UNIVERSITY OF CALIFORNIA,
IRVINE

Structuring Feeling, Feeling Structure: Affect and the Game Engine

THESIS

submitted in partial satisfaction of the requirements
for the degree of

MASTER OF SCIENCES

in Informatics

by

Cass Zegura

Thesis Committee:
Associate Professor Bo Ruberg, Chair
Associate Professor Aaron Trammell
Professor Tom Boellstorff

2022

DEDICATION

To my maternal grandparents, Betsy and Robert Whitte.

Grandma Betsy— When I first started going by Cass around my family, the last thing I expected was that the extended family would be so quick to catch up. There aren't words for the amount of comfort and warmth that your efforts to understand me have brought to my life. Even if you don't fully get it, or, indeed, this thesis, I will forever be inspired by your ceaseless and sincere curiosity. I hope to embody your tireless inquisitiveness, relentless desire to help out, and general optimism and kindness every day of my life.

Poppop— It's not MIT, but I hope an M.S. from UCI still does you proud.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to thank my parents and my sister, who have for the past two weeks been unwittingly sending me pictures of their fun, international vacations while I have been drowning in revisions. The jealous was certainly a good motivator to put my nose to the grindstone and keep working, and the pictures of your smiling faces helped me feel a lot less lonely during this last leg in California.

I want to thank my friends and the various forms of support I've gotten from them other last two years, from the 2020 Informatics cohort to my fellow CATS lab members to my friends outside of the university, including King House, the D&D groups who have tolerated my cancellations for the past month, and the Tumblr mutuals who put up with me liveblogging this project. Every single like on an expletive-laden rant or a caffeine-addled attempt at a joke has meant the world to me.

A special thank you to Kathryn Scholl and Mary Woodard for the boundless enthusiasm that you've shown for my research. I'm blessed to have such enthusiastic, curious, and caring friends.

I would also like to thank my committee members. I am endlessly grateful for the time you have all taken to review this work and to support my scholarship in general.

Aaron, you've been such a kind and soothing presence since Day 1, and I've always cherished our interactions. Your thoughtful and considerate construction of the CATS meetings was a regular source of comfort, and I've soaked up a lot of second-hand wisdom from listening to you give us advice.

Tom, your digital anthropology class was my favorite class I took at UCI, and you ran the most effective hybrid Zoom sessions I've ever seen. I really appreciated being able to talk with you and learn more about queer anthropology, and I'm sad I won't be here when you next offer the course.

Bo, "No Fun" is the reason I'm doing queer games work and you're the reason I'm at UCI. Working with you over the last two years has been an unparalleled honor, and I will never not be immensely grateful for your presence as a mentor. You are always helping me refine my thoughts to get at what really interests and excites me, and this thesis would be a mess of other people's ideas if you were not constantly pushing me toward my own voice. Thank you for everything.

Finally, thank you Ryro. Thank you for the beach.

ABSTRACT OF THE THESIS

Structuring Feeling, Feeling Structure: Affect and the Game Engine

by

Cass Zegura

Master of Sciences in Informatics

University of California, Irvine, 2022

Professor Bo Ruberg, Chair

Critical engine studies is a burgeoning subfield of games studies that takes as its point of departure not games themselves but the tools used to create them. Scholars in this field take seriously the influence that game development software, also known as game engines, have not only on the development process but on games culture and beyond. Far from viewing them as abstract tools or apolitical pieces of software, critical engine studies insists upon understanding engines as sociocultural objects as much as technical ones. This work requires, and the field calls for, new conceptual frameworks and analytical methods to advance our understanding of engines and the work that they do.

In this thesis, I consider affect as one approach to developing a more nuanced perspective of game engines. Affect is, I propose, well-equipped to refuse the abstraction that critical engine scholars problematize by not merely attending to game engine use but to the game engine user—and doing so with radical specificity. I consider three approaches to affect across the social sciences and the humanities in order to think broadly about what its multiple modalities can offer critical engine studies. First, I look at how affective game engines like GAMYGDALA strive to reproduce affect in the games that they are used to make. By providing prebuilt code structures

that developers can then deploy in their own games, these engines formalize but also flatten emotional experience in order to render it machine-legible. Second, I turn to a more expansive definition of affect to consider how engines structure feeling for users themselves. By looking at objects such as Become a Great Artist in Just 10 Seconds, I argue that game engines are always already affective, although some are felt more intensely and more deliberately than others. Finally, I consider how feeling could itself act as a mode of inquiry for critical engine studies. Drawing from queer theory and personal experience feeling different game development software, I argue that we can use affect to advance our understanding of game engines and how they act upon the user. Even when a game engine's structure cannot be seen, it can still be felt, and it is through such feeling that we can come to characterize engines beyond a binary discourse of freedom versus control.

INTRODUCTION

> *I view the process of creation as a collaboration between you and your medium [...] You have something you want in the creative process, but also your tools have something they want. You're trying to negotiate how much of what you want you can get and they're trying to negotiate how much of what they want they can get.*[1]

What do our tools want? This is the question at the heart of this quote from queer game maker Andi McClure. McClure is responding here to a question about how computation informs her creative practice, likening computers to guitars insofar as they both encourage some forms of creative production and limit others. They both make it easy to create certain things and harder, if not impossible, to create others, and this is, McClure points out, true for any tool. Take, as a more concrete example, my own instrument, the viola. The things that the viola makes are sounds, like McClure's guitar, but not all sounds are made equally. Compared to other instruments such as the guitar or piano, the viola has difficulty with harmonic chords— that is, chords where all notes are played at once. The viola prefers single note sounds because it is bowed and has a curved bridge which raises the strings off the fingerboard at different angles. You can play harmonic intervals by playing two adjacent strings at the same time. However, the angle between the strings is so great that bowing three (or four) strings simultaneously is outright impossible. Instead, to play chords, violists have to "roll" the notes, or play them successively from the lowest note to the highest. The viola, in short, makes some things (harmonic intervals) difficult and prohibits others (harmonic chords) entirely. It does not want these things.

1. Andi McClure, "Algorithms, Accidents, and the Queerness of Abstraction," interview by Bo Ruberg, *The Queer Games Avant-Garde: How LGBTQ Game Makers Are Reimagining the Medium of Video Games* (Duke University Press, 2020), 75.

Want, then, may function as a kind of limit, but this limit is as much generative as it is restrictive, foreclosing certain creative possibilities but enabling—even promoting—others. For example, violas want to play in C and G major/minor because these keys favor playing the thick, resonant lower strings open (without any fingers on them). An open string vibrates even after you have stopped playing it, and the lower the string, the longer the vibrations last. Subsequently, when the lowest note in a chord is an open C or G, that note will continue to sound even as the player rolls to the higher notes. This gives the aural impression of the chord's notes being played simultaneously, which sounds better (at least to a classical Western ear) than the notes being played separately. This is what the viola wants.

These are not, of course, novel observations. The basic concept that objects are designed to facilitate certain interactions and not others is one that is well-established within the field of Human-Computer Interaction (HCI). Don Norman's landmark *The Psychology of Everyday Things* (1988), also known as *The Design of Everyday Things*, established at-length the psychology behind our interaction with objects, the implications this has for designing them, and the nomenclature for discussing this phenomenon.[2] Norman established terms from affordance (what actions are possible?) to signifiers (how are these possibilities conveyed?) that have over the decades become vocabulary staples in the designing of hardware and software.[3] It may seem, then, that McClure is simply using different language to describe the same, well-established phenomenon. Norman says, for example, that a constraint "prohibits some activities and

2. Donald A. Norman, *The Design of Everyday Things*, Revised and expanded edition (New York: Basic Books, A Member of the Perseus Books Group, 2013).

3. Norman, *The Design of Everyday Things*, 14.

encourages others."[4] This recalls conceptually and, if not word-for-word, then synonym-for-synonym, McClure's assertion that a tool "is going to have things it's well-suited for and things it fights back against."[5]

Yet it is precisely in the words used that the distinctions between these two projects, subtle as they may be, become evident. The first half of this introduction is dedicated to these distinctions. I discuss two primary takeaways from McClure's use of *want* as her specific framework for thinking about how our tools act in this world and are in turn acted upon. These takeaways form the theoretical basis for the rest of my thesis. At the core of this project, I aim to clarify and characterize this reciprocal relation between tools and their users, and the emotions that result thereof, as they manifest within a particular class of tool: the game engine (discussed in the second half of the introduction). For now, I focus on McClure's brief but conceptually rich discussion of want as the theoretical hinge by which to open up my own conversation. Want lays the groundwork for considering tools not as designed objects that passively receive the desires and intentions of their users but as affective, agentive objects that actively participate in structuring our emotional experiences.

Acting and Being Acted Upon

To discuss our tools as wanting things, in and of themselves, rather than having been designed to afford things, is to conceptualize our tools as affective and agentive. This is inherent in the grammar of wanting. What I mean is that to want is a transitive verb: by its very linguistic

4. Donald A. Norman, "Affordance, Conventions, and Design," *Interactions* 6, no. 3 (May 1, 1999): 41, https://doi.org/10.1145/301153.301168.

5. McClure, "Algorithms, Accidents, and the Queerness of Abstraction," 75.

nature, to want ascribes agency, or a capacity for action, to its subject, and that action is directed toward an object. Our tools want, and they want actively, and they want *something*. It is this wanting something that suggests affect because emotion determines how we want an object. This is clearest in the case of wanting some*one*. In English, the words "I want you" often connote sexual desire; to want someone is to want them carnally, to want their body. But there are of course other ways to want someone, and they are just as affectively-charged: as a life partner, romantic or otherwise, as a friend, as a mentor and a source of support, as someone we take care of, as someone who takes care of us. To want someone is, in short, to have feelings for them, such as lust or love or camaraderie.

And tools want someones as well. They can, in fact, sometimes be quite exclusionary in the someones that they want. Take the viola and the guitar, which both want particular players with particular physical attributes that enable them to better handle the instrument. In both cases, this means someone with two, five-fingered hands, the muscular strength to hold the instrument, and the dexterity to move their fingers over it quickly and precisely. This restricts the instrument's desired someones to people of a certain ability, though workarounds are possible. A dear friend in undergrad who was missing their left forearm would strum their ukulele by taping a pick to what would have been their elbow. They had to negotiate their wants with the wants of the guitar, making a minor alteration to their own materiality in order to accommodate its desires. In doing so, their initial frustration with the instrument transformed into joy.

We can understand both emotions, different as they may be, as results of an encounter with an object and its wants. This is what I mean when I say that tools are affective. My intention in discussing want is not to anthropomorphize our tools or suggest that they want (us) with all the passion and intensity that that verb can imply. Rather, my point is that a tool's feelings,

whatever they may be, inform *our* feelings while using them. A tool's wants have implications for users as affective agents ourselves. These implications are what my thesis is dedicated to: how tools structure our feelings and what we might learn about the tools themselves through such feelings, particularly when these feelings may be misaligned. What is the value in attending to moments when our wants and the wants of our tools diverge? When one of us wants *differently*?

Desire and Difference

This brings me to McClure's second contribution, queerness. This above all else is why I have elected to begin with Andi McClure: because her work is steeped in it. Here, as in many contexts, including my own research, queerness has at least two meanings, and rather than distinguishing between them, it is useful to think that it refers to both at once. On the one hand, queer is a label encapsulating non-heterosexual sexualities and non-cisgender genders, and it is one that McClure identifies with on both bases. On the other, queer names a broadly antinormative mode of being and doing. This second meaning coalesced in the second half of the twentieth century across politics and academia, as activists grew more radical in their demands for queer liberation and scholars began to contest the treatment of sexuality and gender as stable, self-given, innate, and natural. Both challenged the assumption that heterosexuality was the natural default, instead revealing it to be a constructed norm and destabilizing it in the process.[6] Over time, queerness has become a method unto itself, naming an antinormative approach to

6. Siobhan Somerville, "Queer," in *Keywords for American Cultural Studies, Second Edition*, ed. Glenn Hendler and Bruce Burgett (New York, UNITED STATES: New York University Press, 2014), 203–7.

analysis and critique used across academic disciplines, from film and television to history to nature studies to quantum physics to human-computer interaction.[7]

In both instances, however, queer is a project of difference. In the case of the identity, queer is about desiring differently, in so far as there are desires that have been and continued to be marginalized— desiring people of the same gender, desiring multiple genders at once, desiring to be another gender. In the case of the broader theoretical project, there is still a desire for difference, but it is a desire to be, do, and live differently than what is expected and required by a given system of norms. In the case of McClure, then, to call her a queer game maker is not only to signal an identification but an approach to game making. Indeed, McClure is part of the queer games avant-garde, a term derived from Bo Ruberg's book of the same name to refer to LGBTQ game makers who are also working outside of and against the norms of game design in a variety of forms.[8] For McClure, this manifests as a quest to destroy the medium of video games itself through games that are radically abstract and expressively algorithmic— queer, not in their

---

7. Alexander Doty, *Making Things Perfectly Queer: Interpreting Mass Culture* (Minneapolis: University of Minnesota Press, 1993).

Heather Love, *Feeling Backward: Loss and the Politics of Queer History* (Cambridge, Mass: Harvard University Press, 2007).

Joshua Russell, ed., *Queer Ecopedagogies: Explorations in Nature, Sexuality, and Education*, vol. 8, International Explorations in Outdoor and Environmental Education (Cham: Springer International Publishing AG, 2021).

Amrou Al-Kadhi, *Life as a Unicorn: A Journey from Shame to Pride and Everything In Between*, Hardback edition. (London: 4th Estate, 2020).

Katta Spiel et al., "Queer(Ing) HCI: Moving Forward in Theory and Practice," in *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI EA '19 (New York, NY, USA: Association for Computing Machinery, 2019), 1–4, https://doi.org/10.1145/3290607.3311750.

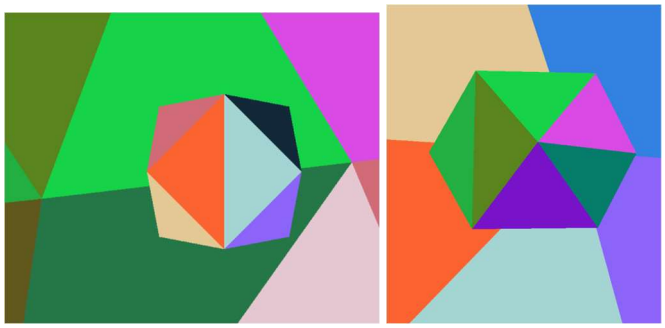
8. Bo Ruberg, *The Queer Games Avant-Garde: How LGBTQ Game Makers Are Reimagining the Medium of Video Games* (Durham: Duke University Press, 2020), 1.

representations of queer characters, but in their desire to imagine different possibilities for games.[9]

Take, for example, *Icosa* (2014). *Icosa* would not be considered a game by many definitions of the term, such as that given in Katie Salen and Eric Zimmerman's seminal game design text *Rules of Play* (2003): "A game is a system in which a player engages in an artificial conflict, defined by rules, that results in a quantifiable outcome."[10] *Icosa* has no conflict and little about it is quantifiable. Each time it is run, it generates a new game world consisting only of two mirrored icosahedrons, one that the player is in and one that the player is looking at (figs. 1 & 2). They can move in this world, but they cannot otherwise interact with it. Their only form of interaction, then, is through playing around with the settings that generate the world. They can, for example, pick the algorithm for shading each polygon face, but this is the extent of their control, and most of the settings are opaque and difficult to predict. Finally, the game does not provide any reward for mastery over its system. There are no outcomes or goals given to the player. Indeed, they must choose for themselves when the game is over. Personally, the end came either when I grew bored with the current configuration or when I accidentally created a headache-inducing psychedelic nightmarescape of constantly flickering colors. Perhaps not all games should end this way, but to make a game that does, even in all the unpleasantness that this may entail, is to do queer work because it defies the norms of what games should be.

9. McClure, "Algorithms, Accidents, and the Queerness of Abstraction," 78-79.

10. Katie Salen Tekinbaş and Eric Zimmerman, *Rules of Play: Game Design Fundamentals* (Cambridge, Mass: MIT Press, 2003), 80.



Figures 1 & 2. Screenshots from *Icosa* showing both the inner and the outer icosahedrons and the relationships between their colors. The background colors in figure 1 are the foreground colors in figure 2, and vice versa. Taken by author.

McClure offers queerness as the theoretical underpinning for conceptualizing the wants of our tools. By this I mean that, as queerness acts as the intellectual, creative, and material context for McClure's life and her work, so too must we understand it as part of how she frames her relationship to her tools. Desire is not innately queer, but the desires at the heart of McClure's games and her game making process are. So, in addition to opening up the possibility that our tools want things, and that this makes them affective and agentive, my takeaway from McClure is that it is also possible for tools to want *queerly*. In the case of games, as we saw with *Icosa*, this means to want more than what is prescribed for games by design textbooks. For tools more generally, this might look like rejecting design norms that demand beauty and utility, instead embracing ugliness, mess, and nonfunctionality.[11] It might also be speculative work, imagining beyond the bounds of what is currently possible and thinkable, toward other, more ethical constructions.[12]

11. Don Norman, "Emotion & Design: Attractive Things Work Better," *Interactions* 9, no. 4 (July 1, 2002): 36–42, https://doi.org/10.1145/543434.543435.

12. Fiona Barnett et al., "QueerOS: A User's Manual," in *Debates in the Digital Humanities 2016*, ed. Matthew K. Gold and Lauren F. Klein (University of Minnesota Press, 2016), 50–59, https://doi.org/10.5749/j.ctt1cn6thb.8.

Whatever their form, these are the works that animate me as a scholar. The potentials for queerness beyond identity, as a mode of being and doing, are of immense and immediate personal relevance. My work, like McClure's, is steeped in queerness, is inextricable from it. It informs everything I do inside and outside of academia. Even my research that is not specifically about queerness is colored by my background in queer theory; it is my default mode of critique and analysis and my intellectual go-to for relating to and understanding new scholarship. This is especially true in games studies, since my entry into the field was an undergraduate class called Queerness and Games. My primary understanding of games—as entertainment, as art, as industry, as cultural objects, as a general object of academic inquiry—is irrevocably colored by queerness. I prioritize queer games and their makers as the objects and subjects of my research, and I draw primarily from queer games studies for my theories and methods. Which is also to say that while this piece of games studies work is not primarily about queerness, queerness will nonetheless be present. It is my frame of reference for understanding wanting, desiring, and feeling; it is my baseline for considering my actual object of inquiry, affect.

It is, in a word (though a strange word it may be), my game engine.

The Heart of the Matter

Here, I pivot to the other half of this work's intellectual grounding. In what follows, I will discuss what game engines are and why myself and other scholars in the burgeoning field of critical engine studies want to research them. I will advocate for attending to what I term the affective capacity of game engines— that is, their ability to affect us and be affected by us in turn, and the emotions that result from this reciprocity. I argue that this will help bolster and further the work of critical engine studies, which calls for and requires new conceptual

frameworks and analytical methods to advance our understanding of these overlooked tools as more than mere abstract technology. Affect is the lens I offer for doing this.

