Buffering Bergson: Matter and Memory in 3D Games.

Julian Oliver

“I find, first of all, that I pass from state to state”
(Henri Bergson, *Creative Evolution*)

Henri Bergson would have liked 3D computer games, or at least the way they are made. Naturally this may seem unusual given the field of 3D graphics is itself the grandchild of Euclids *Elements*, a geometric construction of the Universe as a mesh of points connected by measurable lines. However contemporary 3D games deploy a range of techniques in their presentation that differ greatly from the alienable universe as it is mathematically observed; at the center of the 3D game there lives a conspicuous, albeit unlikely echo of Bergson's (less than popular) ideas.

Bergson believed in a kind of creative evolution whereby the perceived world is itself in a state of perpetual change, a morphology produced at the intersection between memory and actions derived from experience. Bergson countered the idea that this state of change is outside the subject, rather the subject produces change through action itself. This annoyed Bergson's contemporary, Bertrand Russell. In a famous showdown Russell insisted that there is no such thing as a 'state of change' so much as a scientifically determinable series of observable states.
Russell was an empiricist, and so was frustrated by Bergson's notion that time exists as a continuum of perceived *durations*, a self-preserving past that inevitably converges upon a mutable present:

> **"In reality, the past is preserved by itself automatically. In its entirety, probably, it follows us at every instant; all that we have felt, thought and willed from our earliest infancy is there, leaning over the present which is about to join it, pressing against the portals of consciousness that would fain leave it outside."** (from *Creative Evolution*)

On this level, Russell's universe is comparable to the frame by frame transformations in a 3D game, each a state of transformation relative to the last. But in Russell's world, the user is innately independent from the unfolding of the world itself. It's here that Bergson's morphological universe finds company in the architecture of a 3D game engine.

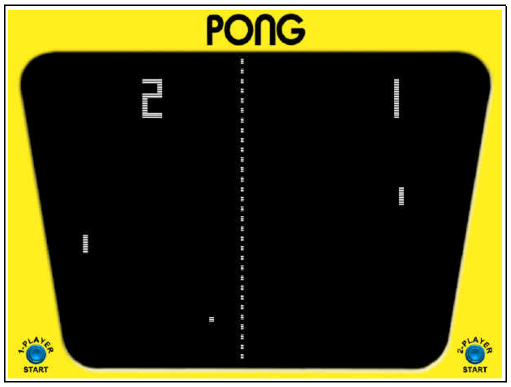

Fig 1. Still from the game, Pong. Basic forms ('primitives') comprise the elements drawn to the screen.

Right from the origins of the video-game as a flat field of interactive primitives[1] to the rich media-scapes of the contemporary game, one artifact remains intact, that of the in-game camera. Both a trigonometric construction and the hemisphere around which action is grouped, this window to the game-world delimits the periphery of our immediate capacity while also serving as the context from which we chart our vectors in play. At first it seems the worlds of games somehow pre-exist the camera; that regardless of whether or not we're playing, the shape of the world itself persists platonically. However this is not the case. The very mathematical construction of this camera, and how the world is drawn around it, defines a peculiar, and even metaphysical difference between the worlds of game and life. In a 3D game, it is the viewer that *produces* the world. When I say 'produces' I don't mean 'make' or even put in a shiny box, so much as usher or *bring into effect.*

In a 3D game the camera deploys what is called a 'projection matrix', which can be thought of as a geometric keel against which all visible objects are shaped and sheared. Each object in a 3D game is comprised of many differently oriented triangles that comprise 'meshes.' These 'meshes' can be skewed during play to give the illusion of depth, drawn relative to the orientation of the camera. There is in fact no 'depth' in a 3D world, just a preferential stacking of triangles drawn around the attention of the software camera. This order of drawing is done in what's called the 'Z buffer', where Z is the axis of representative depth. When we are looking at a scene of hills, giant killer goats and inviting bridge in a 3D game (for instance), we are in fact looking at a flat surface of triangles, each angularly attenuated to give the appearance of form (Fig. 2). Form itself however, in the sense of separable, material integrity, doesn't and has never existed in a 3D game. As the player moves the camera around the world, these triangles flex and foreshorten, some drawn on top of others, and with different surface effects, all of which help inversely affirm the player as the occupant of a visually distinct location.

