2006

Oblique reflections: software art and the 3D game engine

Brogan Bunt
University of Wollongong, brogan@uow.edu.au

Oblique Reflections: Software Art & the 3D Games Engine

Brogan Bunt
Faculty of Creative Arts
University of Wollongong
Wollongong, NSW, 2522
+61 2 4221 4642
brogan@uow.edu.au

ABSTRACT
How does new media art conceive its relation to the technological infrastructure that surrounds and enables it? Specifically, how does software art reflect upon the phenomenon of the 3D games engine? This paper considers a range of tactical responses to the dilemmas of scale, encapsulation and conventional aesthetics that the game engine raises for software art. The main focus is on the strategy of anachronism. Anachronism resists the rhetoric of technological novelty, working instead to discover areas of creative purchase within the detritus of industrial (commercial gaming) progress.

Categories and Subject Descriptors
D.3.2 [**Programming Languages**]: Design, Experimentation, Theory

General Terms
Design, Experimentation, Theory

Keywords
Software Art, Alternative Gaming

1. INTRODUCTION
This paper considers how new media arts practice engages with commercial gaming technologies. It is concerned with the various strategies that experimental new media adopts to position itself in relation to means of production that are not its own. My particular interest is in how code-based software art, even when it is not directly producing games, reflects upon the aesthetic and technological phenomenon of the 3D games engine.

I will begin by describing a variety of tactical responses to the dilemmas of scale, encapsulation and conventional (illusionistic) aesthetics that the 3D game engine raises for experimental new media and then move on to a consideration of specific software art examples.

2. DILEMMAS
Let us attempt to clarify some of the problems that commercial games technologies raise for experimental new media arts practice.

Scale and Complexity
The development of a cutting-edge commercial game depends upon huge financial investment and a large-scale, multi-tiered production process that involves work at the hardware, software and creative design levels. Despite the heroic myth of its cottage-industry genesis – stories of John Carmack and John Romero laboring away in relative isolation on **Castle Wolfenstein 3D** and **Doom**; hacking together the architecture and iconography of first-person game navigation through sheer force of geeky genius and popular cultural will – the 3D gaming engine is clearly the complex product of decades of military, scientific and computing/entertainment industry research into the visual and interactive possibilities of computer graphics. This is not to deny the visionary role Carmack and Romero played in developing specific technological solutions and, more generally, in linking emerging trends in computer graphics to the genre of visceral shoot-em-up gaming, but it is to insist that 3D gaming engines, like factories and telephone systems, are sophisticated industrial forms that resist efforts at individual authoring. Their sense of alienating technological scale and complexity presents strategic problems for a field such as experimental new media art where the ideology of individual (or local level) creative control remains important. How can artists engage with the potential of technologies such as the 3D gaming engine when the scale of the technical apparatus radically exceeds the space of individual creative effort?

Encapsulation
Within the field of object-oriented programming, encapsulation refers to the principle whereby code modules – specific areas of data and functionality - are protected from unwanted external modification through the creation of explicit public interfaces and a formal etiquette of access definition. At the general strategic level, encapsulation represents an effort to manage technological complexity, to enable systems to be pieced together in a modular fashion. Code modules are positioned as black boxes that take input and produce output while the details of implementation can safely be ignored. The whole conception of a gaming engine is based upon this principle. Contemporary game engines often encapsulate their functioning to such an extent that it is possible to develop original games without any dedicated work of coding at all. However, this raises a problem of aesthetic purchase. Is it acceptable to bracket the problem of the engine and focus exclusively on the dimension of alternative game content, or is a closer engagement with the underlying technology necessary? There is no single answer to this question. While some so-called 'art games' represent a fairly straightforward work of game 'mapping' (creating new levels and game art for an existing engine), software art takes a greater interest in the dimension of code. To provide some background, although there have been significant software artists (and schools of software art) since the late 1950's, the focus within new media art has tended to be on the visible result of computational processes rather than the underlying creative work of programming. Florian Cramer and Ulrike Gabriel's jury statement for the 2001 Transmediale festival marked a clear shift in focus. Disputing the view that programming represents little more than a context for technical implementation, they argue that the conceptual work of

abstraction and system-building that programming involves is fundamentally creative; 'This award is not about what is commonly understood as multimedia - where the focus is on data that can openly been seen, heard and felt. This award is about algorithms; it is about the code which generates, processes and combines what you see, hear and feel' [5]. At this level then, software art is concerned with the experimental possibilities of code-based generic abstraction and spatial-interactive representation that the game engine represents. It wants to crack the game engine open and reinvent it. The encapsulated character of gaming technologies, the many layers of abstraction and hiding that enable their functioning are a source of both frustration and inspiration, suggesting all kinds of opportunities for experimental intervention and revelation (uncovering).