Like McClure, I am not principally concerned with just any tool but specifically those used in video game development. More precisely still, of all the many pieces of software that can be used at any point in the game design process, from 3D modeling programs like Blender to composition software like GarageBand to special effects platforms like Houdini, I am interested in a class of software called the *game engine*. We can understand engines as the platforms that video games are both built *in*—as in engines provide a development environment where users can create, edit, and test their projects—and *on*— as in engines provide a baseline suite of functionality that users can adopt and repurpose to suit their games. Both of these configurations help explain the popularity of engines in the game development process because they both make it demonstrably easier. The development environment introduces a level of abstraction between a game maker's work and the internal code representation of that work. This is what allows, for example, developers to drag and drop objects into the game world, rather than having to write a line of code to spawn them. The baseline suite of functionality also makes life easier since developers do not have to spend time implementing general purpose code and can instead focus on what is unique to their game. As a simple example, the mathematical formula for determining if two shapes are intersecting—known as collision detection—is straightforward and universal, so it would be redundant for every single game developer to implement it in their game. Rather, they can just use their engine's existing collision detection algorithm and prioritize what should happen *after* a collision has been detected, such as collecting a power-up in a platformer or executing an attack in a fighting game.

While engines are by no means necessary for game production (all video games before the 1990s were made without them, and even some modern games, such as *Stardew Valley* [Eric Barone, 2016], do not make use of them), they are nearly ubiquitous. From the in-house, proprietary engines of AAA studios like EA's Frostbite and Bethesda's Creation Engine to the indie powerhouses Unity and Unreal to popular 2D engines such as RPGMaker and GameMaker, game engines are part and parcel with the game development pipeline. Depending on what you consider to be a game, the term also includes substantially smaller pieces of software. Niche platforms associated with hobbyist and personal game making include Twine, Bitsy, and Ren'Py. Like *Icosa*, the works produced on this platform do not always count as games, being light on rules and mechanics and sometimes lacking conflict entirely. In the spirit of queerness, however, in my work, I try to think about games differently and expansively. Thus, the definition I will use here is from *Rise of the Videogame Zinesters* (2012), by queer game maker Anna Anthropy: "A game is an experience created by rules."[13] By extension, my definition of game engine is equally expansive, including all of the software discussed above and many more tools that can be used in the creation of experience.

This is, furthermore, in keeping with the values and goals of critical engine studies, the burgeoning field in which this project is located. Exemplified by recent works such as Benjamin Nicoll and Brendan Keogh's *The Unity Game Engine and the Circuits of Cultural Software* (2019) and Eric Freedman's *The Persistence of Code in Game Engine Culture* (2020), this subfield of games studies takes as its object of inquiry not games themselves but the tools used to develop them. At present, critical engine studies is most notably defined by its analytical

13. Anna Anthropy, *Rise of the Videogame Zinesters: How Freaks, Normals, Amateurs, Artists, Dreamers, Drop-Outs, Queers, Housewives, and People Like You Are Taking Back an Art Form* (New York: Seven Stories Press, 2012), 43.

orientation toward game engines. Its few but varied works are united by a belief that game engines are not neutral, apolitical tools but are culturally- and socially-situated pieces of software— and that game production, whether AAA or indie or hobbyist, is only one such situation. Others include the scandal around Unity's military contracts or Hollywood's use of the Unreal engine in film and TV production.[14] For critical engine scholars, game engines are not tightly-bounded software but instead form ecosystems with far-reaching influences and sociocultural and political entanglements. But engines also matter within radically specific and singular instances of game production. If games studies is concerned with games as final products, then it is necessarily (if not always significantly) also concerned with game engines because engines are fundamental to the context in which games come to be.

When I said that queerness is my game engine, this is what I meant: queerness undergirds my work the way that the engine undergirds the games it is used to make. Even when my work is not explicitly about queerness, it lurks underneath the explicit, just as the engine code lurks beneath the game code that is built on top of it. Both inform and influence what they are used to construct, whether that is an intellectual argument or a video game. Despite at times disappearing beneath that construction, queerness and the game engine are always there, are always foundational.

---

14. Patrick Klepek, "Unity Workers Question Company Ethics As It Expands From Video Games to War," *Vice*, August 23, 2021, https://www.vice.com/en/article/y3d4jy/unity-workers-question-company-ethics-as-it-expands-from-video-games-to-war.

Alison Harvey, "Twine's Revolution: Democratization, Depoliticization, and the Queering of Game Design," *G.A.M.E. (Reggio Calabria)*, 2014.

"Film & Television | Unreal Engine - Unreal Engine," Unreal, accessed May 5, 2022, https://www.unrealengine.com/en-US/solutions/film-television.

Nowhere is this fundamentality clearer, at least for game engines, or more curiously stated than in the words of one of the men who first designed them. This technology was initially developed by game design pioneers John Carmack and John Romero. Within the annals of gaming history, the "two Johns" are best known for the technologically revolutionary and wildly popular franchises that they established while working at id Software in the 90s, including *Wolfenstein 3D* (1992), *Doom* (1993) and *Quake* (1996).[15] One of these technological revolutions was the game engine, a term that Carmack and Romero first used to refer to the code shared across all entries in the *Commander Keen* (1990, 1991) franchise.[16] While it would take years for engines to become the more general-purpose development environments that they are today, the Keen engine (and the subsequent Wolfenstein, Doom, and Quake engines) still served a similar function by separating engine code from game-specific code and assets.

But why call this shared codebase a game engine? The true oddness of the term did not strike me personally until I once had to describe my research in Spanish and translated it as "motor de juego." The phrase "game motor" draws sharper attention to the mechanical origins of the word engine. While it has acquired digital connotations over time, such as search engine and browser engine, Carmack and Romero were labeling their codebase before the popularization of the World Wide Web, when such associations were acquired. In fact, we know from Romero that they were inspired exclusively by the mechanical meaning of the word. The term game engine was derived from Carmack and Romero's shared hobbyist enthusiasm for automobiles, with

15. David Kushner, *Masters of Doom: How Two Guys Created an Empire and Transformed Pop Culture* (Westminster: Random House Publishing Group, 2003).

16. Henry Lowood, "Game Engines and Game History," in *History of Games International Conference Proceedings*, eds. Carl Therrien, Henry Lowood and Martin Picard (Kinephanos: January 2014), 184-85.

Romero explaining that "the engine 'is the heart of the car, this is the heart of the game; it's the thing that powers it.'"[17]

The engine is the heart of the game.

Andi McClure's opening quote is called back to mind here. Indeed, these two quotes seem almost natural extensions of one another. Of course our tools—by which I mean game engines—want things. They want things not because they have a heart with which to feel—to desire, to love, to hate—but because they *are* a heart. Game-making, then, is a heart-to-heart, as in your heart as a creator entangles with the engine as heart, and together, your hearts must negotiate a complex series of related and competing desires in order to produce a game. In order to create, you must touch the game's heart, and, by the transitive property, you must be touched by it in return. Game production is an intimate pipeline indeed.

Affect in Critical Engine Studies

This thesis seeks to explore these intimacies and the emotions that they elicit. My contribution to critical engine studies is an attention to the affective capacities of our tools. By way of closing out my introduction, then, I want to frame my project in contrast to how feeling has entered into critical engine studies thus far. My argument is not that critical engine studies has necessarily neglected or ignored emotion. Rather, affect is present beneath the surface, as it were, of key works in critical engine studies, and without surfacing them, certain nuances go unaccounted for.

To explain this gap, I want to consider how critical engine studies has discussed the engine Twine and its alleged revolution. Twine, a hypertext authoring platform first released in 2009, is a fascinating object for critical engine studies because it inspired something of a

17. Romero, quoted in Lowood, "Game Engines and Game History," 186.

precursor to the field: a critical discourse within games popular culture that took seriously the political and cultural valences of game making tools.[18] Both this discourse and Twine's popularity originated in 2012 from the nascent queer games scene. Defining figures from this period such as Anna Anthopy, Porpentine Charity Heartscape, and merritt kopas took up Twine and began to make queer games with it. These games were again queer in multiple senses: they often represented the personal, lived queer experiences of their creators through their characters and narrative, but they also presented a profound challenge to the norms of game design. Anthropy's *Queers in love at the end of the world* (2012), for example, is about a queer couple infinitely looping through the last ten seconds of the apocalypse before the world is erased forever. This is a game without winning, losing, or even ending, since even as the world does always end, the game always loops back. The timer always resets.

The challenge that Twine offered was more fundamental than that, though. Through the tool's technological and financial accessibility, these works articulated new possibilities for game making. Game making does not have to be specialized, expensive, or hard, and it does not have to require extensive coding experience or complicated graphics and mechanics. It could instead just look like writing and using links to connect the different parts of the writing (fig. 3). Twine, in short, challenged the status quo of game production by allowing game makers, particularly those without a coding background, to express themselves differently within the medium. This

---

18. Joe Bernardi, "Choose Your Own Adventure-Maker: Twine and the Art of Personal Games," *Vice* (blog), February 19, 2013, https://www.vice.com/en/article/xyyp9a/twine-and-the-art-of-personal-games.

Cara Ellison, "Anna Anthropy and the Twine Revolution," *The Guardian*, April 10, 2013, sec. Technology, https://www.theguardian.com/technology/gamesblog/2013/apr/10/anna-anthropy-twine-revolution.

Harvey, "Twine's Revolution."

challenge was termed the Twine revolution in a discourse that framed Twine as something that was going to allow *anyone*, especially oppressed people, to make games. It was viewed as the tool that was going to rock the very foundations of gaming culture and usher in a new era of mechanical creation and expression for queer and trans people.[19]

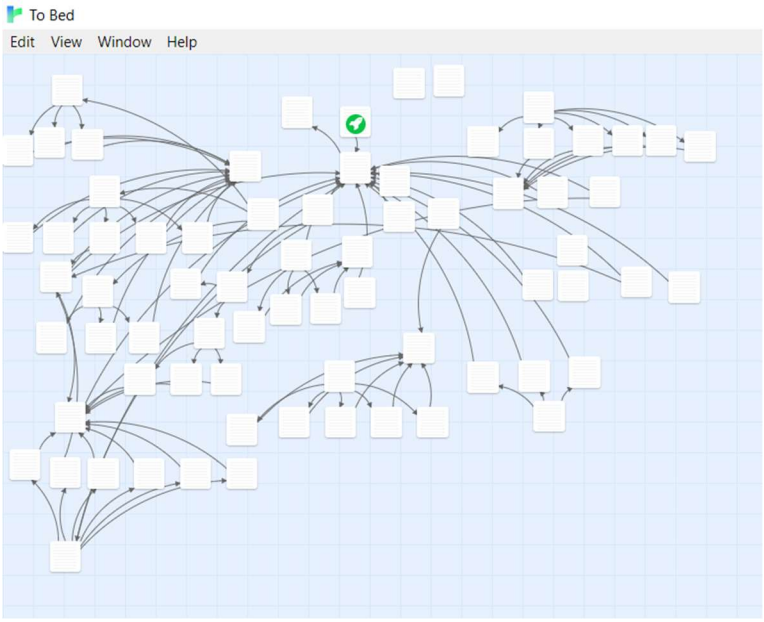


Figure 3. A screenshot of a personal Twine project, showing off the complex interconnections between passages.

Yet some critical engine scholars express skepticism around the queer, liberatory potential of Twine. Eric Freedman, for example, remarks:

> For queer game artists, the simpler mechanics (and alternative game languages) of Twine and GameMaker seem to facilitate qualities such as empathy, community and communicative openness. But we need to avoid understanding even the value of engine choice in such binary terms, as freedom, expressivity, multivocality, and queerness are in every case delimited by software mechanics.[20]

19. Harvey, "Twine's Revolution."

20. Eric Freedman, *The Persistence of Code in Game Engine Culture* (Taylor and Francis, 2020), 163.

Freedman's point here is not dissimilar to McClure's or Norman's: engines want things, and sometimes those wants impinge upon the wants of their creators. Some desires are so different from that of a given engine that they are irreconcilable. There are certain things that are impossible to do even with engines lauded for their freedom, such as Twine and GameMaker. In the specific case of Twine, Freedman is right to be skeptical. Its revolution was never fully realized. While our collective cultural understanding of a game might be more inclusive now, industry design norms were not substantially changed by the time the revolution fizzled out in 2014, and they remain largely unchanged now, with exploitative labor practices like "crunch time" continuing to define how mainstream games are made.[21] Meanwhile, queer game makers remain in positions of social and economic precarity.[22] They have not reaped the rewards of their revolution, such as greater cultural legitimacy or equitable financial compensation for their work.

Still, this retrospection overlooks the specific entanglements and intimacies that Twine had with specific creators which led to it being imbued with revolutionary, queer potential. It glosses over a key, affectively-charged moment in games history in which Twine was deliberately chosen and taken up by early queer game makers. Freedman reduces this choice to one of convenience, but attending to the feelings that queer game makers had toward Twine reveals a much more strategic approach.

In illustrating this, I want to highly a pair of quotes from queer trans game designer Porpentine. The first is from her 2012 article "Creation Under Capitalism and the Twine

---

21. Jason Schreier, "Crunch Time: Why Game Developers Work Such Insane Hours," Kotaku Australia, May 16, 2015, https://www.kotaku.com.au/2015/05/crunch-time-why-game-developers-work-such-insane-hours/.

22. Bo Ruberg, "The Precarious Labor of Queer Indie Game-Making: Who Benefits from Making Video Games 'Better'?," ed. Greig de Peuter and Chris J. Young, *Television & New Media* 20, no. 8 (2019): 778–88, https://doi.org/10.1177/1527476419851090.

Revolution," which acts as both a fiery game design manifesto and an instructional guide. The second is a tweet cited in an essay by games scholar Alison Harvey called "Twine's revolution: Democratization, depoliticization, and the queering of game design" (2014). Both of these quotes capture Porpentine's passionate feelings toward Twine and its capacity, if not for total liberation, then certainly for imagining different ways of doing and being:

> Our engines shape our output. [...] The history of game design has been working with a canvas that screams at you and changes shape and rejects your strokes if they aren't just right– working with machines.
> Not all of us want to become machines, some of us just want to have frank discussions with the machines.[23]

> twine succeeded precisely because of its violence--because it was suited for guerilla warfare--a cheap, disposable weapon of underdogs[24]

In the first quote, we see Porpentine's frustration with existing engines and their hostility toward her— the reciprocal though uneven capacity to act and be acted upon as Porpentine struggles to meaningfully affect her tool without being yelled at. This is not a matter of wants being misaligned, but an intolerance on the part of the engine for any want that is not compatible with its. But Twine is configured as being open to discussion, to negotiation. It does not want to scream; it does not feel contempt for its users.

And yet the second quote asserts that it is still capable of violence, and a specific kind of violence at that: the dirty, underhanded tactics of underdogs who know that they cannot win otherwise. The violence of revolutionaries. Yet at the same time that this quote suggests a

---

23. Porpentine, "Creation Under Capitalism and the Twine Revolution," Nightmare Mode [Archived], November 25, 2012, https://nightmaremode.thegamerstrust.com/2012/11/25/creation-under-capitalism/.

24. Porpentine (@aliendovecote), "twine succeeded precisely because of its violence--because it was suited for guerilla warfare--a cheap, disposable weapon of underdogs," Twitter, May 18, 2013, cited in Alison Harvey, "Twine's Revolution: Democratization, Depoliticization, and the Queering of Game Design," *G.A.M.E. (Reggio Calabria)*, 2014.

struggle—the endless struggle of queer people fighting for their lives in a homophobic and transphobic society—it also suggests the hope that this struggle will succeed… and that Twine will contribute to its success. Porpentine values Twine not for the values that Freedman mentions, such as empathy and communicative openness, but as a weapon trained with destructive intent against the norms of game design and even (as the title "Creation Under Capitalism and the Twine Revolution" implies) against the capitalist heteropatriarchy itself.

Twine's capacity to evoke affect in its users includes all of this and more, thousands, perhaps even millions, of additional, specific sites of entanglement and interaction which have resulted in users (queer or otherwise) feeling something through and toward Twine. This capacity teems in Freedman's original diagnosis of Twine, particularly the line: "freedom, expressivity, multivocality, and queerness are in every case delimited by software mechanics."[25] But without fully attending to that affect and its history, it becomes flattened. Twine's specific intimacies and the feelings that result from it are overlooked in favor of asserting that it is, like any game engine, like any tool, constraining.

Of course Twine is constraining. All tools are. If critical engine studies seeks to characterize game engines as more than abstract technical objects, then we will also need to consider more than constraint. We will have to reckon with engines as affective and agentive, as desiring and wanting and being potentially queer in doing so. Drawing affect to the fore allows us to consider in greater detail how game engines act upon—affect—us.

25. Freedman, *The Persistence of Code in Game Engine Culture*, 163.

Looking Ahead

But what exactly am I bringing to the fore? What actually is affect? My definition of the word has thus far been vague, relying on treating it as synonymous with emotion and feeling, as well as the common-sense, collectively intuited meaning of those words. Despite the centrality of affect to my project, however, this vagueness has been intentional. Like queer, affect contains multiple definitions, particularly across disciplinary bounds. Each meaning has unique implications for the game engine. Each configures my project of theorizing them together in unique ways, drawing my attention to some intimacies and feelings while deemphasizing others. Different definitions of affect prioritize different engine wants. Rather than picking one, then, part of my project is to consider multiple definitions and how they each open up valuable possibilities for considering game engines not as neutral, apolitical technologies but as affective agents embedded in the sociocultural weave of our world.

Subsequently, this thesis is divided into two chapters. Each considers a different phenomenon of the game engine that affect helps to illuminate, and each is organized around a particular disciplinary theory of affect: that of Human-Computer Interaction in Chapter 1, and that of the humanities in Chapter 2. Chapter 1 explores how game engines structure the feelings of others— that is, how they provide the scaffolding by which emotions are felt by users, players, and even non-player characters. I focus on two structures in particular, code and experience, and I pair these structures with two modalities of affect within HCI, affect as information and affect as interaction. In comparing and contrasting these approaches and their implications for the game engine, I argue that certain ontologies of affect and the design methods that they produce risk narrowing and overdetermining the affective capacities of game engines. The alternative may be the creation of highly experiential, deeply felt game engines that do not

want to prescribe a singular, specific emotional response but rather permit and help constitute a broad spectrum of reactions.

In Chapter 2, I consider what these reactions might reveal about the game engine itself. I propose that engines are structures that can themselves be felt by users. This feeling, both as a metaphorical physical sensation and as an emotion, can tell us something about how game engines are structured. In much the same way that we can understand the shape of an object even with our eyes closed by turning it around in our hands, we can understand something about the engine—about its wants and desires, about its limits, constraints, and contours, about its capacity to act upon the user—through feeling. Here, I make use of what I call affect as relation. I derive this definition of affect from the humanities. This definition emphasizes that affect is neither fully the purview of one object nor another but instead arises in between them. That is, affect is neither projected onto a game engine by its user nor given, ready-made, to its user by the game engine. Instead, it results from their entanglements with one another. This perspective of affect demands radical specificity, an attention to singular engine-user encounters like the encounters between Porpentine and Twine laid out previously. Ultimately, I argue that by attending to such encounters and the feelings that result, we can better characterize the engine's relationship to the user. We can move beyond the binarism of engine choice that Freedman criticizes and toward a pluralistic—if always partial—understanding of how engines affect us and our work.

