attributes that had previously been exasperating to try and keep up with by straight code. However it keeps the expressive openness that software development provides for and allows technological invention in the code to be easily controlled at the scene level.

## 6. Evolving Platform

At this time this approach is evolving to bring in a number of new API's to facilitate high resolution, tiled displays and MMPOG support. The projects that are currently in development particularly engage procedurally driven structures within these interactive environments. Creating scene development interfaces to these procedural processes is the biggest challenge to this approach.

### Endnotes

[1] The film industry is a legacy example of how a hierarchical, Fordian production strategy creates a top-down approach to film creation, which is tellingly referred to as film production. Avant-garde filmmakers bucked these approaches throughout the 20th century. We now see the video game industry devising approaches to game development which are generally following a similar model of content specification, technology development and content production occurring in intentional isolation from each other.

[2] The term "multi-media" has undergone significant redefinition, such that what it specifically refers to is very context dependent. One definition of it would be a work that engages a combination of mediums of expression that often are often used in a more singular manner. In the more traditional artworld, multi-media means a combination of forms such as painting and drawing, or sculpture and photography. Multi-media in digital media context developed in the 1980's as a describer of works that combined video, audio, text and imagery. Today, it is a legacy catch-all term to differentiate works from the 20th century cinematic and electronic media, and is similarly meaningful/meaningless as the terms "digital art" or "new media".

[3] This is somewhat of an overstatement. My 1992 installation "The Vorkapitchulator" exhibited in the Machine Culture show at SIGGPAPH 1992 [2], utilized a bizarre, anaglyphic stereoscopic device. The device was an ironic machine of stereoscopic desire, which served as the interface to an assortment of devices which physically manifested digital cinematic forms such as 3D logo's, morphing, procedural animation and montage.

[4] Algorithms such as Bresenham's famous one [3] demonstrate how to create an "alias" of a line that can't be directly drawn on a raster scan screen - a straight line between 2 non-axial points and the use of shaded pixels to perceptually smooth the line, "anti-aliasing".

[5] The project participants included a number of students in several undergraduate and graduate courses. In the end some of the key participants were: Payton White, Brian Duggen, Christa Erickson, Jason Ditmars, Tim Nohe, Andy Mirkis, Mark Tribe, Kelly Coyne, Cheryl Deverauz, Dorota Jakubowski, Eric Riel, and Niklas Vollmer.

[6] For the Mi Casa project I worked with Ryan Mckinley and Harry Castle, as well as with Ramesh Jain and employees of Praja Inc.

[7] This project "Smoke and Mirrors" is currently on display at the Fleet Science Center in San Diego and has been shown in a scaled down version at the PlayEngines show in Melbourne Australia. The project was developed with assistance from Ryan Mckinley, Wes Middleton, Chris Berg, Craig Donner, David Poulin and Joel Murphy

### References


[1] "Real Art and Virtual Reality" by Sheldon Brown, ACM SIGGRAPH Computer Graphics, Volume 31, Number 4, November 1997

[2] "The Vorkapitchulator" by Sheldon Brown, Computer Graphics Visual Proceedings, pg. 116 New York: ACM 1993.

[3] Bresenham, Jack E. Algorithm for Computer Control of a Digital Plotter, IBM Systems Journal, 4(1):25-30, 1965.

[4] Keith, Clinton, From the Ground Up: Creating a Core Technology Group. Gamasutra, August 1 2003. www.gamasutra.com/features/20030801/keith_01.shtml