Conventional Aesthetics

Closely related to the issue of encapsulation is the sense that gaming technologies (particularly game engines) are not neutral entities. They encode specific aesthetic assumptions. For example, the emphasis upon perspective, back-face culling, naturalistic shading algorithms and the like within 3D graphics engines reveals a clear orientation towards visual realism. While the tradition of avant-garde experimental art is suspicious of spatial illusion, commercial media (films and games) position perspective-based immersion as the essential axis of representational and interactive aesthetics. The gliding optical vector of the first-person shooter represents space as utterly seamless. None of this comes easily. The technical problem of stitching a three-dimensional game world together - of enabling smooth movement from one space to another and of managing the display of large complex spaces - is a vital one in 3D engine design. It typically involves the crafty use of portal systems so that the player constantly moves between partial worlds (from one rapidly loading data structure to another). The whole world, as such, never exists. The illusion of holistic space is a bubble with the player at its center. The bubble changes as the player moves about and everything else is darkness (or the barest map). This is, of course, conceptually very interesting, suggesting links to notions of the Cartesian subject, etc., but it is not explicitly highlighted within commercial games. It is represented as a technical problem rather than as a creative option worth exploring. There is a vital need then to engage with the mechanics of the engine to open up other aesthetic possibilities.

3. RESPONSES

These dilemmas mean that commercial gaming engine is positioned awkwardly for new media art. It is tempting (constituting an iconic popular form of virtual interaction and suggesting vital areas of aesthetic enquiry), but at the same time arcane, inaccessible and hidden. How has experimental new media dealt with this problem? How has it conceived and practically negotiated its relation to the technical means of production? Five strategies seem evident.

Alliance

The first strategy involves an alliance between artists and computer scientists. In their famous 1825 article, 'The Artist, the Scientist, and the Industrial: Dialogue', the social philosophers Saint Simon and Leon Halevy suggest the possibility of a utopian accord between art, science and industry, in which these traditionally separate disciplines form an alliance to advance progressive societal interests [16]. From a contemporary perspective, informed by the legacy of critical theory, post-structuralism and postmodernism, this early vision of the avant-garde is likely to appear naïve. We are less confident about the benign legacy of Enlightenment reason and very suspicious of the rhetoric of progress. Nonetheless, contemporary new media often summons up the rhetoric of a progressive alliance of artistic, scientific and industrial interests. The British collective Blast Theory provides an example of this approach. Their augmented reality games such as **I Like Frank** (Adelaide Fringe Festival, 2004) explore the poetic relations between real and virtual spaces and players [2]. Produced in collaboration with Nottingham Mixed Reality Lab, Blast Theory supplies the creative vision, while the Mixed Reality Lab researchers provide the cutting-edge technical infrastructure. Blast Theory is also involved in a larger research initiative, 'Integrated Project on Pervasive Gaming' (IPerG) which links together a range of creative and scientific organizations with the aim of developing 'a radically new game form that extends gaming experiences out into the physical world.' This involves exploring 'new technologies to support the creation of new compelling forms of content' [8]. It is worth noting that this strategy of alliance preserves a very traditional distinction between creative and technical contributions. Art focuses on the conceptual imaginative realm, while science focuses on the underlying engineering. This strategy maintains its critical aesthetic credibility by occurring at a slight remove from the realm of industrial, commercial application. Alliance is pursued in the guise of art-science collaboration, rather than as industrial R&D.