CHAPTER 1

Under the Hood: How Game Engines Structure Feeling

> *I can work with Twine when I'm too tired to deal with anything else. You don't have to wrestle with anything between the emotion and the page, your fragile thoughts survive.*[1]

How do our tools structure our feelings? In what ways are our emotions influenced by them, informed by them, given meaning by and through them? This quote from queer game maker Porpentine Charity Heartscape articulates a profound connection between her own emotions and her experience using the game engine Twine, a niche tool for authoring hypertext fiction and games. Porpentine argues here that other tools structure her feelings *too* much, imposing barriers between her emotions and her creative output. While these barriers can be circumnavigated, this is taxing work, especially for an already exhausted queer trans woman or for any other marginalized game maker fighting for their own survival. When it is hard enough to preserve ourselves, it is harder still to preserve those ideas that are, like us, vulnerable. Our tools can overwhelm these fragile feelings by imposing their own affective structures on them. Yet these same structures can also be protective and enabling, a site where these feelings might be able to be expressed and, in so doing, become stronger and better understood. For Porpentine, this is the case for Twine.

In summary, I identify two possibilities for how game engines can intentionally structure feeling, not only for their users but for their users' work: as a site for emotional determinism or as a site for emotional meaning-making. I emphasize intentionality in characterizing these two

---

1. Porpentine, "Interview with Porpentine, author of howling dogs," interview by Emily Short, *Emily Short's Interactive Storytelling* (blog), November 23, 2012, https://emshort.blog/2012/11/23/interview-with-porpentine-author-of-howling-dogs/. Archived at https://web.archive.org/web/20170312075143/https://emshort.blog/2012/11/23/interview-with-porpentine-author-of-howling-dogs/, accessed May 7, 2022.

approaches because there are plenty of game engines—arguably, the majority of them—which are not interested in explicitly structuring feeling. Even if such influences inevitably result from the inherent affective capacity of all tools, they are not my priority here and will instead be taken up in Chapter 2.

In this chapter, then, I explore these two possibilities at both a conceptual and practical level. For the conceptual work, I pair each with a definition of affect that suits their respective goals and ontologies: affect as information and affect as interaction. Both approaches are derived from the larger project within Human-Computer Interaction (HCI) of *affective computing*, but they have distinct and important implications for game engines as well. Indeed, in the case of affect as information, this has already manifested in the speculative and actual (if limited) design of so-called affective game engines. This points to the more practical component of my work, as I will supplement this theoretical consideration with specific game engines that intentionally structure affect in accordance with these two definitions. My approach to these engines will also be in accordance with these two definitions, focusing on code in the case of affect as information and emotional determinism, and experience in the case of affect as interaction and emotional meaning-making. I ultimately argue that game engines can be strongly felt and emotionally impactful without prescribing what form that feeling and emotion takes.

Affect as Information

Affect as information, a term first proposed by Boehner et al. in their paper "How emotion is made and measured," names an approach to studying emotion that focuses on feelings as

discrete, stable, quantifiable, and objective states.[2] Rather than relying on the comparatively subjective and messy process of self-reporting emotions, scholars who subscribe to affect as information prefer to measure physical signals and determine from the quantified data that they collect what someone is feeling. In short, emotions are viewed as facts that can be measured objectively. Such a perspective is valuable for certain modes of scholarly inquiry. Researchers experimenting with emotion in a laboratory setting, for example, might be interested in capturing participants' feelings "in the moment" rather than after participants have had time to reflect on and make subjective sense of their own experiences.

Indeed, the problem of studying emotions in a laboratory setting is essential to understanding affect as information, insofar as we should not take it as a neutral, self-evident approach but rather the product of a specific knowledge project. This becomes evident by turning to its history. In "Counting the Affects: Discoursing in Numbers," Otniel Dror lays out in detail how, up until the late 1800s, emotion was previously excluded from the sciences because it was viewed as incompatible with rational inquiry.[3] Emotion was viewed as subjective, messy, and feminized, and it threatened objective, orderly masculine study. Thus, for emotion to become a valid object of scientific inquiry, it first had to be reconceptualized. It had to be contained so that it could be studied without jeopardizing the objective integrity of the larger project. Measurement, Dror argues, was the solution, the making of emotion into a number: "[i]t created a 'hard' variant of a 'soft' object of knowledge."[4] In practice, this has involved thousands of

2. Kirsten Boehner et al., "How Emotion Is Made and Measured," *International Journal of Human-Computer Studies* 65, no. 4 (2007): 275–91, https://doi.org/10.1016/j.ijhcs.2006.11.016.

3. Otniel E. Dror, "Counting the Affects: Discoursing in Numbers," *Social Research* 68, no. 2 (2001): 357–78.

4. Dror, "Counting the Affects," 370.

studies in psychology and beyond that measure observable, physical, and quantifiable phenomena, from facial expressions to vocal intonations to physiological signals like heart rate and galvanic skin response. The numerical data collected from these signals is then mapped to discrete emotional states. Researchers might, for example, measure startle as a function of changes in skin conductivity, stress as a function of variation in vocal pitch, or depression as a matter of electrodermal activity and sleep patterns.[5]

And not all of this work is being done inside a psych lab. Affect as information has spread to other fields as well, including Human-Computer Interaction in the form of affective computing. Affective computing is a unique project, first conceptualized in 1997 by MIT researcher Rosalind Picard, that considers not merely how emotion can be measured, but how *computers* can measure it and respond accordingly. As Picard put it, her vision was to "give computers the ability to recognize, express, and in some cases, 'have' emotions."[6] She speculates on a broad number of applications for this technology, including an affectively-aware tutor software, emails with dynamic and accurate indications of tone, and virtual or robotic pets.[7] In practice, as the field has developed and expanded in the decades since Picard's proposal,

---

5. Jennifer Healey and Rosalind. Picard, "StartleCam: A Cybernetic Wearable Camera," in *Digest of Papers. Second International Symposium on Wearable Computers (Cat. No.98EX215)*, 1998, 42–49, https://doi.org/10.1109/ISWC.1998.729528.

Hong Lu et al., "StressSense: Detecting Stress in Unconstrained Acoustic Environments Using Smartphones," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12 (New York, NY, USA: Association for Computing Machinery, 2012), 351–60, https://doi.org/10.1145/2370216.2370270.

Asma Ghandeharioun et al., "Objective Assessment of Depressive Symptoms with Machine Learning and Wearable Sensors Data," in *2017 Seventh International Conference on Affective Computing and Intelligent Interaction (ACII)*, 2017, 325–32, https://doi.org/10.1109/ACII.2017.8273620.

6. Rosalind W. Picard, *Affective Computing* (Cambridge, Mass: MIT Press, 1997), 1.

7. Picard, 85-111.

research has often focused on using wearables and mobile phones in order to collect and interpret data, since these are unobtrusive computing technologies that could be more easily and more willingly used outside of a laboratory setting.[8] Even in lab experiments, affective computing scholars often design their data collection instruments with real usage in mind, as in a 2020 experiment that developed a wearable to measure gastric signals from the skin of the stomach. Researchers specifically designed their device to be low profile and easily concealable underneath someone's clothes and argued that it could have a host of practical applications, such as monitoring gastrointestinal disorders or helping users with eating disorders minimize their stress around eating.[9]

Affective Gaming

Of most immediate relevance to my own work, however, is how affective computing has influenced HCI scholarship on video games. It is within this realm that we can clearly and practically see not only how emotion is made information but how computer systems might be made to dynamically respond to this information. The same physiological signals used to monitor stress for research into personal health wearables, for example, can be used to create games that are affectively-adaptive, meaning that they change according to the player's arousal levels. For example, researchers at the University of Saskatchewan have done experiments in which they have adjusted game difficulty according to changes in players' galvanic skin

8. Akane Sano and Rosalind W. Picard, "Stress Recognition Using Wearable Sensors and Mobile Phones" (IEEE, 2013), 671–76, https://doi.org/10.1109/ACII.2013.117.

9. Angela Vujic et al., "Going with Our Guts: Potentials of Wearable Electrogastrography (EGG) for Affect Detection," in *Proceedings of the 2020 International Conference on Multimodal Interaction* (New York, NY, USA: Association for Computing Machinery, 2020), 262, 266-67, https://doi.org/10.1145/3382507.3418882.

response. Using a modded version of *Half-Life 2* (Valve, 2004), they were able to change gameplay mechanics in real-time, such as making the player character faster or decreasing the number of enemies, in order to keep players from being either over-stimulated or bored.[10]

The game, in short, structures the feelings of its player: it changes itself to elicit a particular affective response, restructuring itself until galvanic skin response measurements fall within the desired parameters. Until the desired level of arousal is reached. This relentless insistence toward a particular affective state is indicative of what I named at the beginning as emotional determinism. By this I mean that the game has as its goal the production of a specific, predetermined affective response. This is at times a desirable approach to game design. As the paper on affectively adapting *Half-Life 2* points out, if a player is bored or overwhelmed by a first-person shooter, they may quit the game. Keeping players in the "sweet spot" between these two feelings might keep them engaged and playing longer.

However, affective gaming literature also tends to overstate its value for game design writ large, since not all developers want to determine their players' emotions for them. For example, a foundational paper in the field, Gilleade et al.'s "Affective Videogames and Modes of Affective Gaming," asserts that "if affect is not conveyed properly during game play [...] then the player's suspension of disbelief can be negatively affected and so spoil the gaming experience."[11] This presumes that a game has a proper, singularly identifiable affect that it wants to convey and that this affect should be prioritized at the expense of other emotional experiences

10. Faham Negini, Regan L. Mandryk, and Kevin G. Stanley, "Using Affective State to Adapt Characters, NPCs, and the Environment in a First-Person Shooter Game" (IEEE, 2014), 1–8, https://doi.org/10.1109/GEM.2014.7048094.

11. Kiel Mark Gilleade, Alan Dix, and Jen Allanson, "Affective Videogames and Modes of Affective Gaming: Assist Me, Challenge Me, Emote Me," in *Proceedings of DiGRA*, 2005, 1–2.

that players might inadvertently have. This level of determinism is not desirable for all game designers. From highly experimental, interpretive, and glitch-based works like JODI's *Untitled Game* (1996-2001) or Andi McClure's *Icosa* (2014), to works that are indifferent toward or even hostile to the player and their feelings, such as Mattie Brice's *EAT* (2013), the emotional determinism offered by affect as information is not the only nor even the best way to structure players' feelings.[12]

And players' feelings are not the only ones that are overly determined. Affective gaming opens up the possibility of applying affect as information to the emotions of game *characters* as well as players. This strand of work imagines creating games whose NPCs are emotionally and socially realistic— that is to say, NPCs who respond to events and other characters in the same way that a real human would. At the moment, to the extent that NPCs have emotions, it is primarily through their writing. In a conversation with the player character in a role-playing game, for example, if the player chooses a dialogue option that upsets the NPC, they will give a scripted response that is written to be (and often voice acted to be) upset. Depending on the sophistication of their graphical representation, their sprite might change to a preset sad variant, or their facial mesh might contort and make an upset expression. In any case, the emotion is superficial and prescripted, not a deeper, dynamic mechanic.

Work such as Yuda et al.'s "Creating an Affective Fighting Game AI System with Gamygdala" seeks to address this by creating internal, quantified representations of emotion and designing algorithms to modulate those emotions in response to events in the game. Yuda et al.'s fighting game, for example, stores an internal representation of the emotional state of the

12. Mattie Brice, "Death of the Player," *Mattie Brice* (blog), October 29, 2013, http://www.mattiebrice.com/death-of-the-player/.

player's opponent, changes that state based on actions in the game like being damaged by the PC, and then changes the NPC's behavior accordingly.[13] Thus, when the opponent is experiencing higher anger and distress, their combo efficiency is decreased, representing narratively a lack of focus and an inability to execute complex maneuvers. Similarly, when they are hopeful, satisfied, and relieved, the opponent takes more time to make decisions, indicating perhaps being lulled into complacency.[14]

Even as I describe these changes in narrative terms, however, it is essential to keep in mind that what is actually being discussed is code: emotions represented as discrete, numerical intensities and varied according to an algorithm that quantitatively appraises the world state. In an even more literal sense, then, here we see affect as information structuring feelings through code. Indeed, the structure and the feeling are one in the same, since there is no exterior phenomenon being measured. Emotion is not made into information as an epistemological product. Rather, information is made into emotion by programmers ascribing semantic labels to strictly numerical code. By this I mean that the game does not know that an NPC is, for example, fearful. Nor does the NPC know it themself. We identify this state as fear as a conceptual convenience, but it is detached from the plurality of contexts, relations, and secondary feelings that that word can articulate.[15] All the game does is store an intensity and update it regularly in accordance with an equally quantified algorithm.

---

13. Kaori Yuda, Maxim Mozgovoy, and Anna Danielewicz-Betz, "Creating an Affective Fighting Game AI System with Gamygdala" (IEEE, 2019), 1–4, https://doi.org/10.1109/CIG.2019.8848031.

14. Yuda, Mozgovoy, and Danielewicz-Betz, "Creating an Affective Fighting Game AI System with Gamygdala," 3.

15. See, for example, Sara Ahmed, "The Affective Politics of Fear," in *The Cultural Politics of Emotion*, 2nd edition (Edinburgh University Press, 2014), 62–81, https://doi.org/10.3366/j.ctt1g09x4q.

And that algorithm, in the case at least of this paper on developing a fighting game AI, comes from a game engine.

Affective Game Engine

I turn now to the project of the affective game engine in order to consider its role in producing emotional determinism: as a progenitor of the structures that game developers then use to narrowly produce affect in players and non-player characters alike. After giving an overview of the project, I consider the specific engine used to develop the fighting game AI discussed in the last section, GAMYGDALA. I closely read GAMYGDALA's code to illustrate how structuring feeling and emotional determinism operate at the base level. Finally, I critique this process of producing realism and reductionism in equal measure by turning to queer theory.

First proposed in 2009 by affective computing scholar Eva Hudlicka, the original affective game engine was at once a profoundly pragmatic and a deeply speculative turn in affective gaming research.[16] Hudlicka envisioned a piece of software that could bring affective gaming techniques beyond the laboratory. No longer solely the purview of researchers with the specific training necessary to realize affective gaming's techniques, the affective game engine would provide the baseline functionality for structuring affect to all game developers. This would allow them to make their games affective without worrying about implementing these techniques themselves. In this sense, then, the "affective" of affective game engine is not the same as the "affective" of affective computing or affective gaming. It is not designed to deliberately structure the feelings of its users. Rather, it is what provides a structure that developers can then deploy in

16. Eva Hudlicka, "Affective Game Engines: Motivation and Requirements," FDG '09 (ACM, 2009), 299–306, https://doi.org/10.1145/1536513.1536565.

their games in order to determine the affect of players and NPCs. To that end, Hudlicka proposed four functions that the affective game engine should provide developers: "[1] emotion recognition in the player, [2] emotion expression by game characters and player avatars, [3] representation of the player's emotions via dynamic affective user models, and [4] modeling of emotions within the game characters."[17]

Of course, Hudlicka also recognized that the actual state of affective computing technology could not realize this vision. Thus, the paper ends with a lengthy "Affective Game Engine Requirements" section. This details the strides in affective computing and gaming research and design that would need to be made to produce a fully realized affective game engine. This not-yet-ness of the affective game engine is among its more curious qualities, encompassing a looking forward, a desiring beyond what is at the moment possible, which recalls some of HCI's earliest works, such as Vannevar Bush's "As We May Think".[18] Indeed, the very project of affective gaming may be summed up as "as we may feel". How may we feel? Through the structures that the game engine could provide.

In the decade since Hudlicka's original proposal, the affective game engine has lost some of its speculative shine. Although no technology boasts of providing all four functionalities, numerous projects both within affective gaming research and beyond have tackled one or two of them. The *Half-Life 2* affective modifications discussed earlier, for example, made use of an Affective Middleware Engine to support recognizing the player's emotions. The AME mediated

17. Hudlicka, 303-04.

18. Vannevar Bush, "As We May Think," *The Atlantic Monthly* 176, no. 1 (July 1945): 101–8.

between the main game engine and the device used to measure GSR signal.[19] Pixel Crushers, an independent studio that develops assets for the game engine Unity, has created a plug-in to handle the modeling of game character's emotions and personalities, called Love/Hate.[20] The fighting game discussed early uses a similar system called GAMYGDALA, described by one of its creators, Dutch affective computing scholar Joost Broekens, as an emotion engine.[21]

These partial realizations of the affective game engine project make compelling objects for considering how affect as information works at a micro design level. This is because, where affective games are often designed exclusively for laboratory experiments, game engines are designed for general use by game makers. They are much more readily accessible than the products that they are used to make, which can often only be considered through the papers written about them. Even when an affective game is readily accessible, its inner workings, the precise mechanisms of its structures of feeling, are often obscured. As a practical matter, game engines are easier to get at. I turn now to look under the hood, as it were, of the GAMYGDALA engine. I have chosen GAMYGDALA because it is particularly readily accessible, being free to download from Broekens' personal site, and because he and Hudlicka have published papers on it that offer further insights into its development. In considering this game engine, I illustrate how affect as information has informed its design and operates at the level of code. Finally, I critique the reductionist structures of feeling that it offers through the lens of queer theory before moving on, in the second half of the chapter, to more expansive possibilities.

---

19. Negini et al., "Using Affective State to Adapt Characters, NPCs, and the Environment in a First-Person Shooter Game," 3.

20. https://assetstore.unity.com/packages/tools/ai/love-hate-33063

21. https://ii.tudelft.nl/~joostb/gamygdala/index.html

Coded Feelings: GAMYGDALA

GAMYGDALA is an emotional appraisal engine— that is, it provides base functionality for non-player characters to have emotions and for those emotions to change in response to the game world. It was developed in 2014 at the Delft University of Technology by researchers Joost Broekens, Alexandru Popescu, and Maarten van Someren. In conjunction with other affective gaming scholars, including Eva Hudlicka, Broekens has written extensively about the tool's implications for game design and even presented two games built off of it.[22] The rhetoric around GAMYGDALA is ambitious, hyping up its capacity to make games unequivocally better by giving NPCs realistic and dynamic emotions. By comparison, the games that Broekens has built with it are underwhelming. *Friend or foe*, for example, is a clunky platformer in which goomba-esque enemies respond emotionally to you collecting stars (fig. 1).[23] However, these responses are always the same: three of the NPCs will always respond positively to you collecting the stars, becoming "friends" that let you jump off of them, and three of them will always respond negatively, becoming "enemies" who attack you. For all the talk of affective gaming permitting

22. Joost Broekens, Eva Hudlicka, and Rafael Bidarra, "Emotional Appraisal Engines for Games," in *Emotion in Games*, Socio-Affective Computing (Cham: Springer International Publishing, 2016), 215–32, https://doi.org/10.1007/978-3-319-41316-7_13.

Joost Broekens, "Emotion Engines for Games in Practice: Two Case Studies Using Gamygdala" (IEEE, 2015), 790–91, https://doi.org/10.1109/ACII.2015.7344662.

Frank Kaptein and Joost Broekens, "The Affective Storyteller: Using Character Emotion to Influence Narrative Generation," in *Intelligent Virtual Agents*, Lecture Notes in Computer Science (Cham: Springer International Publishing, 2015), 352–55, https://doi.org/10.1007/978-3-319-21996-7_38.

23. https://ii.tudelft.nl/~joostb/gamygdala/friend_or_foe.html

more nuanced, complex, and realistic emotions, the results in practice are not much more complicated than the standard numerical NPC approval tracker used in an RPG.