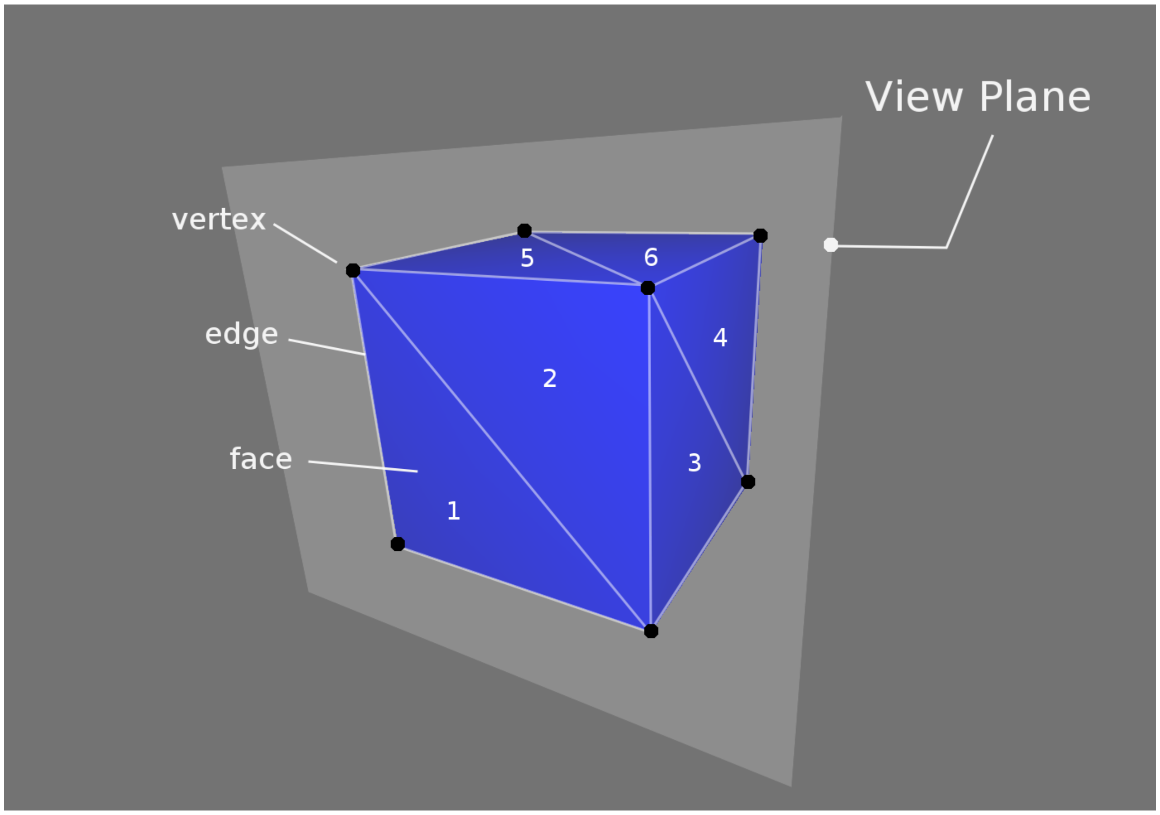

Fig 2. Illustrates the translation of 6 3D triangles (faces) into the screen area of the client computer system, giving the illusion of a 3D cube. The Projection Matrix calculates the transformation or skewing required to give this illusion, while the 'view plane' is the flat area where pixels are finally drawn on the computer screen. This is also where the 'front clip plane' is typically found, which ensures nothing 'behind' the viewer is drawn.

But there is another question, not just of what is drawn and when, but how much. *"How much world can I see"?* Here 3D game development takes a very different approach to the construction of worlds - not out of conscious design but from an architectural necessity whose roots reach down to the system hardware itself.

Game data comes in a variety of types. Some are *persistent* (User Interface, soundtracks, player character mesh and items) while others are *non-persisten*t, meant only to be encountered when relevant to the current stage of gameplay (non-player characters, other players, a hill, a flying whale). As the camera is pushed around the world by either the human player at the input device, or entities in the game, non-persistent game data is 'culled' from view and new data is drawn current to the camera position. Parts unseen by the camera, like the back of objects or objects immediately behind the camera are removed from view giving preference to logically perceptible objects. **Fig 3.** describes this process.