Abstraction

Like the previous strategy, abstraction accepts the conventional distinction between the field of creative design and technical implementation, but rather than entering into an alliance with the technological vanguard (whether conceived in scientific or industrial-commercial terms), it operates in isolation from them. Instead of portraying the possibility of an avant-garde that combines cultural-aesthetic and technological novelty, the sphere of the cultural-aesthetic becomes separated and abstracted from the technical. The focus shifts to the game concept as an abstract space that precedes any particular form of implementation and that can take shape experimentally without the machinery of cutting-edge gaming technology. This is the approach that Katie Salen and Eric Zimmerman adopt in their innovative account of the field of game design, **Rules of Play: Game Design Fundamentals** (2004) [15]. They open up the potential for an alternative, creative and theoretical space of game design by deliberately bracketing issues of implementation. This has considerable value, especially as commercial gaming works within such an impoverished conceptual space, but also clearly represents a strategic withdrawal from the problems of scale and complexity that the technological dimension of contemporary gaming presents. So while strategies of abstraction risk devaluing the creative, imaginative dimension of technical implementation, they play a key role in delineating avenues of dedicated conceptual-aesthetic interest. The politically oriented web games of Gonzola Frasca provide an example of this approach. "September 12, A Toy World" (2003) [6] employs the model of a simple isometric shooting game to make a critical point about the uselessness of addressing the 'war on terror' via missiles. The originality of this game lies not in its technical features, nor even in its mode of game play, but hinges instead upon a work of conceptual recontextualisation. It is an experiment in employing games as a form of simulation-based political commentary. The technical and generic remain important as ironic points of

reference, but the key creative work is abstracted from issues of implementation.

Aggregation

The sense of effective exclusion from the arena of cutting-edge technological development leads to another strategy: the formation of communities of cottage-industry level producers who together build alternative game-related graphic engines and the like. The **Ogre** [10], **Blender** [3] and **Xith3D** [18] communities provide examples of this approach. While certainly building sophisticated pieces of graphics technology, their collaborative work is not positioned as technologically cutting-edge. Instead the emphasis is upon access, upon providing means for small independent producers to engage with the esoteric and typically proprietary space of contemporary gaming technology. However, for my purposes, very few of these communities are oriented towards the sphere of experimental new media arts. Rather than questioning the aesthetic assumptions of commercial gaming technologies, they are more likely to provide ever so slightly pale copies. Undoubtedly the engines can be put to other uses, but any work of fundamental modification is likely to occur in a less collective context. Paradoxically, more relevant communities have a less explicit relation to the realm of gaming. The **Processing** community [11] focuses on providing artists with access to the creative space of programming, supplying technologies and a supportive context for the development of experimental projects that explore alternative possibilities for 3D rendering and the like. It supplies nothing like a game engine, because the focus is not upon games as such. The **Processing** environment is much more concerned with enabling artists to engage with code (and the aesthetic possibilities of code) at a more fundamental level. It deliberately strips away scale, complexity and encapsulation in order to establish a technical context in which genuinely creative questions can be posed. In its tactical effort to simplify, **Processing** has conceptual affinities with the previous strategy of abstraction. **Processing** aggregates precisely in order to enable individual, cottage-level experimental practice.

Appropriation

The popular practices of 'modding', 'mapping' and hacking bits of commercial game technology to produce alternative critical or whimsical pieces of new media art provide examples of appropriation. Gaming engines have been 'appropriated' to enable, among other things, abstract animation and drawing, data visualization, performance, political satire and film-making (machinima) [12]. At times appropriation can represent a deliberate assault on proprietary game formats. Cory Arcangel's (aka 8-bit Collective) **Super Mario Clouds** (2003) [1] hacks into the hardware of the Nintendo game cartridge to strip away everything in the game but the floating background clouds. However appropriation can also work in more agreeable harmony with the interests of commercial gaming. Many mainstream game developers allow and encourage efforts at creative modification and reconfiguration. They release source code and mapping tools to facilitate grass-roots production of new versions of an original game. They deliberately position their products as emergent cultural and technological phenomena. The interesting implication is that commercial games, as abstract engines and as generic fields of parametric possibility, may be said to logically anticipate all their various aesthetic appropriations. The game engine is a protean meta-level space of aesthetic potential that imaginatively encompasses all of its specific creative instances - even those that criticize and deconstruct it.

Anachronism

This strategy, specific to software art, engages closely with technology but in a distinct, deliberately 'out-of-time' critical-aesthetic manner. Excluded from - and avoiding - the rhetoric of technological novelty, anachronism tinkers, reflects, reconstructs and re-imagines aspects of the computational heritage. It overlaps to some extent with strategies of appropriation but places a greater emphasis on 'original' coding.

Sketching a cultural context for software art, Lev Manovich [7] suggests that whereas the postmodern media artist (of the 1970s - 1990s) engages in a work of pastiche and appropriation, the software artist (of the late 90s and early 21st Century) insists upon a creative *tabula rasa*. The emphasis shifts to coding things from scratch, avoiding both the tools and illusory mimetic rhetoric of contemporary commercial new media (animation, games, etc.). According to Manovich, this represents a return to an earlier model of artistic practice – the model of the 'romantic/modernist' genius. Instead of drawing cynically upon the available media culture, iconography and creative tools (with the sense that there is no viable aesthetic space beyond), the software artist 'makes his/her mark on the world by writing the original code.' He stresses that '[t]his act of code writing itself is very important, regardless of what this code actually does at the end' [6, p.211].