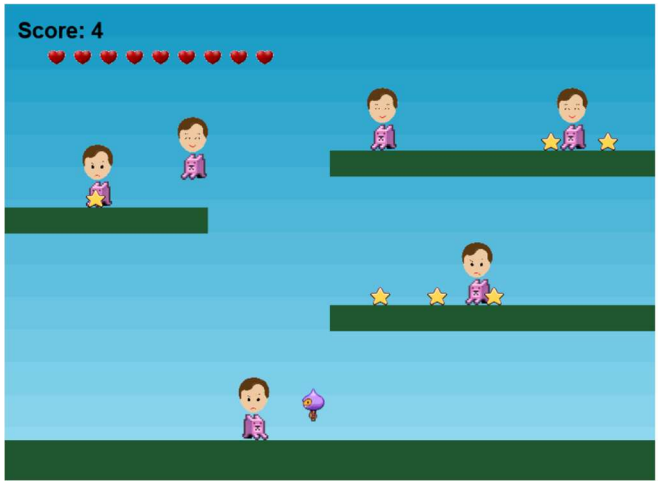


Figure 1. A screenshot of *Friend or foe* showing the emotive faces that appear about NPCs after collecting a star. These faces demonstrate how each NPC feels toward the player. NPCs that are smiling like the player. NPCs that are frowning will attack them. Screenshot taken by author.

Under the hood of the engine is a different matter. I want to set aside the question of how GAMYGDALA is, or could be, used and instead consider the engine itself and how, precisely, it implements emotion in its code. Though GAMYGDALA provides support for both internal and social emotions, for the sake of simplicity, I will be focusing only on internal emotions— that is, emotions that are not influenced by relationships between NPCs. Technical specificity is key to my argument, but I endeavor to present GAMYGDALA's mechanisms as straight-forwardly as possible and supplement this with examples where helpful.

Before discussing the appraisal mechanism (how emotions are generated from the world state), it is important to briefly define GAMYGDALA's fundamental base objects, or classes for those familiar with object-oriented programming: agents, goals, emotions, and beliefs (and relations, but again, I will not be discussing this). Agents are representations of non-player characters; they have a list of goals and a list of emotions. Goals are game states with a set utility value that determines whether that goal is desirable for the agent that owns it. Emotions have a

name and an intensity, which determines how strongly the agent that owns it is experiencing it. There are eight internal emotions in GAMYGDALA: hope, fear, joy, distress, satisfaction, fearsConfirmed, disappointment, and relief. Finally, beliefs are game events, and they are the basis for emotions being generated. Beliefs track how they affect the goals of non-player characters, either positively or negatively, as well as their own likelihood of being true. Taking an example from the paper in which GAMYGDALA was originally proposed, these four components work together as follows. A knight has the goal to kill a monster. When they find a magic sword, a belief is triggered that affects this goal because it is now easier to kill the monster. Finally, a new emotion is generated by appraising the belief. Because the knight wants to kill the monster but has not done so yet, the emotion that is generated is hope.[24]

But this is a high-level overview of the process. What is happening under the hood? Let us stick with the example of generating hope and consider first, the general algorithm for determining if hope is being experienced, and second, what this algorithm looks like in code. The former, also known as pseudocode, looks like this:

$$(des(b, g, self) > 0, L(g) < 1, \triangle L(g) > 0) \vee$$
$$(des(b, g, self) < 0, L(g) > 0, \triangle L(g) < 0) \quad [25]$$

Derived from Otorny, Clore, and Collins' model of appraisal in *The Cognitive Structure of Emotions*, this is a series of true/false evaluations that determine whether the conditions for experiencing hope have been fulfilled.[26] The top line says that you should begin by evaluating

---

24. Alexandru Popescu, Joost Broekens, and Maarten van Someren, "GAMYGDALA: An Emotion Engine for Games," *IEEE Transactions on Affective Computing* 5, no. 1 (2014): 36-37, https://doi.org/10.1109/T-AFFC.2013.24.

25. Popescu et al., "GAMYGDALA," 37.

26. Andrew Ortony, Gerald L. Clore, and Allan Collins, *The Cognitive Structure of Emotions* (Cambridge: Cambridge University Press, 1988), https://doi.org/10.1017/CBO9780511571299.

the desirability of a given belief (game event) for a specific goal and agent (in this case, self). Desirability is measured on a scale between -1 and 1 (inclusive) depending on to what extent the belief advances the goal, from completely undesirable (-1) to completely desirable (1). For example, the knight attaining the sword makes it easier to kill the monster, so that belief's desirability would be positive. However, having the sword would not guarantee that the monster died, so the desirability would be less than 1. This is the case represented by the first evaluation: the belief is desirable, but it would not achieve the goal itself. The other two expressions in this line evaluate the likelihood of the goal occurring ($L(g)$) *and* how much that likelihood has changed over time ($\triangle L(g)$). Likelihood is similarly measured on a scale between 0 and 1 (inclusive), from impossible to achieve (0) to definitively achieved (1). Thus, if a goal likelihood is less than 1, it has not happened yet, but if the change in goal likelihood is positive, that means it has become more possible to achieve. Hope can thus be defined as "a desirable goal has become more likely" or, interpreting the second line of pseudocode, "an undesirable goal has become less likely".

We can see an imperfect translation of this definition in the code of GAMYGDALA itself:

```
if( utility >= 0){
      if ( deltaLikelihood >= 0){
         positive = true;
      }else {
         positive = false;
      }
   } else if ( utility < 0){
      if( deltaLikelihood >= 0){
         positive = false;
      } else {
         positive = true;
      }
   }
if(likelihood > 0 && likelihood < 1){
```

```
if (positive === true){
    emotion.push('hope');
}else {
    emotion.push('fear');
}
```[27]

Rather than use desirability, in practice, GAMYGDALA uses the utility of the goal, which is similarly measured on a scale of -1 to 1 but is instead (as mentioned earlier) a set property of a goal itself instead of being calculated dynamically. Utility is then used to determine whether the experience being evaluated is positive or negative. Interpreting the first chunk of code, we can see that if the goal's utility is positive and its change in likelihood is positive (i.e., it has become more likely), then this is a positive experience, with similar conditions for a positive but less likely goal (negative), a more likely negative goal (negative), and a less likely negative goal (positive). In another series of if-statements, the positivity of the current game state is then evaluated against likelihood, which operates in much the same way as the pseudocode: if likelihood is greater than 0 and less than 1 for a positive goal, then that is labeled as hope. By the same token, however, if likelihood is greater than 0 and less than 1 for a *negative* goal, then that is labeled as fear.

This example brings the mechanics of affect as information into sharp relief. Feeling is given specific and rigid structure. It is determined with inflexible mathematical precision. A desirable goal becoming more likely can only ever be called hope, not anticipation or sanguinity, and hope can itself never be anything else. It is fixed within a narrow set of inequalities and positivity. This leaves out a great many expressions of hope, particularly negative ones. Rare as

27. Gamygdala.js, lines 377-395, archived at https://github.com/broekens/gamygdala, last update March 18, 2016, accessed May 2, 2022

they may be, they are still present within other cultural artifacts. Take, for example, the hit Mountain Goats song "No Children."[28] Structured around the refrain "I hope," it features the bitter, violent seethings of a man in a hateful marriage who wishes miserable things for both himself and his wife if only to bring her down with him. So goes the only lines repeated in the chorus-less tune: "I hope you die. I hope we both die."[29]

GAMYGDALA does not claim to capture the complete range of human emotional experiences. It can perhaps be forgiven for not including this particularly negative and irrational take on hope. But it does claim that it, and the epistemologies and ontologies that undergird it, *could* capture this by asserting that emotion is basically, fundamentally, and universally explainable. In an essay explaining the tool's design and utility, Broekens et al. remark that, like it is possible to model physics in games, it is possible to model emotions because "the laws of physics don't change and the laws of emotion don't change either."[30] This notion, that emotions have universal and eternal governing laws on par with those that control the most basic and objective forces of the observable universe, gives me immense pause.

So too does Broeken's usage of desirability in GAMYGDALA's pseudocode, flattening this complex and highly political evaluation into simple quantification. Speaking as a queer person, desirability is anything but quantifiable or a simple matter of marking an event as positive or negative. Indeed, some of the things I desire most in this world also cause me the most fear— a result of the systemic homophobia which has conditioned me to view my attraction

28. Rebecca Jennings, "What Happens When Your Favorite Thing Goes Viral?," Vox, October 26, 2021, https://www.vox.com/the-goods/22745655/mountain-goats-no-children-tiktok.

29. The Mountain Goats, "No Children," track 7 on Tallahassee, 4AD, 2002, Spotify: https://open.spotify.com/track/5cxnSTLzGD1t9xcdmJYFVB?si=8723611e3b144091.

30. Broekens, Hudlicka, and Bidarra, "Emotional Appraisal Engines for Games," 220.

to women as predatory, less-than, aberrant. How could such irrationality or competing emotions be represented in an NPC through a system like GAMYGDALA? How could a universal law of emotion capture their messiness? How could it be detected through physiological data, translated into signals, and passed through data clean-up and noise reduction algorithms until coming to be definitively and objectively reproduced by a computer? How, recalling the Porpentine quote that started the chapter, could the fragility of queer desire survive being determined by affect as information?

The answer is that it cannot.

There is an irreconcilability here that queer theory illuminates between the subjective, political messiness of emotion in everyday life and the emotions on offer in a game engine like GAMYGDALA and a framework like affect as information. I come to this understanding of queerness and messiness from works like Riki Wilchins' *Queer Theory, Gender Theory*, which traces the postmodernist origins of queer theory. Wilchins frames queerness as that which resists being ordered by tidy binaries, pointing, for example, to her "female-ish" partner and her inability to be located within male/female or heterosexual/lesbian dualisms.[31] She connects this to the pursuit of singular truth at the expense of all difference— everything that cannot be explained by a singularity, everything that is messy and queer. For example, a singular truth of gender and sex as fixed, innate, natural, and binary excludes transgender, nonbinary, gender nonconforming, and intersex people like Wilchins and her partner. As Wilchins ultimately concludes, "[f]inal, singular truths may make perfect sense when we're dealing with measurable phenomena [like physics] but almost none when it comes to highly politicized bodily

31. Riki Wilchins, *Queer Theory, Gender Theory: An Instant Primer* (2004; New York: Magnus Books, 2014), 45-46.

characteristics like sex, gender, desire, or race."[32] We can see echoes here of Broeken et al.'s laws of emotions, which is perhaps not a binary but certainly imposes a singular, quantifiable truth of affective experience. This is the structure of feeling present in engines like GAMYGDALA and other attempts at creating an affective game engine: an emotional determinism by which all feelings are reduced to one of eight or sixteen or twenty-four or x finite, discrete, quantifiable states.

Enumerating everything that is lost in this reduction, that fails to be represented by GAMYGDALA, that escape affect as information, would be an endless and largely pointless task. There is no complete enumeration, that's the point. When we start, in Dror's words, discoursing in numbers, a lot of queer mess and unquantifiable emotions get left out. So too does the political construction of the body, reduced as it is to a transmitter of signals for a computer to interpret or, in the case of NPCs, left out entirely. And this is, in programming parlance, not a bug but a feature. A necessary loss wrought the moment that emotion entered the psych lab as a consequence of making it fit within a logical epistemology which demanded rational measurement.

There is a great irony at the heart of the project of affective gaming. In seeking to produce emotionally complex games and characters, under the hood, it has enacted a reductive, simplistic structure of quantifying and determining emotion that cannot hope to capture the realism that it strives for.

32. Wilchins, *Queer Theory, Gender Theory: An Instant Primer*, 48-49.

Affect as Interaction

But the structuring of feeling is inevitable, inescapable, a result, as I argued in the introduction, of any contact with any tool. We are in an endless state of affecting others and being affected in turn. For the second half of this chapter, then, I want to consider an alternative to affect as information and emotional determinism, something more amenable to queer expression and a whole mess of other feelings. To that end, I am not the first scholar in HCI to be skeptical of affective computing's means and ends. Indeed, my own concerns have been refined and expounded through intradisciplinary critiques, particularly Kirsten Boehner, Rogério DePaula, Paul Dourish, and Phoebe Sengers' 2006 paper, "How Emotion is Made and Measured." As mentioned at the top, this is where the specific language of "affect as information" came from. Additionally, Boehner et al. oriented me toward the epistemological origins of affective computing and the process by which emotions were, in their words, rehabilitated: divorced from the feminized connotations of hysteria and losing control by being recast as a part of rational decision making.[33] But more than mere critique, Boehner et al. also offer an alternative to affect as information. This other formulation, which they call affect as interaction, is at once a definition of affect and a loose set of design and evaluation parameters. They derive its definition not from lab experiments in psychology but from anthropology. They point out that emotions are not (as Broekens et al. make them out to be) universal and inflexible but are rather shaped by cultural context. For them, affect is the result of a mutually constitutive *interaction* between people and their sociocultural context. Through encounters not only with each other, but with nature, ritual, tradition, history, artifacts, and systems.[34]

33. Boehner et al., "How Emotion Is Made and Measured." 276-78.

34. Boehner et al., 278-80.

This definition of affect then intersects with HCI through the aforementioned design principles. Boehner et al. develop affect as interaction through a series of paired case studies that highlight the design differences that result from each paradigm even when building similar technologies. They synthesize their results as follows: the interactional approach "[1] recognizes affect as a social and cultural product [2] relies on and supports interpretive flexibility [3] avoids trying to formalize the unformalizable [4] supports an expanded range of communication acts [5] focuses on people using systems to experience and understand emotions."[35] In general, there is an emphasis on subjectivity, interpretation, and relationality which breaks with the quantification and reductionism discussed with regards to affect as information. This has significant implications for considering how our tools structure our feelings. Affect as interaction explicitly rejects a deterministic approach in which the production of a singular, discrete, stable, quantifiable emotion is the goal.[36] Rather, they focus on how emotion is co-constituted by user, tool, and the context in which they encounter one another, which also includes the researcher. Thus, while our tools still play an active role in structuring our feelings, this role opens up interpretive and expressive possibilities, rather than narrows them down.

As an approach to affect, however, affect as interaction has not enjoyed the same widespread usage and academic respect as affect as information. In researching this framework, I found little by way of direct follow-up. Boehner and co-author Phoebe Sengers have published some other works that are similarly critical with how HCI has approached affect and aesthetic.[37]

35. Boehner et al., 283-84.

36. Boehner et al., 288.

37. Phoebe Sengers et al., "The Disenchantment of Affect," *Personal and Ubiquitous Computing* 12, no. 5 (2007): 347–58, https://doi.org/10.1007/s00779-007-0161-4.

However, these papers are more than a decade old, and I struggled to find any more recent engagement, whether in the form of expansions or generous critiques. Indeed, the few major works that cite "How emotion is made and measured" seem resolutely unwilling to take seriously its criticisms of affect as information. They often frame the piece's contributions in terms of a cognitivist versus phenomenologist[38] epistemological turf war, rather than engaging in good faith with the anthropological model of emotion being offered.[39] UX scholar Marc Hassenzahl, for example, had this to say about Boehner et al.'s work: "we all like to be challenged; we all like beautiful things, we all care about what others think about us and we all like romantic sunsets [...] A poet may find beautiful words to describe her experience, this does not make it superior to what more mundane people experience."[40] This is a bizarre detour into analogy and truism for an otherwise rhetorically straightforward paper, but I take him to be arguing that it is the experience "itself" that matters, not the words used to describe it. However, it is precisely Boehner et al.'s point that there is no experience "itself", exterior to cultural and social context. We do not all, for

---

Kirsten Boehner, Phoebe Sengers, and Simeon Warner, "Interfaces with the Ineffable: Meeting Aesthetic Experience on Its Own Terms," *ACM Transactions on Computer-Human Interaction* 15, no. 3 (2008): 1–29, https://doi.org/10.1145/1453152.1453155.

38. Phenomenology, like affect, is a very different matter across the disciplinary divide between the humanities and the social sciences. Rather than being concerned with the philosophical matter of bodies and their orientations, here, phenomenology refers to a qualitative research approach focused on capturing a single experience (phenomenon). See John W. Creswell, *Qualitative Inquiry and Research Design: Choosing Among Five Traditions* (Thousand Oaks, Calif: Sage Publications, 1998).

39. Effie L.-C. Law and Paul van Schaik, "Modelling User Experience – An Agenda for Research and Practice," *Interacting with Computers* 22, no. 5 (2010): 313–22, https://doi.org/10.1016/j.intcom.2010.04.006.

40. Marc Hassenzahl, "User Experience (UX): Towards an Experiential Perspective on Product Quality," IHM '08 (ACM, 2008), 14, https://doi.org/10.1145/1512714.1512717.

example, like beautiful things because what is considered normatively beautiful is shaped by cultural systems of oppression like white supremacy, fatphobia, misogyny, and ableism.

The fact remains, however, that affect as interaction's impact on the broader field of affective computing has been limited. There have been no specific interventions into affective gaming nor, by extension, the affective game engine. So, I will draw my own conclusions about what affect as interaction could mean for the game engine. I will then connect these takeaways with work in the queer games scene that is, I believe, allied (if unintentionally so) with the project of affect as interaction. In lieu of a specific affective game engine to study as I did with GAMYGDALA, I will focus on my experience using/playing one of these works: a recalcitrant drawing program called *Become a Great Artist in Just 10 Seconds* by Michael Brough and Andi McClure. By attending to how the engine structured rather than prescribed my feelings, I argue that this tool envisions new affective possibilities for engine design, usage, and relations in-line with the ethos of affect as interaction.

The Point is the Process: Speculative Development Tools

Taking affect as interaction as our starting point, then, how might we revise our understanding of affect's relationship to the game engine? Notably, from this perspective, we can understand affect as more than something game engines can produce in their games, more than the code that it provides. Game engines can themselves be affective because they are socially and culturally-embedded sites of interaction. Stated more strongly still, game engines are always already affective by virtue of being sites of interaction, insofar as all sites of interaction give rise to affective responses. What affect as interaction orients us toward, then, is not a binary, yes or no

question of "is this game engine affective or not?" but a more open-ended question of *how* a game engine is affective.

In answering this question, I am particularly drawn to Boehner et al.'s final distinction between affect as information and affect as interaction: that affect as interaction takes as its goal allowing users to experience and understand their own emotions. This pairs well with the primary consideration of this chapter: how game engines structure feeling. From this perspective, an (actively) affective game engine would not determine a user's emotions for them but rather "provide a resource for emotional meaning-making."[41] Feeling would still be structured, but it would be done in such a way that users can reflect on and make sense of the experience for themselves.

What this means for designing game engines is a trickier matter. I cannot, for example, offer something paralleling the four requirements that Hudlicka laid out for her affective game engine because I do not want to prescribe what is required to experience emotions. In considering affect as interaction as a practical matter, then, I want to turn to queer game developer and scholar Robert Yang. In particular, I am drawn to his writings—in the form of a blog post adaptation of a talk he gave at Indiecade East 2015—on what he terms speculative development tools.[42] Speculative development tools are not precisely defined, but they can be understood as creative software that emphasizes process over product. That is, they can be used to make digital art objects of a variety of forms, from 3D meshes to fractal drawings to music and sounds, but this is not the point of using them. The point, Yang says, is to feel them: to

41. Boehner et al., "How Emotion Is Made and Measured," 287.

42. Robert Yang, "We Are Drugs; Speculative Dev Tools and Psychedelic Hologram Futures.," *Radiator Design Blog* (blog), February 18, 2015, https://www.blog.radiator.debacle.us/2015/02/we-are-drugs-speculative-dev-tools-and.html.

experience using them and reflect on how they make you feel about the creative process. And speculative development tools lend themselves well to being felt because they are speculative. That is, they imagine new, often strange, unintuitive, and queer ways of doing art, such as a modeling program that lets you sculpt meshes like wet clay or a music program that lets you breed sounds.[43] The point is the process because the process is often unlike anything you have ever encountered before.