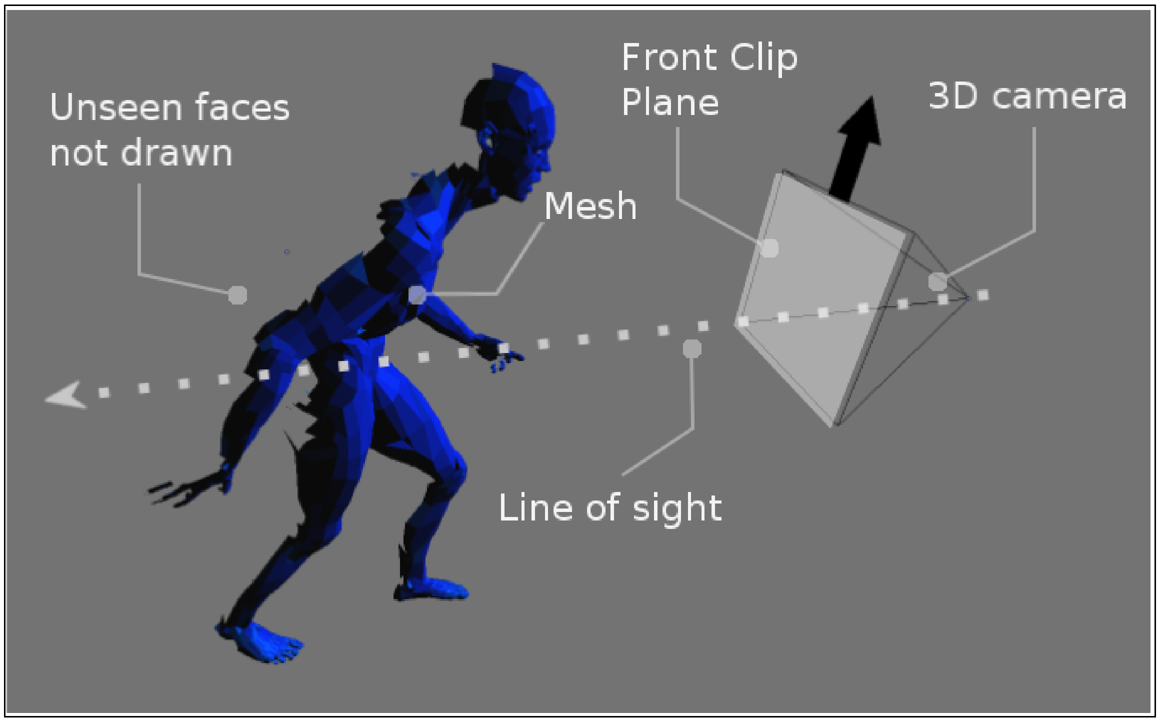

Fig 3. Illustrates the technique of 'Backface Culling'. Mesh 'faces' (triangles) not seen from the perspective of the player's 3D Camera are not drawn.

Another technique used (often concurrently) is that of 'far clipping'. This is best understood as a maximum viewable distance or hard limit of ocular perception designed to draw only what is considered close enough to be relevant to the scene. The 'far clip plane', together with the front or 'near clip plane' comprise the 'view frustrum' or *renderable region* in a 3D world. These two technologies, of 'culling' and 'clipping' are designed precisely to the ends of restricting what can exist in the physical graphics memory of the gaming platform. Here the world, as a quantifiable universe of measured and positioned parts, is both held and shaped by the limits of *memory*.

In a 3D computer game, the subject both reveals and shapes the world as a surface of transformational effects from which we produce our own frames of recounted experience. It isn't so much that we 'create' the world in a 3D game, as evolve, manifest and disappear it's parts in and out of shared memory as a function of action itself. This configuration, where the perceiving subject is intrinsically bound to the unfolding of the world itself, is ulterior to popular understanding of how our own corporeal world works; a persistent, indifferent enclosure of which we are subject visitors.

It's therefore notable that the art of creating 3D worlds has itself derived from the traditions of mathematics and geometry, two sciences that seek to describe and project universal relationships in spite of the subject. Where strict computational simulation of the real in concerned, such languages of universal description have retained integrity (envisage simulation of a car impacting with a wall, or of sunlight acting on a glass). However, once a player is introduced, with choices, actions and the capacity to navigate, the technology must adapt to support a world that is *produced through and for the subject*.

Bergson was often asked, "so who does the remembering, us or the world?". In a game, and in Bergson's universe, it's always a bit of both.

1. 'Primitive' is a term used in graphical programming to describe basic geometric forms, like plane, cube, sphere, cone, cylinder.

References:

OpenGL Architecture Review Board, ***OpenGL Reference Manual**,* Addison Wesley Publishing Company 1994.

Henri Bergson, ***Matter and Memory**,* Zone Books 1991.

Henri Bergson, **Creative Evolution**, Dover Publications (Unabridged edition) 1998.

I Mueller, ***Philosophy of mathematics and deductive structure in Euclid's 'Elements'*** Cambridge, Mass.-London, 1981.

Julian Oliver is a software developer, artist and teacher. In 1998 he establised the game-based artist collective Selectparks (http://www.selectparks.net) and is currently based in Berlin.
More information available at http://www.selectparks.net/~julian