While this work of writing is clearly very significant, I am not convinced that it summons a pure terrain of original expression. Indeed the small qualification that Manovich makes - the acknowledgement that this code may be inconsequential, that it may not do anything especially significant or novel – suggests a tension and uncertainty surrounding the nature of 'original coding'. The blank sheet of code is not a simple surface. It is both a veil and an unveiling. It is both clean (creatively open) and thoroughly inscribed. It floats above a framework of encapsulated processes that extend down to the hardware level and is structured as a palimpsest, in which the software artist repeats, writes and interrogates the coding tradition. The software artist makes his/her 'original marks' in the space that is left once technological progress has moved on. Originality lies in summoning up a dimension of alterity within this abandoned landscape, discovering through self-conscious anachronism ('non-original' coding) a field of aesthetic possibility. Whereas the direction of commercial technological development is to develop more and more sophisticated layers of abstraction that work to make human engagement with computer processes as intuitive and kinaesthetically engaging as possible, software art deliberately returns to the earlier, retro model of arcane text-based interaction. The GUI (and the dream of the GUI) disappears to be replaced by the IDE and the text console. Software art partakes of anachronism in its very concern to structure human-computer interaction in terms of the traditional metaphorics of programming.

The strategy of anachronism then engages creatively with the technological tradition by deliberately withdrawing from any attempt to appear at the cutting-edge. The aim is less to project an unseen future than to re-imagine and re-invent the computational wheel, to work over the detritus of technological progress searching for points of creative intervention. Anachronism acknowledges the asymmetry between art and the space of technological development, but insistently searches for means to reflect upon the technical, to open it up to a process of critical-creative enquiry. In the process, the relation to gaming

technologies often becomes indirect. While there are many artists producing alternative games, there are significantly more engaged in formal experimentation with aspects of 3D drawing and rendering. This genre of creative practice (very prominent in the **Processing** community) represents a response, at least partly, to the conventional illusionistic assumptions that inform the structure of the commercial game engine.

I have described these strategies separately, however it needs to be acknowledged that they very often communicate and overlap. For example, although abstraction, as I have defined it, resists engaging with the sphere of technical implementation, it has vital importance in terms of describing a dedicated space for conceptual-aesthetic reflection. Anachronism, at its best, incorporates a dimension of abstraction; it distills the conceptual-aesthetic relevance of specific technical processes rather than simply reconstructing them. Similarly, strategies of appropriation can often blend into strategies of 'original authoring' (anachronistic re-invention). Appropriation finds itself opening on to an original space while attempts to code from scratch discover a relation to the legacy of coding achievement. Even alliance can reveal other dimensions. The Blast Theory augmented reality projects, despite the rhetoric of avant-garde interdisciplinary accord and technological novelty, represent an impressive effort to re-orient aspects of standard industrial R&D; to gently appropriate science towards an investigation of critical-poetic issues related to virtual identity and emplacement.

The five strategies represent less a static set of antagonistic options than a dialectical constellation - a field of productive tension. The key tension concerns how the relation between art and *techne* is conceived, but there are also more subtle tensions concerning issues of originality and the tactical relation of art to the broader sphere of technological progress.

4. SOFTWARE ART EXAMPLES

With this general scheme in place, let's consider how software art reflects on the 3D game engine. My interest is in how the various tensions that I have described above are played out within a specific new media arts context and in the texture of specific software art works. My initial focus is on two exemplary projects of appropriation (resource hacking) – JODI's **SOD** (1988) and **Untitled Game** (2002). These visionary works reconfigure the **Doom** and **Quake** engines and anticipate vital paths of investigation for contemporary software art. If **SOD** and **Untitled Game** address the 3D games engine directly, the relation is more oblique within contemporary software art. A consideration of one of my own recent projects will provide a means of clarifying the nature of this relation.

JODI Game Modifications

JODI is the duo of Dirk Paesmans and Joan Heemskirk. Their modifications of the **Doom** and **Quake** engines are sublimely deconstructive reflections on the formal architecture of the first-person shooter. Although their work involves code-based intervention, it is clearly not software art that begins with a blank page (the imaginary, theatrical scene of a blank page). It is work that explicitly highlights the slippage between postmodern strategies of appropriation and (undecideable) strategies of 'original authoring'.