It is this prioritization of feeling and experience that aligns speculative development tools with affect as interaction, despite being academically disparate projects. Speculative development tools are not concerned with their end results but with how the user feels in the meantime, and rather than determining their emotional response, they provide an interpretive context in which emotions can be felt and understood. Furthermore, they are of profound relevance to my question about how game engines structure feeling for their users because they are a type of game engine. This may seem a strange claim, since I just said that speculative dev tools do not care about their products, and what is a game engine if not a tool for producing games? Recalling a bit of definitional work done in the introduction, I want to again highlight the definition of game and engine that I am using in my thesis. First, from queer game maker Anna Anthropy, "[a] game is an experience created by rules."[44] Second, from the co-creator of the first game engine, John Romero, "the engine 'is the heart of the car, this is the heart of the game; it's

---

43. https://stephaneginier.com/sculptgl/

https://candle.itch.io/evosfxr Note that EVOSFXR was built in Adobe Flash and thus no longer runs, but its itch.io description conveys the point well enough.

44. Anna Anthropy, *Rise of the Videogame Zinesters: How Freaks, Normals, Amateurs, Artists, Dreamers, Drop-Outs, Queers, Housewives, and People Like You Are Taking Back an Art Form* (New York: Seven Stories Press, 2012), 43.

the thing that powers it'."[45] If a game is an experience, and the engine is what powers a game, then an engine is that which powers an experience, that which is at its heart. I think that is justification enough for considering such highly experiential creative software as speculative dev tools game engines.

Returning to the heart of my own matter, I want to further illuminate how game engines might structure feeling by "providing resources for situated emotional meaning-making" and consider at-length a specific speculative dev tool.[46] For this purpose, I have chosen the first tool on Yang's list, Michael Brough and Andi McClure's *Become a Great Artist in Just 10 Seconds*. This delightfully unintuitive art program is exemplary of how our tools can simultaneously strongly structure our feelings while also leaving room for interpretive flexibility and meaning-making through process rather than product. I offer a deep analysis of my own experience using *Become a Great Artist* and consider how it co-constituted my affective response through its formal properties. I emphasize its deliberate affective construction, such that it provides the resources to think, feel, and do creative making differently, and to make meaning through this process. Though this may not be a goal or a want for all game engines designed under the paradigm of affect as information, it is a generative place to get started.

Sensory Overload: Become a Great Artist in Just 10 Seconds

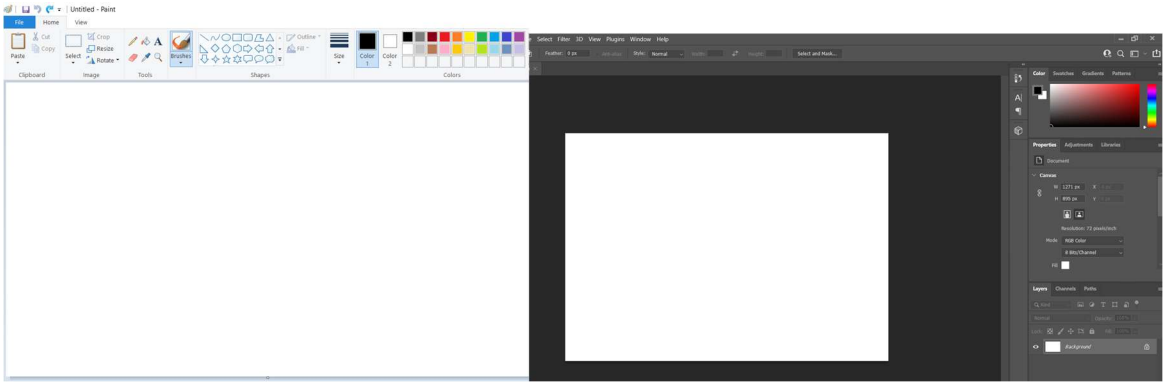
Opening *Become a Great Artist in Just 10 Seconds* for the first time, it becomes immediately obvious that this is not, despite whatever is intimated by the title, a normal drawing program.

---

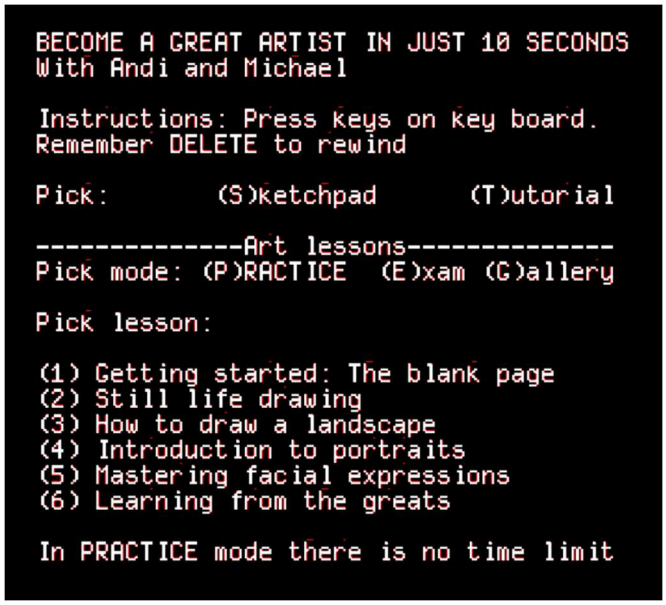
45. John Romero, quoted in Henry Lowood, "Game Engines and Game History," in *History of Games International Conference Proceedings*, eds. Carl Therrien, Henry Lowood and Martin Picard (Kinephanos: January 2014)186.

46. Boehner et al., "How Emotion Is Made and Measured," 282.

Programs like MS Paint or Adobe Photoshop open to a blank canvas (figs. 2 & 3), but *Become a Great Artist* opens on a menu (fig. 4). This menu is purely textual and responds only to keyboard strokes, and, as the instructions make clear, this is true for the program as a whole. Conventional art programs generally try to simulate the real experience of drawing on paper by relying on a mouse or stylus as the primary form of input. These modes of input require gestures that more closely resemble their analog analogues and have over time produced numerous conventions such as left click to "touch" the brush to the canvas, drag the mouse to smear the desired color across, and release the left mouse button to "pick up" the brush once more. But there is, as the title screen warns, no left-clicking in *Become a Great Artist*. There is no clicking at all, only pressing keys, not only to navigate the menu but as the mechanism for drawing itself.



Figures 2 & 3. Screenshots of the default MS Paint and Adobe Photoshop interfaces. Taken by author.



Figures 4. Screenshot of the menu screen for *Become a Great Artist*. Taken by author.

Of all the various modes that *Becomes a Great Artist* offers, sketchpad sounds the most like what we would expect from a typical drawing program, so let us start there. We might anticipate something like the Photoshop interface: a white rectangle imitating the blank canvas awaiting our input with a brush picker somewhere, somewhere else a color picker that lets us run the gamut of our computer-legible color space of choice, perhaps some miscellaneous tools for selecting and manipulating parts of our drawing or eye-dropping colors, and so on. What we get instead is a canvas pre-populated by one of a dozen or so pixelated defaults, no tools of any kind, (fig. 5) and, perhaps most surprisingly, sound.

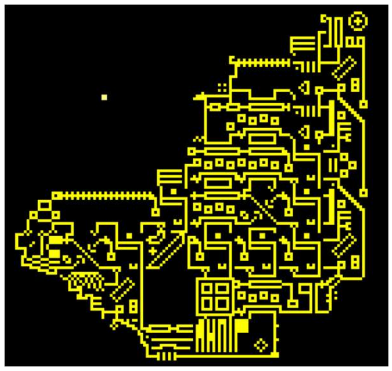
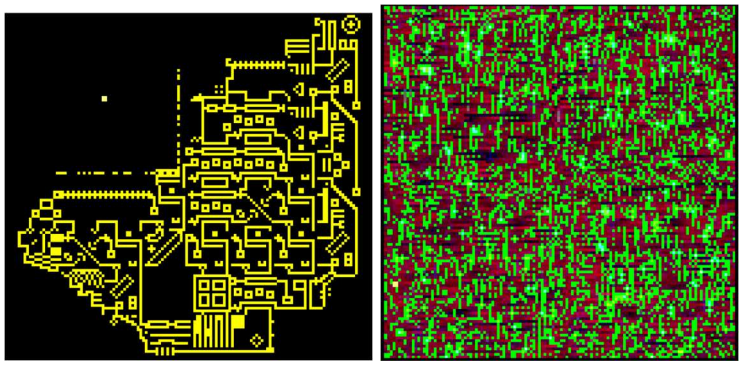


Fig. 5. Screenshot of a pre-populated canvas in *Become a Great Artist*. Taken by author.

As I am typing this, *Become a Great Artist* open in another window, I am treated to a short, looping soundtrack consisting of a brief, wet, mechanized sliding sound—a computerized *whoop*, if you will—followed by some silence and then some muffled noises that seem, to me at least, to be people talking and moving objects around or perhaps closing doors. Then, after a brief beat of silence, the whole thing repeats. At least at first, it is not particularly unpleasant, distasteful, or annoying, though it becomes repetitive quickly and can be disquieting as the only sound looping over and over again in your ears. Still, it is more strange than anything else. Ambient noise is a rare thing to encounter in any piece of software, let alone cultural software in which noises are almost always and only user-generated. Google Docs does not queue the classical music I use to focus when I open up my thesis. Unity does not automatically provide background music for my games. Even music software like GarageBand and Sibelius require user input before the noise making begins. *Become a Great Artist* is, in a word, obtrusive. Although it lacks MS Paint or Photoshop's plethora of menus and encroaching sidebars with its stripped-down interface, it more than compensates by presenting an already filled canvas and pumping strange, distorted sounds into the user's ears.

Returning to the moment of opening up the prepopulated sketchpad in *Become a Great Artist*, we begin "drawing". But this is unlike any drawing I have ever done before. How do you even draw with a keyboard? I start, then, with a few hesitant, experimental flicks of the fingers over the keyboard, alighting ons the spacebar. The program chirps with another computerized sound and the drawing changes, adding dashed yellow lines along an off-center vertical and horizontal axis (fig. 6). Why does this happen when I press the spacebar? I have no clue. Indeed, further experimentation reveals that, while each alphabetical and arrow key is mapped to exactly one effect in the program, these effects are themselves varied, strange, impossible to predict in

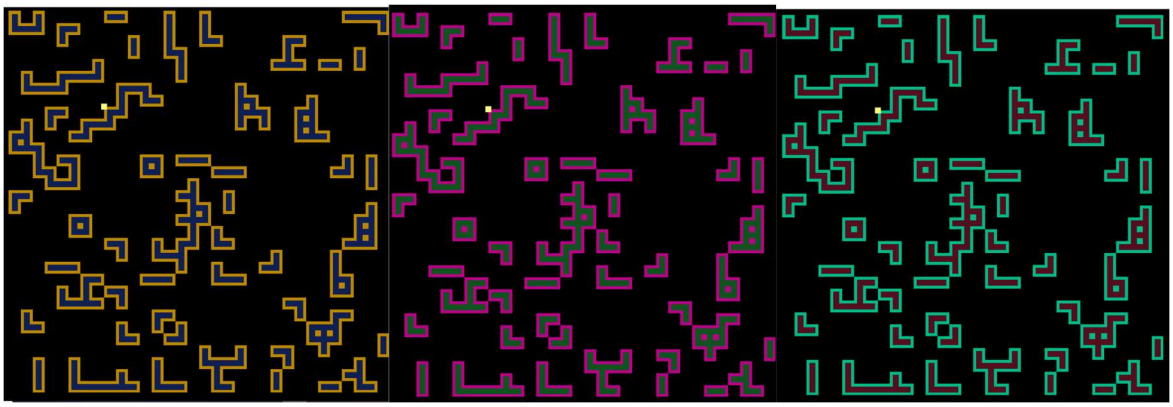
advance, and frequently accompanied by even more sound effects. There is seemingly no rhyme or reason to any of it— just distorted pixels and distorted noises. Figure 7 shows what a few seconds of key smashing has made of the yellow building that my canvas began with: a wash of formless neon pixels, a mess of shapeless color, a symphony of industrial noise.



Figures 6 & 7. Screenshots of a canvas in *Become a Great Artist* after manipulating the drawing in a variety of ways: subtly, in the case of figure 6, and drastically, in the case of figure 7. Taken by author.

None of the keys do things that are particularly conducive to using *Become a Great Artist* as the drawing software it advertises itself to be. A small but illustrative example of this would be how *Become a Great Artist* has complicated an otherwise ubiquitous task for drawing programs: working with color. There are many keys that affect the color of the image. <V>, for example, invades the screen with black and white alphanumeric characters, and <T> has a kind of smear-effect that also washes out color. As both these examples suggest, however, these keys also affect the canvas in other, potentially undesirable ways. There is only one key that *only* alters color, but its logic for doing so is difficult to parse. <J> maps each color on screen to a new color, retaining the overall form of the drawing. However, the basis for doing so is far from something as simple as inverting the image. It instead switches between different color palettes, and though it may alternate between a set few, it regularly generates new ones, making it impossible to know if it will cycle back to an old one or throw something different at you (figs. 8, 9, 10). Thus, even when you have some understanding of what keys do what, a combination of

randomness in the outputs themselves, the different starting image you work with each time, and the imprecise controls for manipulating form and figure, make it a difficult tool to use intentionally.

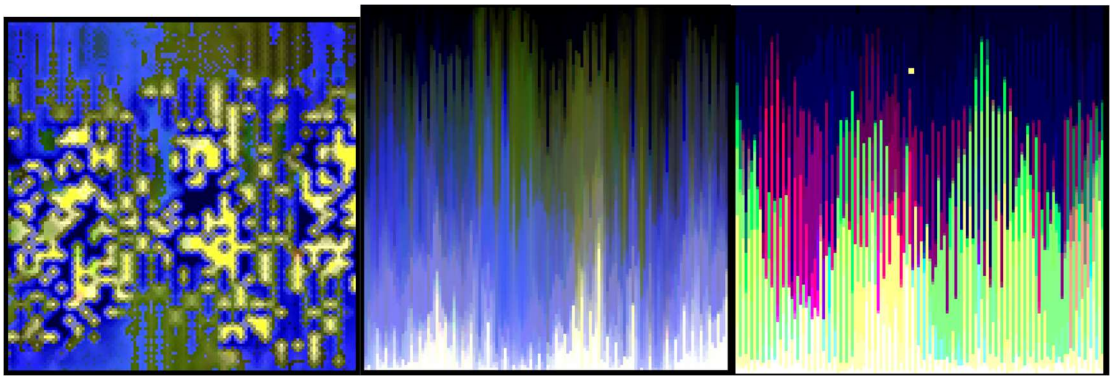


Figures 8, 9, and 10. Screenshots of the same canvas after pressing <J> to alter the color scheme. Taken by author.

This is made all the more apparent in the Art Lessons mode. After all, *Become a Great Artist* promises more than being a drawing tool. It offers to teach you to be a great artist in a mere ten seconds. Premised on learning through imitation, each of the six exercises offers a model image that you are trying to perfectly match by manipulating your randomly generated base image with the ever-temperamental keyboard. However, the models in question would be difficult for a novice artist to approximate in a more yielding drawing software, and are downright impossible with the tools at hand… such as recreating Vincent van Gogh's The Starry Night (fig. 11). This is an immensely challenging task given the tools that you are working with, and that is before even taking into account the ten-second limit imposed in exam mode. This mode adds a ticking stopwatch which mocks you as your fingers scramble over the keyboard in an earnest but fruitless effort to reshape your canvas into something resembling the target image.

Figure 11. A screenshot of Art Lesson mode, showing off the image you are supposed to be attempting and a percentage of similarity between it and your efforts. Taken by author.

Yet this does not preclude *Become a Great Artist* from being used to make something artistically meaningful or emotionally resonant. Because nothing can be taken for granted and everything must be accomplished and learned through rote trial and error, there is an immense joy to be found in encountering particularly pleasing interactions between keystrokes. A personal favorite combines the <F> key and the <D> key. <F> causes all the colors to "drip", creating pools at the bottom of the canvas and lines stretching up to the top. It also layers the colors, creating a vertical gradient of streaks (figs. 12 & 13). <D> pans the image, but not every part of the image moves at the same rate. This may be less obvious in other states but is quite clear after using <F> because the lowest colors move fastest, while the higher up colors remain more stationary. Furthermore, because of the spire-like shape of the streaks, while moving, the lines of lower color blur together and almost take on the form of buildings. Against the more slowly moving "background" colors, this creates a deceptive sense of depth and gives the impression of a city skyline panning past. In sidescrolling through scenes like figure 14, I find a surprising moment of rest and peace in a software that generally feels frustrating, abrasive, and even mocking.



Figures 12, 13, and 14. 12 and 13 demonstrate the effect that <F> key has on a canvas. For 14, because the image moves when holding down <D>, the individual lines of each color blur together, creating skyscraper-like forms with the faster-moving lime green buildings "in front of" the slower moving magenta ones, both against the dark blue night sky at the very top.

And *Become a Great Artist* has other pleasures as well. In addition to visual stimulation, this tool engages with touch and sound. The touch point may seem counterintuitive, given again that its input method is so radically different from other drawing methods that try to replicate the actual action. But typing is a form of touch, and one that I am personally quite fond of. I like typing. I like the sound that my keys make when I click them, I like the sense of the little ridges on the <F> and <J> keys that orient me, I like how my fingers feel when they fly across the keys with particular grace and precision. I like keyboard smashing, not only as a physical action but as a queer articulation, and *Become a Great Artist* encourages it as a more fruitful mode of interaction than trying to plan my strokes and inputs carefully and precisely.[47]

The other sensory engagement is, of course, sound. There are many background tracks, and they change with each new project. Indeed, working on a piece for a while reveals that these sounds do not merely loop but start to evolve after a few minutes. This produced a shocking

47. In certain Internet circles—or, at the very least, the Tumblr communities that I frequent—keysmashing is specifically associated with gay users. Urban Dictionary supports this connection, at the very least, calling it a "staple of the LGBT community": ur the best, ur amazing, hbd, "Urban Dictionary: Key Smash," Urban Dictionary, August 24, 2018, https://www.urbandictionary.com/define.php?term=Key%20smash.

moment for me personally when I had the tool open in the background while working on my thesis and suddenly, the droning noise I had been listening to turned into a high, piercing, alarm-like sound. Of all of *Become a Great Artist*'s aesthetic properties, these crunchy, artificial, harsh sound effects are perhaps the most difficult to understand as pleasurable. They can be grating, annoying, and even outright uncomfortable, as the anecdote about the sudden piercing noise (originally recorded with more expletives) can attest. Nonetheless, there is not anything inherently aesthetically bad about the sound design being unpleasant. It is, after all, an essential component of the affective experience of interacting with this engine. The soundtrack's obtrusive, caustic nature acts as a constant call to the present and the process. It is difficult to get lost in your work, to sink into the product with the machine's contours blurring and disappearing around you, when your tool is shrieking at you. When it demands to be felt, heard, and seen at every moment.