SOD [13]

SOD hacks the **Doom** engine to represent the grim corridors of the original game as abstract black and white shapes. Stripping away the illusion of figurative, textured, shaded space and maintaining only minimal perspectival cues, **SOD** highlights the underlying architecture and artificiality of first-person space. Structural features such as the option screens, HUD (heads up display), portals and targeting system gain a new and uncanny visibility. Whereas in ordinary game play, these features support the game play and remain subservient to it, here they are fore-grounded through deliberate strategies of abstraction. Option screens become lists of semantically void geometric shapes. The HUD displays numerical information about a game space that we can only marginally engage with. Doors (portals) float in space, manifesting forms of transition that undermine any naturalistic conception of a doorway and that very evidently involve the sudden loading of new spatial data. The only element that remains largely unchanged from the original game is the sound; it provides a residual sense of spatial integrity and indicates that despite the obvious work of modification we remain in the **Doom** engine space.

Untitled Game [14]

JODI's deconstructive strategies become even more radical in **Untitled Game**. Working now with the **Quake** engine, **Untitled Game** is a set of fourteen game variants that explore the coded-ness (or metaphysics) of 3D games. Gone, on the whole, is any lingering concern with maintaining aspects of three-dimensionality. The focus is on the pre-space of conceptual abstraction that shapes the underlying possibility of perceptible game space. This is evident in the title itself which playfully employs the archetypal name within abstract art ('untitled').

The names of the individual game variants are also worth considering. Many indicate what appear to be logical ranges within the alphabet – **A-X**, **G-R**, **M-W**, however the ranges clearly overlap and bear no relation to the content of each game. Other games are named after command key combinations – **Ctrl-9**, **Ctrl-F6** and **Ctrl-Space** – however the games make no apparent use of these combinations. Only three of the games are named in a more ordinary descriptive manner (**Arena**, **Slipgate** and **Spawn**). The generally arbitrary character of the names makes the gulf between the sphere of language and reference (engine processes and 'game plsy') very explicit. The names are indicative of the central critical, deconstructive concern with the disjunction between the spheres of coded representation and spatial perception.

Turning now to a brief analysis of three of the game variants:

Arena

Arena (Figure 1) represents a sublime near-zero point of the **Quake** engine. There is the sound of attacking enemies and the player can click and fire, but 3D space itself has been altogether eliminated, leaving only a framed white screen and the HUD. This variant points to the non-space at the heart of 3D simulation and stages it literally, visually.

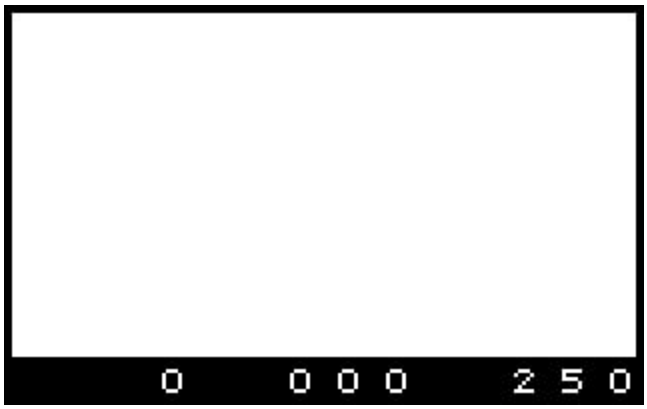

Figure 1. *Arena*

A-X

A-X (Figure 2) dispenses with even more features of the original game. There is no longer even the frame or the HUD. The player encounters a cascade of data; they encounter 3D space as the engine (at some relatively high-level) conceives and processes it. This is a particularly clear example of the critical focus on the discontinuity between code and the illusion of space.

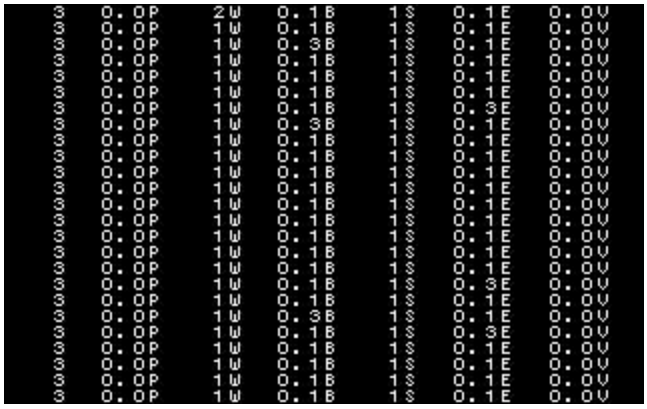
Figure 2. *A-X*

Q-L

Q-L includes recognizable aspects of **Quake** 3D space, but in a totally vertiginous manner. Instead of predictable visual orientation and motion, the camera spins wildly out of control and none of the usual interactive controls work as expected. This game variant works then to unsettle the sense of continuous 3D space and confident first-person motion through space. It suggests that spatial continuity and first-person interaction are only a fragile mathematical fiction.

The exit screen for **Q-L** (Figure 3) indicates a characteristic deconstructive strategy employed in **Untitled Game**. Instead of the usual set of options - 'New Game', 'Single Player', 'Exit', etc. – the user encounters a jumbled set of letters that are semantically meaningless but that adhere to the formal layout of the ordinary options screen. Anyone with experience of **Quake** can infer that 'PTJS' signals 'EXIT' because it is four letters long and positioned where 'EXIT' would normally be.

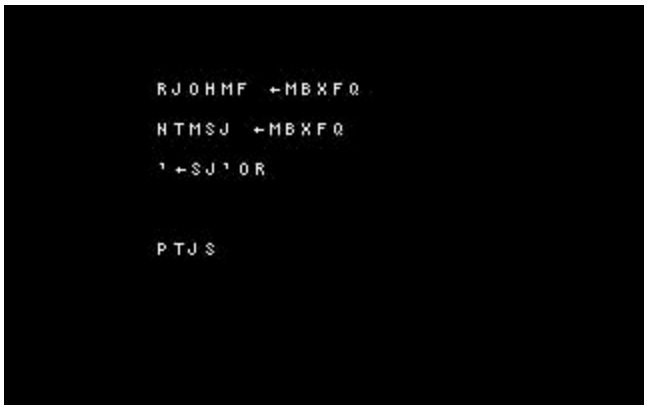

Figure 3. *Q_L*

This strategy of maintaining formal identity while simultaneously engaging in a work of semantic scrambling is a key characteristic of the JODI modifications. However much they deconstruct the **Doom** and **Quake** engines, they also remain true to aspects of their underlying structure. JODI's work obtains its critical imaginative (and political) force precisely inasmuch as it develops a tension between the formal mechanics (and cultural imaginary) of game engines and a more open space of creative possibility. Their work represents a limit form of game modification. While remaining within the orbit of **Doom** and **Quake**, they signal concerns that extend beyond the field of 3D games as such. The **Untitled Game** variants, for example, are very clearly not games. They are unplayable critical interventions that focus upon the underlying logic of spatial representation – upon the engine as a conceptual system rather than the structure and articulation of game-play. As much as they play upon the formal features and iconography of **Quake**, they also manifest a fundamental and more general concern with the abstract logic of code. Working at the limits of appropriation, they indicate the necessity of other approaches; ones that not only modify engines but also imagine them differently. In this sense the JODI modifications anticipate contemporary strands within software art that focus on the creation of alternative, typically abstract, graphic engines.

Contemporary Experiments

Trawling through the **Processing** forums [10] it is possible to find many examples of posts such as the following:

> Hello, As my brain is starting to smoke, and google cant seem to give me an understandable answer, turn to you for a possible solution. my problem is as follows: I have a point defined in polar coordinates (Zrotation, Yrotation and distance), now i need to find out what the absolute rotation is, by that i mean a single rotation around a defined axis that returns the same point in space. I've been speculating that this angle is sqrt(Zrotation^2 + Yrotation^2) but i havent been able to verify this. Is there anyone out here that have a better understanding of this than me? also i would like to know how to calculate the axis of this rotation. hope my explanation is understandable: Regards, Henrik, IP Logged

Along with Henrik, many software artists are becoming increasingly absorbed in the technical intricacies of 3D graphics. A huge amount of energy is being devoted to solving entirely trivial mathematical and programming problems, ones that have been solved innumerable times in the past (and much better). To code with this sense of larger irrelevance, with this awareness of

stupidity and anachronism - this is a substantial part of what it means to be a contemporary software artist. Typically there is no explicit aesthetic rationale; ostensibly it is just about tinkering around with the mathematical and technical infrastructure of 3D graphics. However, this may actually indicate the key point; this process of tinkering is about bringing the problem of technological complexity down to a human scale. Each specific technical problem is tantalizingly soluble at a local, human level. In this sense, each question projects a horizon of distant but attainable technical competence. The legible hope is that artists can obtain the relevant skills and understanding to direct code in their own aesthetic interests rather than inevitably be swept along by existing technical regimes and aesthetic assumptions.