I take both this tactile pleasure and this aural discomfort to be part of Yang's call to use speculative dev tools to "take joy in doing."[48] This is obvious in the case of typing, as I have already established that I take great joy in the physical act, but it might seem contradictory for *Become a Great Artist*'s soundtrack. However, outside of the realm of affect as information, our emotional responses to our tools need not only be positive, do not have to be guided toward an ideal level of engagement or any feeble notion of emotional realism. To that end, *Become a Great Artist*'s aural landscape forces us to confront the fact that discomfort is a part of all doing, all creating— and it asks the user to try to enjoy it anyway. It could even be that there is, for some of us, a certain joy in this discomfort, strange as that notion may be (and utterly unthinkable through affect as information). For me, this joy is felt not in spite of the fact that it

48. Yang, "We Are Drugs; Speculative Dev Tools and Psychedelic Hologram Futures."

sometimes startled me but precisely because it did. Few tools in my life are so bold as to jar me. It was, dare I say, refreshing.

The final affective response I want to consider in relation to *Become a Great Artist* is that of the freedom in surrender. By this I mean that *Become a Great Artist* is hard to use. Even after experimenting with the program for a few hours and having a decent understanding of all the possible transformations, it remains so unpredictable that this understanding ultimately amounts to little and must be very carefully applied to be useful at all. After all, with just a single errant keystroke, the image is irrevocably changed. Because mistakes are so easy to make and even harder to rectify, because intentional actions are harder still, this engine forces me to focus on the process rather than the product: I cannot think about what I want to do, only what I feel while I am doing it, and what I felt most often was freedom. There is an ecstasy in letting go of precision, a release in succumbing to the software and the profound upheavals that result from otherwise small input actions, and this is not a realization that I would have come to without *Become a Great Artist*.

This points back, by way of conclusion, to affect as interaction and the capacity for our tools to act as sites for emotional meaning-making. Crucially, *Become a Great Artist* was able to sharply elicit various affective responses from me by insisting on its own presence as emotional context. It refused me control and it refused to disappear into the background of my creative process. But it also refused to control my actions by shepherding them toward some goal, and it refused to control my emotional response by adapting itself to pressure me in a singular direction. Together, then, we were able to work toward and through the affects that we co-constituted. This has profound implications for game engines as structuring feeling. It points to the possibility for engines to be strongly felt without being overly determining, and it suggests

that this structuring should be pluralistic and messy, never resolving in a single affective state but being perpetually reworked and renegotiated.

As Yang posits in his blog post, "the computer is sharing the act of drawing with you, encouraging messiness and spontaneity. Maybe true participation means not wielding complete control over something?"[49] Maybe this means not only participation in doing but participation in being and feeling as well.

49. Yang, "We Are Drugs; Speculative Dev Tools and Psychedelic Hologram Futures."

CHAPTER 2

At the Heart: How We Feel Game Engines' Structures

> *Do not let Unreal hate-fuck your imagination and squeeze your life into a claustrophobic open world survival game with a white picket fence and 2.5 subcontinents.*[1]

Game engines are always already affective: we feel things when we use them, and they play an active role in constituting what those feelings are. This role is not universally defined but manifests differently across different game engines depending on their own feelings—specifically, their wants and desires. A game engine that wants to produce specific, singular, quantifiable emotions will be structured to reductively overdetermine feeling, and to do so unceasingly toward the desired outcome. A game engine that wants users to take away their own emotional meaning will be structured to offer a deliberately affectively-charged experience that can then act as the raw materials for interpreting feeling.

And yet the above quote from queer game maker Robert Yang's manifesto "KILL UNITY; WE ARE ENGINES" points toward a potential blind spot. There are some engines that are not designed to structure feeling at all. The vast majority of game engines were not built within a given paradigm of affect theory, including those most commonly used in game design practice: the tools like Unity and Unreal which Yang rails against. Indeed, under certain strains of affect theory, these tools would be considered unequivocally *un*affective because they do not directly interpret and respond to the user's emotions.[2] *And yet they are still felt*, as Yang's

1. Robert Yang, "KILL UNITY; WE ARE ENGINES," itch.io, 2018, https://radiatoryang.itch.io/kill-unity.

2. See, for example, Kiel Mark Gilleade, Alan Dix, and Jen Allanson, "Affective Videogames and Modes of Affective Gaming: Assist Me, Challenge Me, Emote Me," in *Proceedings of DiGRA*, 2005, 1–7.

incisive diatribe attests to. They still inspire emotional responses, and those responses can be just as intense as those elicited by a more deliberately affective tool, such as Yang's fear that Unreal will "hate-fuck" his creative process. Interpreting precisely what Yang means by this is secondary to the strong feelings that this claim evinces. There is still a structure of feeling here, a set of wants and desires that users must contend with and be affected by in the process of creative production.

The issue is how to pin these structures down. They are not self-evident for much the same reason that it is not self-evident what Yang means by "hate-fuck". In the absence of a deliberate, affective design methodology, the affective construction of these game engines is obfuscated, and its effects, even more so. If Yang's description of Unreal's influence on him seems overly metaphorical and hyperbolically violent, it is in part because *actual* moments of being influenced during the game design process are hard to concretize and nail down. There is nonetheless a fear for Yang that, subtle and invisible as these influences are, they are occurring, and they are damaging his creative output.

But this inability to locate and characterize the game engine's influence through more grounded, empirical methods has become a recurring problem for critical engine studies. In this chapter, then, I argue that attending to affect can help address it. Our emotions while using game engines draw attention not only to the fact that they structure our feelings (despite any pretension to the contrary), but that they are themselves structures that can be felt. From this perspective, then, feeling becomes itself a mode of scholarly inquiry by which to better known game engines. In order to develop this line of thought, I first offer a new definition of affect, one better equipped to get at my scholarly interests than affect as information and more robustly developed than affect as interaction. I call this definition affect as relation, and I develop it by pulling from

the multiplicities of affect theories that have cropped up within the humanities over the decades. In order to ground this definition in computing technology, I supplement this definition with a brief detour into games studies to consider adjacent and ideologically-similar projects. The work of feminist games scholar Aubrey Anable in particular provides an excellent method and application of affect theory to study video games, which I draw on when I apply it to study game engines.

Thus, I bring this definition of affect to bear on the game engine. I argue that affect as relation exposes those moments where our tools structure our feelings— and, in turn, we are able to feel their structures. Through our emotional responses to game engines, otherwise invisibilized spatial metaphors such as limit, constraint, grain, contour, surface, as well as emotional metaphors such as want, desire, and even hate-fucking, become apprehendable. To demonstrate this, I pull from my own experience with game engines. I narrativize my history with the technology, beginning with a 2D engine design class in undergrad and concluding with personal experiments with Twine and Unity. I consider both moments in which an engine's structure was invisible to me and moments of affective encounter in which it became, if only momentarily, visible. Such emotional moments, even if the emotions themselves were subtle and mundane, allowed me to feel my engine's structure and, in feeling it, allowed me to understand something which I had not previously known about it. They expanded my intellectual understanding of what my tool was.

Finally, I argue that attending to such moments is valuable for advancing critical engine studies. Though I do feel that this has practical, methodological implications, for want of space and time, I will instead focus on what this means conceptually for the field. Specifically, affect as relation offers a theoretical apparatus for moving beyond a discourse that would narrowly

characterize game engines as either universally freeing or unilaterally controlling. Even when a game engine's structure cannot be seen, it can still be felt, and it is through such feeling that we can come to characterize engines beyond this binary.

Affect as Relation

At the core of this chapter is the notion that by attending to affect, we can understand engine-user relationships. It is essential, then, to move forward with a definition of affect that is equipped to deal explicitly with relationality and mutual imbrication beyond unidirectional interaction: not only how our tools affect us, but how we affect them. How they feel (about) us and we feel (about) them. But relationality is not a given in affect's definition in the humanities. This is in part because affect's definition is not a given, period. Affect theory in the humanities is a pluralistic collection of competing and overlapping lines of inquiry. Nonetheless, the problem of relationality is also owed to how the matter is dealt with in the two most prominent of these strands: Silvan Tomkins' "psychobiology of differential affects" and Gilles Deleuze's "ethology of bodily capacities."[3] The former closely resembles the approach of affect as information that was discussed in Chapter 1, in which emotions are considered to be discrete and enumerable, and has inspired works such as Eve Sedgwick and Adam Frank's "Shame in the Cybernetic Fold."[4] [5]

3. Gregory J. Seigworth and Melissa Gregg, "An Inventory of Shimmers," in *The Affect Theory Reader*, ed. Melissa Gregg and Gregory J. Seigworth (New York, USA: Duke University Press, 2010), 5, https://doi.org/10.1515/9780822393047-002.

4. Actually, whereas the terms emotion and affect are used interchangeably in affective computing, a lot of ink has been spilt in the humanities to tease out their differences. In this regard, I am inclined to follow the affective computing scholars. My thesis has enough definitional work as is without also getting bogged down in the physical (or metaphysical) distinctions between emotion, affect, and feeling.

5. Eve Kosofsky Sedgwick and Adam Frank, "Shame in the Cybernetic Fold: Reading Silvan Tomkins," *Critical Inquiry* 21, no. 2 (1995): 496–522, https://doi.org/10.1086/448761.

The latter considers affect to be a pre-subjective, physical force or intensity that is felt in the body before the mind can make sense of it. Notable research in this strand includes Brian Massumi's "The Autonomy of Affect."[6]

These are certainly divergent projects. One is founded on the naming and counting of affects, and the other insists that it cannot be named or counted because it is, at that point, cognitively understood and thus no longer actually affect. Yet there is common ground between them, and their relation to relationality is a part of it. I am inspired here from Ruth Leys' critique of the affective turn as a whole, in which she argues that both perspectives are united by what she terms anti-intentionalism: "the belief that affect is independent of signification and meaning."[7] That is, it is purely a matter of the body, and not the mind. Affect is radically and singularly individual and interior. To the extent that anything exterior to the individual is considered, it is only (and briefly) as a unidirectional stimuli, and it certainly does not have the kind of reciprocal structuring or co-constitutive capacities that I have developed throughout this work so far.

Subsequently, I do not take either of these approaches as my starting place. In fact, in defining affect as relation, I am not inspired by any one coherent stand of affect theory. Rather, I am drawn to the broader overview of general trends that Melissa Gregg and Gregory Seigworth offer in their introduction to *The Affect Theory Reader* (2010). Their writing here resists definitive singularity and embraces poeticism in a way that is intellectually inspiring all on its own, as when they describe affect theory as "chasing tiny firefly intensities that flicker faintly in

---

6. Brian Massumi, "The Autonomy of Affect," *Cultural Critique* 31 (1995): 83–109.

7. Ruth Leys, "The Turn to Affect: A Critique," *Critical Inquiry* 37, no. 3 (March 2011): 443, https://doi.org/10.1086/659353.

the night."[8] Of more immediate relevance, however, is their discussion of relationality Consider the following passage:

> Within these mixed capacities of the in-between, as undulations in expansions and contractions of affectability arrive almost simultaneously or in close-enough alternation, something emerges, overspills, exceeds: *a form of relation* [...] comes to mark the passages of intensities (whether dimming or accentuating) in body-to-body/world-body mutual imbrication.[9]

I see this excess and its temporality as being related to the supposed unmediated, bodily instantaneity of affective experience under anti-intentionalism. That is, there is always something that exceeds the singular moment of affective experience: a relationality that guides how that affect then continues to circulate in the world as it passes between bodies. In other words, even if we might accept the anti-intentional notion that relation does not produce affect, affect certainly produces relation. What's more, because, as Seigworth and Gregg say, affect passes between bodies, relation then begets affect by shuttling intensities from one form to another.

This may seem maddeningly theoretical, but there is also a certain common sense to the notion that relation both precedes and is produced by affect. "[T]he in-between" that Seigworth and Gregg mention as the origin point for affect implies that relationality exists before it. After all, for there to be something between you and I, we must be in some kind of relation with one another, whether physical ("the box is between us") or figurative ("something came between us and drove us apart"). This is the very literal grammatical function of prepositions. But we can also see in this figurative example that relationality also grows out of affective experience. This hypothetical relationship is remade, reshaped, reformed by an emotionally-charged event that soured our feelings for one another. Thus, our relationship acquires specificity through affect…

8. Seigworth and Gregg, "An Inventory of Shimmers," 4.

9. Seigworth and Gregg, 13.

and yet relationality *also* characterizes affect. It hurts more when our loved ones hurt us because of our relationship with them. Our friends can say things to us that we find funny but would consider to be outrageous or offensive coming from a stranger. The joy I feel when my cat climbs into my lap is inextricably linked to our history together, the particularity and specificity of our longstanding in-between-ness.

This, then, is what I mean by affect as relation: a fusion of the two concepts in repetitious, mutual co-constitution. Indeed, this is what I mean by structuring feeling as well, in so far as structure is a relation. Whereas Chapter 1 focused on how structure (relation) gives rise to feeling (affect), then, this chapter endeavors to consider the reverse: how feeling gives rise to structure.

Affect and Games Studies

There is, however, one drawback to this formulation owing to its theoretical basis in the humanities. Unlike the definitions of affect used in Chapter 1, this definition has not been articulated in terms of a clearly delineable zone of overlap with computing technologies. That is to say, I cannot discuss pre-existing design methodologies or evaluation mechanisms that have informed the construction of specific game engine objects. This leaves me with a question of method: how to go about bringing this definition to bear upon the game engine. To answer this question, and more generally bring my definition into contact with technology, I want to briefly consider how a similar, if not identical, understanding of affect is used by Aubrey Anable in her 2018 book *Playing with Feelings: Video Games and Affect*.

Anable also defines both her affective and her game work through relationality and the in-between-ness that it entails. She views playing video games as "opening up a 'form of

relation'" not only to the game itself but to all of its constituent parts and its larger cultural context, up to and including "ideas of leisure and play, ideas of labor, our bodies, other players, and the whole host of fraught cultural meanings and implications that circulate around video games."[10] She goes on to remark that studying these relationships, and the emotions they give rise to, cannot be accomplished by focusing on the player or the game but only by attending to the space in between them.[11] This has significant methodological implications, since it renders certain media studies techniques such as audience reception and textual analysis insufficient. Focusing on the player or the game is not enough; you have to talk about them both at once, in specific relation to one another.

In her introduction, Anable does this through an extended anecdote about a woman playing *Candy Crush Saga* (King, 2012) while waiting for the train on her way to work, which is interwoven throughout the chapter. When I think of an affectively impactful game, *Candy Crush* is not what first comes to mind. Indeed, I would be hard pressed to think of a single mobile game that has left a significant emotional impression on me. However, this is likely why Anable chose this story and this game: to highlight the capacity of *all* games to be affective because all games exist in relation to their player. In the case of Anable's commuter, the emotions that *Candy Crush* provokes are quotidian and unremarkable. They are forgettable shimmers only briefly impinging upon the everyday experience of commuting, from the momentary frustration of failing the level she is on to the disinterest as she intentionally loses again just to be done before she arrives at her stop.[12] This narration shows that the woman's feelings were structured not only

10. Aubrey Anable, *Playing with Feelings: Video Games and Affect* (Minneapolis: University of Minnesota Press, 2018), https://doi.org/10.5749/j.ctt20mvgwg, xii.

11. Anable xiv.

12. Anable, *Playing with Feelings*, ix.

by the game that she was playing but by their specific relationship. By this I mean that, had the woman been playing the game under different circumstances, somewhere other than on the train to work, her affective response might have been different as well. She might have tried harder on her last level, rather than giving up and deliberately losing. She might have been playing the game in the presence of a friend and vented to them about the difficulty. An inopportune encounter with a pet might have led to her losing sooner.

In short, specificity matters in defining relationality.

As a matter, furthermore, of method, Anable's anecdote recalls Ruth Leys' critique of the affective turn, which ends with a consideration of alternative approaches. Leys muses: "the moment one [chooses] some kind of intentionalist interpretation of the affects one finds oneself forced to provide thick descriptions of life experiences."[13] Might a similar approach of description and narration work for the game engine as well?

The Myth of a Neutral Engine

With this definition and method of affect as relation in hand, what I aim to argue in this chapter is an inversion of Anable's claim of video games as structures of feeling. The problem with adopting Anable's approach wholesale for game engines is that game engines do not structure feelings the same way that games do. This is owed to key differences between the technologies. Game engines are mediums, not media, modes of production, not products, creative pipelines, not the creative expression itself. And, crucially, they are often resolutely invested in hiding their own structure. If the point of a structure of feeling is to "[learn] something about the larger

13. Leys, "The Turn to Affect," 471.

picture that cannot all be grasped at once," then certain game engines evade being grasped through a myth of being neutral.[14] Of being structureless.

In order to demonstrate this concretely, I will compare and contrast the interfaces of two game engines: Bitsy and Unity. Bitsy offers a clear example of what a game engine's structure looks like when it is decidedly present in its interface. That is, one can look at the Bitsy interface and understand the engine's capacities: what it will allow a user to do, what it will prohibit or make difficult, what modes of creation it encourages. What, in essence, it wants. By contrast, Unity's wants are deliberately obfuscated, and that obfuscation—that apparent structurelessness—is also evident in the interface. This represents an investment in a myth of neutrality which plagues critical engine studies' epistemologies and which, I argue, affect as relation can help address.

Bitsy is a little game engine created by Adam Le Doux to be "a little editor for little games or worlds."[15] The littleness is part of what makes Bitsy highly structured, insofar as it imposes sharp limits on the game design process. Many of these limits are self-evident in the software's interface, which consists of a series of maneuverable panels that allow users to edit a specific piece of content or functionality. Just from the panels on display in figure 1, for example, we can see the following: you can only use Bitsy to make 2D games, you can only make assets that are 8x8 pixel art and rooms that are 16x16 tiles, you can only have three colors in a given scene, and you can only create four types of game objects, including only one avatar, which means your game has to be single-player as well.

14. Anable, *Playing with Feelings*, xix.

15. Adam LeDoux, "Bitsy," Bitsy, accessed May 2, 2022, https://www.bitsy.org/.

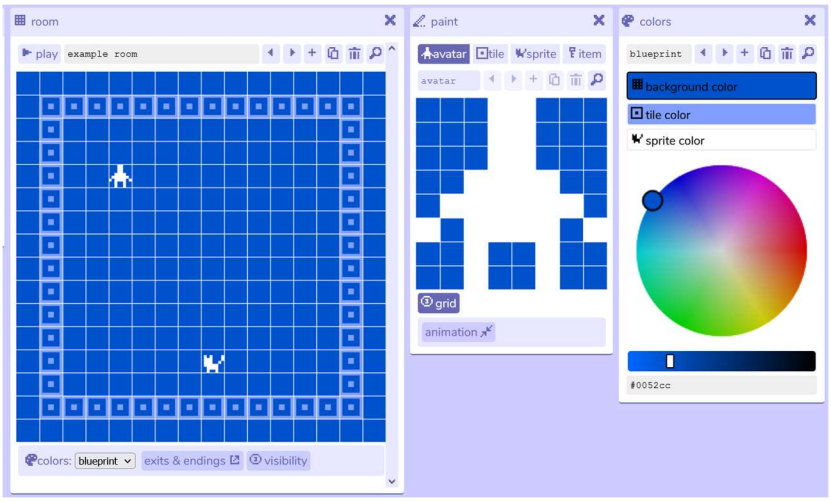


Figure 1. Screenshot of the default Bitsy interface. Taken by author

Beyond these aesthetic restrictions, Bitsy has a major mechanical limit as well, also apparent in its interface: the dialog panel (fig. 2). The dialog panel is the only place where users can write *anything*. This includes diegetic text, such as actual dialogue, extradiegetic narration that might supplement the game's limited visuals, and, perhaps most significantly, code. This means that any mechanic that a game maker wants to implement has to be routed through the dialog system, which furthermore only triggers when the avatar interacts with an object or when they move to another room. In a Bitsy game, then, players cannot look at the corner of the screen to see their character's stats, nor press a key to open their inventory. Game makers must provide objects for them to interact with, such as the reflecting pools from Porpentine Charity Heartscape's *Almanac of girlswampwar territory* (2018) (figs. 3 & 4).