The technological infrastructure of the 3D games engine provides a crucial reference point for this work of contemporary experimental practice. The explosion of sophisticated graphics technologies over the past two decades is closely linked to the increasing economic and cultural sway of real-time 3D games. OpenGL, DirectX, superbly fast graphics cards – all bear the imprint of the commercial gaming industry. If software artists now have access to aspects of this technology it is substantially due to its popularization within games. The JOGL (Java bindings to Open GL) API, which is vital to **Processing** and more general Java-based software arts 3D experimentation, provides a clear example of this debt. Although the API can certainly be applied in contexts that extend beyond games, it is nonetheless the product of the Game Technology Group at Sun Microsystems and can be found at the java.net site by following the following set of hierarchical links; projects, games, games-core, jogl [9]. While software art makes use of this technological infrastructure, gaming itself, as a cultural form, is often only obliquely acknowledged. Whereas the JODI projects explicitly confront the culture and aesthetics of commercial gaming, software art imagines a fragile space of autonomy. Gaming is positioned, in a contradictory fashion, as both a necessary foundation and as an extraneous imposition.

Anachronism (Again) [4]

To illustrate this tension within the self-identity of software art, its sense of relation and non-relation to the larger technological and cultural infrastructure, I will consider the development of one of my own recent software art projects. This project is entitled 'Anachronism**'** in order to highlight the awkward relation to the means of production that is constitutive of software art. I am focusing on my own project here not with a sense of its aesthetic importance but because I can provide an under the hood explanation of how the work is informed by a relation to the 3D games engine.

Anachronism began as a kind of perverse Java 2D sketching program. The idea was to eliminate all sense of an analogue relation to manual drawing. The user would quite literally draw with numbers; defining shapes by writing a series of x and y coordinates (and control points for Bezier curves) to a text file. An additional configuration file would describe sprites, motion and rendering styles. It is worth noting that although none of this engaged closely with the possibility of the 3D games engine, many of the fundamental concepts informing the structure of this experimental drawing program have their basis in gaming technologies. The whole conception of a 'sprite' as a screen instance of a graphic data structure stems from gaming, initially as an aspect of graphics hardware and then as a software abstraction [17].

Having created a version of this initial - deliberately contrary - drawing program, I became more interested in the creative possibilities of code drawing itself, and especially in the potential to draw with animated 3D shapes. It quickly became evident that it was impossibly difficult and long-winded to manually define 3D shapes, so I switched to the algorithmic definition of simple shapes and the parsing and loading of Alias Wavefront 'obj' files. The latter represent shapes as lists of vertices and polygonal faces and can be created in a variety of 3D modeling applications. My aesthetic rationale was to explore alternative, non-figurative means of 3D rendering. This represents a characteristic gesture of resistance to the predominant focus upon visual realism within commercial games and animation. It follows the trajectories suggested by the JODI projects, but would seem to articulate them in a less politically pointed and deconstructive manner. If my first concept playfully juxtaposes code and the ideology of intuitive aesthetic perception, my second encapsulates the dimension of code drawing in order to elaborate a wider space of visual possibility.

However there are also more subtle implications. The shift to 3D prompted a more explicit concern with the graphic-related structure of the 3D games engine. My interest was in stripping back the graphic operations to a bare minimum. There would be no back-face culling, no painter's algorithms, no binary partition trees, it would simply be sets of polygonal objects that could be animated and drawn as points, lines or filled shapes. I actually avoided OpenGL (JOGL) and worked with simple Java 2D drawing methods. This deliberate work of bracketing core areas of functionality was the key to opening up original creative possibilities. Suddenly in the interstices of the conventional engine (here re-written from scratch) there was the potential to explore something other than the simulation of space; something that entered into to dialogue with traditional drawing, that was concerned with the deliberate fashioning of shapes, iterative patterns and conceptual series (Figure 4).

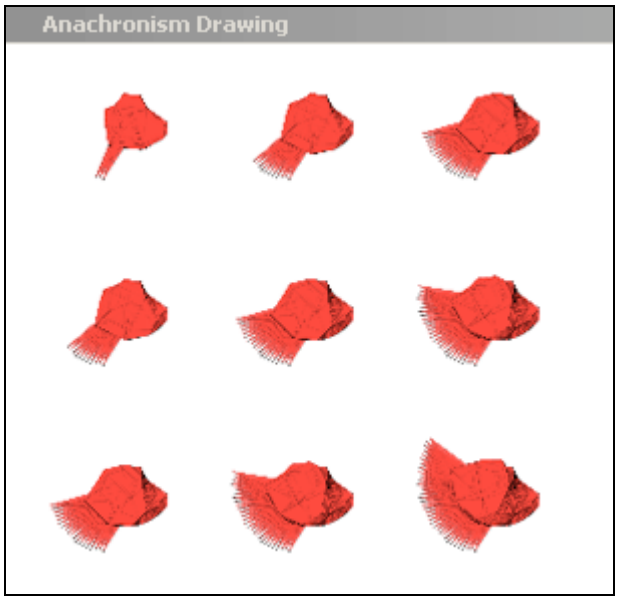

Figure 4. *Anachronism*

The creative work emerges then in the friction between the conventions of the 3D graphics engine and the experimental agendas of software art. **Anachronism** is interesting precisely in terms of the tensions that structure its autonomy and originality. However, there is the difficulty that this underlying dynamic may

not be directly evident in the work. It figures as a background and is articulated obliquely. Perhaps the title makes the point, but the question remains a vital one for software art: how can the concepts and contextual constellations that inform the creative work of programming become lucid at the level of the perceptible work?

5. CONCLUSION

This paper has attempted to describe the awkward relation between contemporary software art and the 3D games engine. It has considered the broad dilemmas of scale, encapsulation and conventional aesthetics that the 3D games engine presents as well as suggesting a range of specific strategic responses. My key interest has been in the ambivalent character of strategies of anachronism. Anachronism appears both original and non-original. It both imagines prospects of creative autonomy and acknowledges relations of dependence and dialectical differentiation. If there is a problem in all of this, it is less in terms of issues of logical contradiction than in the all too common failure to tease out the conceptual implications of anachronistic practice. There is a crucial need for the critical character of 'technological tinkering' to be elaborated within software art. Processes of interrogation that may be apparent to the programmer need not be apparent to the user/viewer. If experimental graphical software art is to avoid being interpreted as apolitical and blandly decorative, then it needs to discover ways to articulate underlying conceptual concerns (and the politics of its problematic creative positioning) more explicitly.

6. REFERENCES

[1] Arcangel, C. *Super Mario Clouds*. (game modification of Nintendo cartridge Super Mario game), web version available at http://beigerecords.com/cory/ , 2003

[2] *Blast Theory*, http://www.blasttheory.co.uk/

[3] *Blender 3D* (open-source 3D application), http://blender.org

[4] Bunt, B. *Anachronism* (experimental Java-based 3D rendering engine), http://brogan.com.au/

[5] Cramer, F. and Gabriel, U. *Software Art*, Transmediale Fesitval, Berlin, 2001

[6] Frasca, G. *September 12, A Toy Story*, http://www.newsgaming.com/games/index12.html

[7] Manovich, L. '*Flash Generation*' in Hui, W., Chun, K. and Keenan, T. (eds.) *New Media/Old Media: A History and Theory Reader*, Routledge, New York and London, pp. 209-218, 2006

[8] *Integrated Project on Pervasive Gaming*, http://iperg.sics.se/

[9] *JOGL* (Java Bindings for OpenGL API), https://jogl.dev.java.net/

[10] *Ogre* (open-source 3D graphics engine), http://ogre3d.org

[11] *Processing* (open-source experimental software art IDE and community), http://processing.org

[12] *Selectparks* (alternative gaming site), http://www.selectparks.net

[13] JODI, *SOD*, http://sod.jodi.org

[14] JODI, *Untitled Game*, http://untitled-game.org

[15] Salen, K. and Zimmerman, E. *Rules of Play: Game Design Fundamentals*, MIT Press, Cambridge, Massachusetts, 2004

[16] Taylor, K. (ed.) *Henri Saint-Simon 1760-1825: Selected Writings on Science, Industry and Social Organisation*, Croom Helm Ltd, London, 1975

[17] *Wikipedia* (entry on graphic sprites) http://en.wikipedia.org/wiki/Sprites_(computer_science)

[18] *Xith3D* (open-source 3D graphics engine), http://www.xith.org