Figure 2. Screenshot of the dialog tab for a personal Bitsy project. Taken by author.

Figures 3 & 4. Two screen shots of *Almanac of girlswapwar* which shows how the reflecting pool object functions: by interacting with it, the player can learn the time of day, the weather, and their current health status. Taken by author.

This is what I mean when I say that Bitsy's structure is obvious. It can be explained by reading the interface, as it were: looking at the functionality provided and the restrictions that this functionality imposes. This is in essence the kind of textual analysis that Anable proposes is insufficient for explaining affect— and I share her concerns. I have focused here only on the engine itself, abstracted from the radical specificity of a particular relationship with a particular user. I have not attended to affect at all. Thus, recalling that affect/relation and feeling/structure are inextricable and co-constitutive, my understanding of Bitsy is woefully incomplete.

However, this approach gets me a lot further toward understanding Bitsy and its influence on the user than it does when it is applied to Unity. Superficially, Unity's interface is not dissimilar to Bitsy's; both consist of modular panels that can be rearranged at the user's discretion and grant access to various parts of the engine (fig. 5). Unity's panels are, however, immensely more complicated, such as what is shown in the inspector for the "Main Camera" object. Even I, with my undergraduate experience in game engine design and computer graphics, do not know what all these labels refer to or what they all do. It is not immediately clear what they enable or what they restrict.

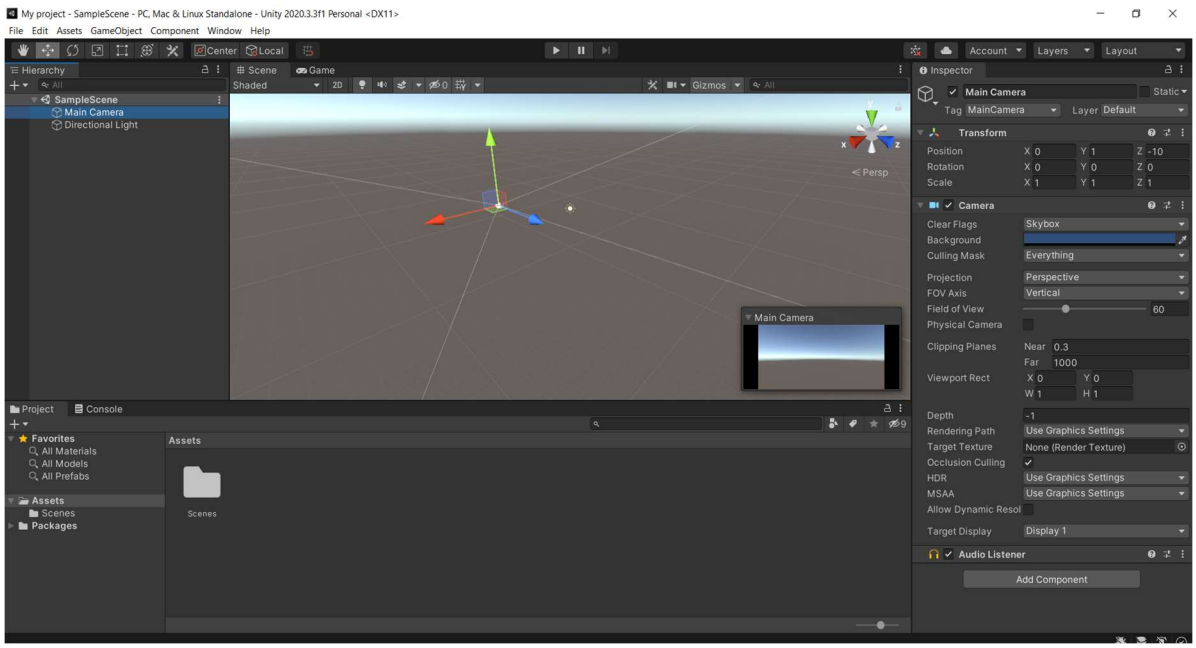


Figure 5. Screenshot of the default Unity interface and default new 3D unity project, in dark mode. Taken by author.

Technical complexity is only one part of the problem, however. The other part is the default game world that is generated. Bitsy's default world is pre-populated with three of the four types of game objects (an avatar, a sprite, and a tile), and they are not simply dropped into the world but have been arranged to give that world a meaningful shape: that of a room with a person and a cat in it. There is, in short, purchase here. There is something to grab on to in order to understand how Bitsy structures itself and how it might, in turn, structure its user. There is no purchase in Unity's default 3D world (fig. 6). There are no clearly identifiable limits, no self-evident constraints, no obvious expressions of desire or structure. There is only the infinite coordinate plane, a light source, and a camera object waiting to perceive a world that is, apparently, boundless in its possibilities.

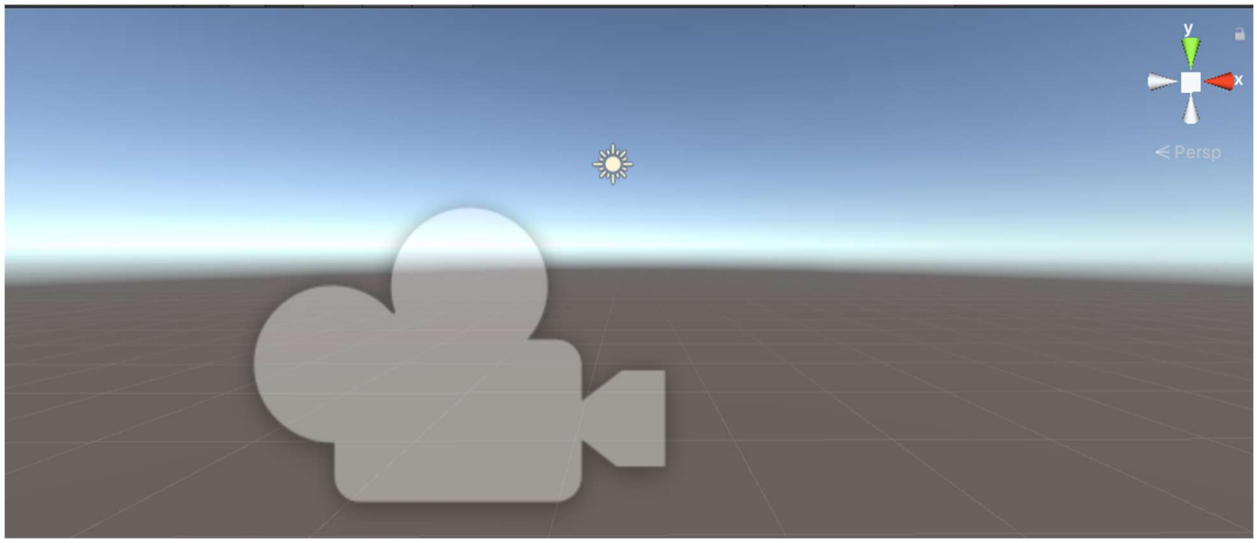


Figure 6. Screenshot of the default Unity 3D world. Taken by author

This is what I term the myth of neutrality: the myth, perpetuated in the design of and discourse around game engines like Unity and Unreal, that they are boundless. That they can be used to make everything because they do not desire to make any one thing in particular. That they are equally receptive to and accommodating of every creative possibility. That they are a neutral participant in an unmediated game design process.

This claim to structurelessness is a well-known problem for critical engine studies, and one that many authors have sought to expose. Some have even done so in relation to the Unity game engine. In their book *The Unity Game Engine and the Circuits of Cultural Software*, for example, Benjamin Nicoll and Brendan Keogh frame the problem as follows:

> While Unity makes claims to empowerment and democratization by virtue of being a general-purpose game engine, most [interview] respondents regarded Unity's [influence on the game design process] as annoying, presumptuous, or limiting. This speaks to the (long-standing) fantasy of a blank, neutral game engine through which 'any' videogame idea can be realized, and upon which 'any' workflow can be utilized.[16]

16. Benjamin Nicoll and Brendan Keogh, *The Unity Game Engine and the Circuits of Cultural Software*, 1st ed. 2019. (Cham: Springer International Publishing, 2019), 65, https://doi.org/10.1007/978-3-030-25012-6.

In this quote, Nicoll and Keogh articulate the fact that, despite Unity's claims to the contrary, it does have a structure— what they term grain in their analysis. They further consider how this structure was perceived by their participants, independent game developers from Australia who use Unity as part of their work. They interpret the negative responses as indication of a fantasy for a less structured game engine, a game designer corollary to the myth of a neutral game engine, which I have located within the engine itself.

Yet even as the neutral engine is only a fantasy/myth, Unity's investment in it makes it difficult for Nicoll and Keogh to characterize its structure/grain in detail. Rather than considering the engine itself, the first few examples that they give are from Unity paratext: those objects such as documentation, tutorials, assets on the Unity marketplace, press releases, blog posts, stackexchange help threads, and official Unity short films which float in the orbit of the engine "itself", the flotsam and jetsam that we also open ourselves up to relating to when we use Unity. They discuss, for example, the official raycasting tutorial, which is a method used to determine what a line projected through digital space will intersect with. While this technique is general purpose—Robert Yang, for example, used it in his game *Polaris* (2010) as a mechanic for stargazing[17]—the Unity tutorial frames it around aiming and shooting a gun.[18] Crucially, through this expectation, Unity enables the conditions through which the expectation will be fulfilled. The tutorial does not set the realm of possibility for using raycasting, but it does *orient* users within this realm such that some possibilities (in this case, shooting from a gun) are closer, conceptually and materially, than others.

---

17. Robert Yang, "Liner Notes: Polaris and How It Works," *Radiator Design Blog* (blog), March 4, 2010, https://www.blog.radiator.debacle.us/2010/03/liner-notes-polaris-and-how-it-works.html.

18. Nicoll and Keogh, *The Unity Game Engine and the Circuits of Cultural Software*, 70.

But this is weak evidence for Unity's own structure because it is exterior to the engine itself. The game engine does not itself enforce that raycasting be used for shooting in the same way that, say, Bitsy enforces a three-color palette. For as much as Nicoll and Keogh insist that Unity games "inherit a particular 'look and feel' from their engine," actually characterizing this look and feel, and the process by which it came to be acquired, is a challenge.[19]

In developing a solution to this problem, I want to seize upon the negativity expressed by Nicoll and Keogh's participants in the block quote. I do not believe that it is a coincidence that they were able to articulate Unity's structure in terms of affect, in terms of how it made them feel. Nor do I think that it is a coincidence that the most concrete evidence that Nicoll and Keogh give of Unity's grain comes from their most scathing participant: Terry Burdak, the lead designer of *Paperbark* (Paper House, 2018). Burdak offers several specific complaints about what the engine either would not let him do or made very challenging. For example, he was confused and frustrated by having to make a 2D user interface in the 3D game editor, describing Unity on the whole as "janky as fuck."[20] This is not a coincidence because affect and relation, feeling and structure, are inextricably intertwined: to have your feeling structured by the game engine is to feel the structures of the game engine in turn. Burdak's emotional responses to Unity revealed precisely what the engine had tried to obscure. He felt its structure, and in so feeling it, made it momentarily visible.

19. Nicoll and Keogh, 64.

20. Nicoll and Keogh, 72.

Knowledge of Forms

The rest of this chapter is dedicated to the implications that affect as relation has for critical engine studies by proposing that feeling can itself be a mode of scholarly inquiry that helps us better understand *all* engines, regardless of their investments in neutrality. In developing this proposal, I supplement my existing definition of affect as relation with the work of queer theorist Sara Ahmed to consider in more theoretical detail affect's capacity to surface structure. I then consider this capacity in practice by turning to accounts of my own encounters with game engines: moments in which I brushed up against boundaries that I did not even know existed until I was being frustrated by them, and then, suddenly, there they were. Attending to such moments, I argue, can help us understand game engines and their influences on their users and their creative work. This understanding is necessarily partial because it is specific to specific engine-user relations, but this too offers an important theoretical contribution to critical engine studies by insisting that game engines have no universal structure. Each engine is felt differently by each of its users, and each user uniquely contributes to the partial and momentary existence of an engine's structure. And this structure is only ever briefly apprehended before the affect fades and the structure disappears once more. We are, in essence, forever groping at the shape of the engine, but this is work well worth doing in order to avoid forever characterizing engines as universally freeing or unilaterally controlling.

Shapes and Surface

I want to bring in Sara Ahmed here first to acknowledge how her works, particularly *Queer Phenomenology: Orientations, Objects, Others* (2006) and *The Cultural Politics of Emotions* (2014), have influenced much of my own thinking about both queer theory and affect theory.

Additionally, Ahmed usefully supplements my theory of affect's surfacing capacities by drawing attention to *our* surfaces as well. That is, while I have considered how game engines come to have certain structures through the decisions of their designers, such as an investment in the myth of neutrality, I have not paid much attention to how our own structures (what Ahmed calls shapes) fit into the equation. This attention is required in keeping with the radical specificity of relation that I view to be so crucial to understanding engines through affect. As part of later narrativizing affective encounters with game engines, then, I must understand my own specific shape, how I came to acquire it, and how it impacted those encounters.

In *The Cultural Politics of Emotions*, Ahmed argues that bodies, both human and non, have shapes and that they acquire their shapes through the operation of power and politics upon them. For example, heteronormativity acts upon bodies to produce straight bodies, but it also acts upon spaces to produce straight spaces.[21] An all too relevant political example would be the "Parental Rights in Education" bill recently passed in Florida. This law has been dubbed by critics as the "Don't Say Gay" bill because it prohibits teaching young children about sexual orientation and gender identity.[22] While heterosexuality (and cisgender) are obviously both also sexual orientations and gender identities, it is also obvious that the terms are meant specifically to prohibit discussing queerness and transness. Thus, Florida classrooms for kindergarteners through third graders have been made straight spaces in which gayness is unspeakable.

---

21. Sara Ahmed, *The Cultural Politics of Emotion*, 2nd edition (Edinburgh University Press, 2014), https://doi.org/10.3366/j.ctt1g09x4q, 145-147.

22. Mark Joseph Stern, "The Biggest Lie About Florida's 'Don't Say Gay' Bill Erases One Crucial Word," *Slate*, March 15, 2022, https://slate.com/news-and-politics/2022/03/florida-dont-say-gay-censorship-republican-lies.html.

What's more, if both us and the spaces we occupy have shapes, there arises the question of the fit between them— such as the *mis*fit between a hypothetical queer teacher in a classroom where they are not allowed to talk about their partner. Ahmed explains fit as a matter of comfort or discomfort: either you fit well, in which case the space causes you no pain, or you do not fit well, in which case the space is, at best, mildly irritating and, at worst, actively violent. But the specific contribution of Ahmed's that I want to highlight is how comfort and discomfort relate to our perception and awareness of shape. Ahmed explains: "To be comfortable is to be so at ease with one's environment that it is hard to distinguish where one's body ends and the world begins. One fits, and by fitting, the surfaces of bodies disappear from view."[23] When we are comfortable, we lose awareness not only of the shape of our space but our own shape as well. We see ourselves as continuous with, a natural extension into, the world around us, which itself appears a rather shapeless, neutral world. If we have acquired a shape that is harmonious with the shapes of our spaces (being, perhaps, structured by the same power), we may never even realize that either of us have shapes at all.

But discomfort is surfacing; it makes shape and structure visible because we are no longer continuous with the world around us. The gaps appear.

Affective Anecdotes: Stories from Twine and Unity

Game makers also have shapes that they have acquired over time and which affect their interactions with game engines— their ability to fit within the space of the engine, their capacity to see its shape and feel its structure. To illustrate this, I want to describe, in as best detail as I can recall, two affectively-charged encounters that I have had with two engines, Twine and

23. Ahmed, *The Cultural Politics of Emotions*, 148.

Unity. This is not the most scientifically sound practice, but I have insisted throughout this work that affect as relation is specific, and the richest specificities that I can offer are derived from my own life and my experiences as an almost-computer science student. This background, particularly my experience taking a 2D engine design class, has given me a great deal of comfort with game engines. Even so, there have been many moments of discomfort and disorientation, of frustration and exhaustion, peppered throughout my limited game design experience.[24] And in such moments, the structures of my tools became, briefly, that much more understandable.

I began using Twine and Unity at the same time: in the summer of 2018, at a National Science Foundation Research Experiences for Undergraduates (REU) site at North Carolina State University. My lab, Dr. Arnav Jhala's Interactive and Visual Narrative Lab, was working on a project to prototype a game that combined branching narrative storytelling with location-based data. The idea was that the game would take players all over historic Raleigh, North Carolina and allow them to play out an interactive story tied to specific locations. For about a month or so, my fellow REU student and I focused on narrative prototyping in Twine, a hypertext authoring tool popular for creating branching interactive fiction. About a third of the way through the program, though, we were asked to pivot to working in Unity to lay the groundwork for incorporating location data with the stories that the history students on the project would continue developing in Twine.

I had a much easier time learning Unity than learning Twine. I came to Unity having already acquired its shape from my engine design class, and I found that many of the principles and concepts (if not the specifics) translated over. I already understood, for example, the basics

24. There have also, I am sure, been moments of joy and other positive affects, but they are not so clearly marked in my memory; certainly, more rigorous methods are needed for exploring affect as relation, but this will function as proof of concept.

of component versus inheritance-based design, the necessity of doing physics calculations in a fixed update call, the difference between game space and world space, the function of a viewport, etc. Even more fundamentally, when I encountered something that I did not know in Unity, I knew precisely how to navigate its documentation to find a solution. I had a very particular expectation for what game engine documentation should look like, and the Unity scripting API and engine manual fulfilled those expectations perfectly.

Twine's documentation did not. When I looked for the first time at the documentation for Harlowe, the story format (Twine-specific mini programming language) that I was using, I was shocked by what I saw. Programming documentation is written in a neutral, informational tone and follows a rigid structure, so seeing Harlowe's documentation blend formal information with informal description and casual instruction was a surprise. To illustrate this disparity in tone, consider how the official documentation for the programming language that Unity uses, C#, describes the dictionary data structure: "Represents a collection of keys and values."[25] By contrast, Harlowe describes its equivalent like this: "Creates a datamap, which is a data structure that pairs string names with data values. You should provide a string name, followed by the value paired with it, and then another string name, another value, and so on, for as many as you'd like."[26] Never before had I seen a direct address in a program documentation before, and certainly not alongside words like "should" and "like".

Lost without the clinical and comprehensive precision of a standard programming API, I struggled immensely in my first few weeks of using Twine. I could not understand how its parts

25. "Dictionary<TKey,TValue> Class (System.Collections.Generic)," accessed May 10, 2022, https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.dictionary-2.

26. Leon Arnott, "Harlowe 3.2.3 Manual," October 22, 2021, https://twine2.neocities.org.

fit together, particularly as I desired to do more complicated things with it. I wanted methods, classes, objects, for-loops, static arrays! For as much freedom as Twine had historically allotted others, particularly queer non-programmers who took it up around 2012 for its accessibility, structure is not felt universally. What had been empowering for them was a terrible misfit for me, and I felt Twine more strongly as a result. I had been too severely shaped by computer science, and so I bumped up against Twine's shape—its structures and limitations—at each and every turn.

This came to a head when I once spent upwards of twelve hours trying to do something that seemed to me very simple: generate new links on one page that would be revisited multiple times. The idea was that players could visit other pages, make choices there, and then come back to a central "hub" page and see new links depending on the choices that they had made. In my mind, this was a simple matter of having a list that kept track of all the links I wanted to show, then running through that list and displaying all of the links inside.

I will spare you the technical details to say that this did not work. And not only did it not work, but my attempt to work around it did not work. Nor, indeed, my attempt to work around that. There were aspects of Twine's shape that I had not previously noticed, but I felt them again and again that day. I spent hours trying to outmaneuver Twine's boundaries, only to smack up against them with each new error message or each dead-end within the documentation. More than frustrated or disoriented, I was obsessed, and I kept at it late into the night until I finally came up with the ludicrous workaround of putting the links in their own tidy segregated passages and running through a list of passage names instead. At least the central hub passage was finally as elegant as I desired, if not resembling in the slightest what a normal, text-heavy, code-light Twine passage tends to look like.

The strangest discovery, however, and the one that most concretely links affective experience with Twine's structure, came in the form of one particular error message that I encountered: "*(a:)*'s 1st value, ?test (a hook name), is not valid data for this macro. The *(a:)* macro must only be given anything" (fig. 7). Translating some of Harlowe's lingo, what this error is saying is that I cannot make a list that includes the object called "?test". The reason why I cannot do this is because "?test" is not the correct type of data. Type errors are common enough in programming since certain operations require certain forms of data. If I have a program that alphabetizes a list of words, for example, I cannot pass it a list of numbers, just like I couldn't multiply the words in that list together. Both of these would result in type errors. What is baffling about this type error, then, is that if *(a:)* can be given *anything*, then a type error should never occur. I was trying to give it a piece of data that, evidently, was not included in Twine's definition of "anything".

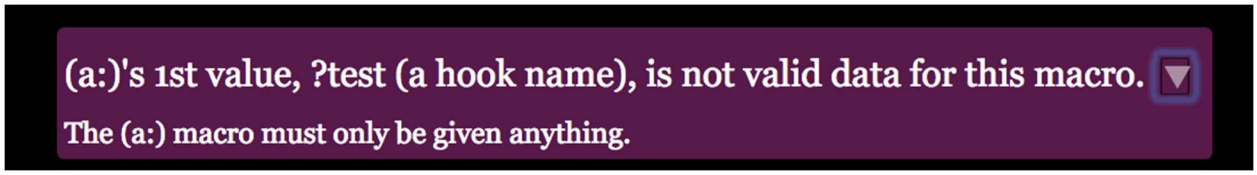


Figure 7. Screenshot of a Harlowe error message. Taken by author.

Retroactively parsing the frustration, disorientation, and discomfort that this specific moment provoked, I interpret this as an instant in which the in-between-ness between myself and Twine became visible. Rather than disappearing into seamlessness, I touched Twine and Twine touched me such that I became angry and Twine's surfaces surfaced. A detail of its contour which could not be read from its documentation nor distantly appraised from its interface became apparent.[27]

27. For those curious about the technical specificity of this detail, the way it shakes out is that hooks are *markup*, not data. The point of confusion was that hooks are described in the

Of course, not all affective encounters illuminate such concrete details of or singular limits in a game engine's structure. This brings me to the second anecdote, which involves Unity. This affective encounter is not so tightly bounded as the first, nor contributes so directly to the project of knowing Unity's structures except, perhaps, to point out that our relationships with our tools are not static, fixed, or isolated from the broader context of our encounter. What is once comfortable can become uncomfortable because affect and relation are both processes, ceaselessly reworking one another.

This incident came a few months later, after the summer program had ended and I had returned to Brown University for the fall semester. That semester, I took the class that changed the trajectory of my life forever: a queer games studies course taught by Dr. Teddy Pozo. The class' assignments had included one written assignment, one game design assignment (which, coincidentally, had to be done in either Twine or Bitsy, and I reused the same Twine code I had labored to build over the summer!), and a final project that could be either. For the final, I decided to make a game about the liminality of gender expression in queer women, paired with an aesthetic of minimalist abstraction that, in retrospect, may not have served my artistic vision but certainly saved me the worry of designing assets. One day, while I was working on it, I was trying to write a method that would dynamically alter the color of the background to evolve from black and white speckles to a gradient of pink, blue, and purple. This would reflect the evolution of the player's gender expression, which they could freely change by pressing the <J> key for fem/pink and the <L> key for masc/blue. As an artistic endeavor, it was, I admit in retrospect, heavy-handed.

'Coding' section of Harlowe's documentation alongside macros and variables, which *are* data and can be added to lists.

As a technical one, though, it was impossible.

Again, the technical details of this process are secondary to what I felt that afternoon as, just in the case of the Twine anecdote, something I thought would be easy to implement took longer and longer and I grew more and more frustrated— again, obsessively so. I didn't eat, I didn't go to the bathroom, I sat cramped in an armchair in the basement of the Sciences Library until my joints ached and my back cracked, and I kept throwing myself at Unity's boundaries. I kept trying to brute force my way through the code and the math behind it, increasingly irate over the apparent disconnect between my understanding of how the background's texture component should work and how I saw my code working when I ran the game.

I wasted the whole afternoon there until I was too hungry and too thirsty to be trusted with regulating my emotions, and only then did I realize that my mental model from hours ago had been correct. I had just tricked myself into thinking it wasn't. What I wanted to do was not impossible, nor did it require the convoluted workaround that Twine did. In fact, it worked exactly as I expected, and seemingly only I had stood in the way of Unity enabling and facilitating my free artistic expression.

I went home that night and I cried. I was even so shaken up, feeling so stupid for having wasted so long on something I had known how to do before this miscommunication with my engine, that I called my mom to vent to her.

The takeaway here is not as simple as an exceptionally cryptic error message. Indeed, if my personal knowledge of Unity was advanced at all, it was through realizing that something was *possible*, not impossible— and yet, because of the larger circumstances of this realization, the response it elicited was not a purely positive one. We still could not fit together because I

was too distressed to sink back into Unity's structure. Its surfaces, and mine, remained apparent until I finally got some food in my stomach and went to bed.

This anecdote serves as evidence that the knowledge project of feeling structure will not be a clean, straightforward one in which emotional experience correlates one-to-one with realizing a new limit or possibility of an engine. Indeed, as Anable reminds us, the vast majority of these emotional experiences may be far smaller, less personally significant. But even subtle affects represent an instance of relation in which we can feel structure and contribute to a forever-partial, forever-specific understanding of the game engine.

Escaping the Freedom/Control Binary

I want to end my thesis by returning to a quote I mentioned in the introduction in order to demonstrate the importance of attending to affect in critical engine studies. From Eric Freedman's *The Persistence of Code in Game Engine Culture*:

> For queer game artists, the simpler mechanics (and alternative game languages) of Twine and GameMaker seem to facilitate qualities such as empathy, community and communicative openness. But we need to avoid understanding even the value of engine choice in such binary terms, as freedom, expressivity, multivocality, and queerness are in every case delimited by software mechanics.[28]

At the time, I argued that this was an ahistorical take that briefly referenced but did not reckon with the affective complexity of engine choice for the early queer games avant-garde. Now, I want to focus not on engine choice but on the claim that freedom, expression, and queerness are fundamentally delimited by the engine. I understand how Freedman has come to this argument. I am even sympathetic to it, born as it is out of a preoccupation that the game engine's influence

28. Eric Freedman, *The Persistence of Code in Game Engine Culture* (Taylor and Francis, 2020), 163.

on the game production pipeline has gone unaccounted for. For more mainstream engines, this takes the form of the myth of the neutral game engine, discussed earlier, but there is a variant of this line of thought in the discourse around smaller, more structured tools as well. This was particularly espoused by those who uncritically celebrated Twine as a means of expressive empowerment for queer game makers while ignoring their enduring economic and social precarity.[29] We certainly cannot understand Twine solely in terms of expressivity or queerness. I agree with Freedman that we must develop an understanding of engines beyond freedom and attend to the ways that they do, in fact, sometimes limit creative expression.

What I object to is the scope of Freedman's argument: the notion that expressivity is in *every case* delimited by the game engine. In his effort to escape one end of the problem, Freedman seems to me to have swung uncritically into the other, from universal freedom to unilateral control. If my thesis has insisted on anything, it is the reciprocity and specificity of the engine-user relationship. It is that we do not only have our feelings structured but feel through structure, and that the feelings and structures in both directions are constituted by everyone involved. Yes, game engines have limits. Every tool does. But there is also always a possibility for negotiation, as Andi McClure articulated at the start of this thesis— a chance to resist having our fragile emotions destroyed and our creativity hate-fucked.[30]

---

29. Alison Harvey, "Twine's Revolution: Democratization, Depoliticization, and the Queering of Game Design," *G.A.M.E. (Reggio Calabria)*, 2014.

30. Andi McClure, "Algorithms, Accidents, and the Queerness of Abstraction," interview by Bo Ruberg, *The Queer Games Avant-Garde: How LGBTQ Game Makers Are Reimagining the Medium of Video Games* (Duke University Press, 2020), 75.

Porpentine, "Interview with Porpentine, author of howling dogs," interview by Emily Short, *Emily Short's Interactive Storytelling* (blog), November 23, 2012, https://emshort.blog/2012/11/23/interview-with-porpentine-author-of-howling-dogs/. Archived at https://web.archive.org/web/20170312075143/https://emshort.blog/2012/11/23/interview-with-porpentine-author-of-howling-dogs/, accessed May 7, 2022.

If the influence of the game engine on its users and their creative processes is to be a central object of inquiry for critical engine studies, neither of these binary positions will do. We might be well-served, then, by returning to John Romero's original definition of the game engine: "the engine 'is the heart of the car, this is the heart of the game.'"[31] And what a heart it is, as complex in structure and feeling as our own. When we touch it, it touches us in return. Rather than suggesting that one touch is more powerful than the other, more influential than the other, I propose that critical engine studies use affect as relation to evaluate them both. And, what's more, to attend to all the relational complexity, reciprocity, messiness, entanglement, positivity, negativity, neutrality, affectivities beyond binary valences, irrationalities, and—potentially—even queerness that can arise when two hearts collide.

Yang, "KILL UNITY."

31. Romero, quoted in Lowood, "Game Engines and Game History," 186.

BIBLIOGRAPHY

Ahmed, Sara. *Queer Phenomenology: Orientations, Objects, Others*. Durham: Duke University Press, 2006.

———. *The Cultural Politics of Emotion*. 2nd edition. Edinburgh University Press, 2014. https://doi.org/10.3366/j.ctt1g09x4q.

Anable, Aubrey. *Playing with Feelings: Video Games and Affect*. Minneapolis: University of Minnesota Press, 2018. https://doi.org/10.5749/j.ctt20mvgwg.

Anthropy, Anna. *Rise of the Videogame Zinesters: How Freaks, Normals, Amateurs, Artists, Dreamers, Drop-Outs, Queers, Housewives, and People Like You Are Taking Back an Art Form*. New York: Seven Stories Press, 2012.

Barnett, Fiona, Sach Blas, micha cárdenas, Jacob Gaboury, Jessica Marie Johnson, and MARGARET RHEE. "QueerOS: A User's Manual." In Debates in the Digital Humanities 2016, edited by Matthew K. Gold and Lauren F. Klein, 50–59. University of Minnesota Press, 2016. https://doi.org/10.5749/j.ctt1cn6thb.8.

Bernardi, Joe. "Choose Your Own Adventure-Maker: Twine and the Art of Personal Games." Vice, February 19, 2013. https://www.vice.com/en/article/xyyp9a/twine-and-the-art-of-personal-games.

Boehner, Kirsten, Rogério DePaula, Paul Dourish, and Phoebe Sengers. "How Emotion Is Made and Measured." *International Journal of Human-Computer Studies* 65, no. 4 (2007): 275–91. https://doi.org/10.1016/j.ijhcs.2006.11.016.

Brice, Mattie. "Death of the Player." Mattie Brice (blog), October 29, 2013. http://www.mattiebrice.com/death-of-the-player/.

Broekens, Joost. "Emotion Engines for Games in Practice: Two Case Studies Using Gamygdala," 790–91. IEEE, 2015. https://doi.org/10.1109/ACII.2015.7344662.

Broekens, Joost, Eva Hudlicka, and Rafael Bidarra. "Emotional Appraisal Engines for Games." In *Emotion in Games*, 215–32. Socio-Affective Computing. Cham: Springer International Publishing, 2016. https://doi.org/10.1007/978-3-319-41316-7_13.

Dror, Otniel E. "Counting the Affects: Discoursing in Numbers." *Social Research* 68, no. 2 (2001): 357–78.

Ellison, Cara. "Anna Anthropy and the Twine Revolution." *The Guardian*, April 10, 2013, sec. Technology. https://www.theguardian.com/technology/gamesblog/2013/apr/10/anna-anthropy-twine-revolution.

Freedman, Eric. *The Persistence of Code in Game Engine Culture*. Taylor and Francis, 2020.

Ghandeharioun, Asma, Szymon Fedor, Lisa Sangermano, Dawn Ionescu, Jonathan Alpert, Chelsea Dale, David Sontag, and Rosalind Picard. "Objective Assessment of Depressive Symptoms with Machine Learning and Wearable Sensors Data." In *2017 Seventh International Conference on Affective Computing and Intelligent Interaction (ACII)*, 325–32, 2017. https://doi.org/10.1109/ACII.2017.8273620.

Gilleade, Kiel Mark, Alan Dix, and Jen Allanson. "Affective Videogames and Modes of Affective Gaming: Assist Me, Challenge Me, Emote Me." In *Proceedings of DiGRA*, 1–7, 2005.

Harvey, Alison. "Twine's Revolution: Democratization, Depoliticization, and the Queering of Game Design." *G.A.M.E. (Reggio Calabria)*, 2014.

Hassenzahl, Marc. *Experience Design: Technology for All the Right Reasons*. Vol. 8. Synthesis Lectures on Human-Centered Informatics. San Rafael, Calif. (1537 Fourth Street, San Rafael, CA 94901 USA): Morgan & Claypool Publishers, 2010. https://doi.org/10.2200/S00261ED1V01Y201003HCI008.

———. "User Experience (UX): Towards an Experiential Perspective on Product Quality," 11–15. IHM '08. ACM, 2008. https://doi.org/10.1145/1512714.1512717.

Healey, J., and R.W. Picard. "StartleCam: A Cybernetic Wearable Camera." In *Digest of Papers. Second International Symposium on Wearable Computers (Cat. No.98EX215)*, 42–49, 1998. https://doi.org/10.1109/ISWC.1998.729528.

Hudlicka, Eva. "Affective Game Engines: Motivation and Requirements," 299–306. FDG '09. ACM, 2009. https://doi.org/10.1145/1536513.1536565.

Kaptein, Frank, and Joost Broekens. "The Affective Storyteller: Using Character Emotion to Influence Narrative Generation." In *Intelligent Virtual Agents*, 352–55. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015. https://doi.org/10.1007/978-3-319-21996-7_38.

Klepek, Patrick. "Unity Workers Question Company Ethics As It Expands From Video Games to War." *Vice*, August 23, 2021. https://www.vice.com/en/article/y3d4jy/unity-workers-question-company-ethics-as-it-expands-from-video-games-to-war.

Kushner, David. *Masters of Doom: How Two Guys Created an Empire and Transformed Pop Culture*. Westminster: Random House Publishing Group, 2003.

Leys, Ruth. "The Turn to Affect: A Critique." *Critical Inquiry* 37, no. 3 (March 2011): 434–72. https://doi.org/10.1086/659353.

Lowood, Henry. "Game Engines and Game History." *History of Games International Conference Proceedings*, edited by Carl Therrien, Henry Lowood, and Martin Picard, 179-198. Kinephanos, January 2014.

Lu, Hong, Denise Frauendorfer, Mashfiqui Rabbi, Marianne Mast, Gokul Chittaranjan, Andrew Campbell, Daniel Gatica-Perez, and Tanzeem Choudhury. "StressSense: Detecting Stress in Unconstrained Acoustic Environments Using Smartphones," 351–60. UbiComp '12. ACM, 2012. https://doi.org/10.1145/2370216.2370270.

Lu, Hong, Denise Frauendorfer, Mashfiqui Rabbi, Marianne Schmid Mast, Gokul T. Chittaranjan, Andrew T. Campbell, Daniel Gatica-Perez, and Tanzeem Choudhury. "StressSense: Detecting Stress in Unconstrained Acoustic Environments Using Smartphones." In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, 351–60. UbiComp '12. New York, NY, USA: Association for Computing Machinery, 2012. https://doi.org/10.1145/2370216.2370270.

Massumi, Brian. “The Autonomy of Affect.” *Cultural Critique* 31 (1995): 83–109.

Negini, Faham, Regan L. Mandryk, and Kevin G. Stanley. “Using Affective State to Adapt Characters, NPCs, and the Environment in a First-Person Shooter Game,” 1–8. IEEE, 2014. https://doi.org/10.1109/GEM.2014.7048094.

Nicoll, Benjamin, and Brendan Keogh. *The Unity Game Engine and the Circuits of Cultural Software*. 1st ed. 2019. Cham: Springer International Publishing, 2019. https://doi.org/10.1007/978-3-030-25012-6.

Norman, Don. “Emotion & Design: Attractive Things Work Better.” *Interactions* 9, no. 4 (July 1, 2002): 36–42. https://doi.org/10.1145/543434.543435.

Norman, Donald A. “Affordance, Conventions, and Design.” *Interactions* 6, no. 3 (May 1, 1999): 38–43. https://doi.org/10.1145/301153.301168.

———. *The Design of Everyday Things*. Revised and Expanded edition. New York: Basic Books, A Member of the Perseus Books Group, 2013.

Ortony, Andrew, Gerald L. Clore, and Allan Collins. *The Cognitive Structure of Emotions*. Cambridge: Cambridge University Press, 1988. https://doi.org/10.1017/CBO9780511571299.

Picard, Rosalind W. *Affective Computing*. Cambridge, Mass: MIT Press, 1997.

Popescu, Alexandru, Joost Broekens, and Maarten van Someren. “GAMYGDALA: An Emotion Engine for Games.” *IEEE Transactions on Affective Computing* 5, no. 1 (2014): 32–44. https://doi.org/10.1109/T-AFFC.2013.24.

Porpentine. “Creation Under Capitalism and the Twine Revolution.” Nightmare Mode [Archived], November 25, 2012. https://nightmaremode.thegamerstrust.com/2012/11/25/creation-under-capitalism/.

Ruberg, Bo. “The Precarious Labor of Queer Indie Game-Making: Who Benefits from Making Video Games ‘Better’?” Edited by Greig de Peuter and Chris J. Young. *Television & New Media* 20, no. 8 (2019): 778–88. https://doi.org/10.1177/1527476419851090.

———. *The Queer Games Avant-Garde: How LGBTQ Game Makers Are Reimagining the Medium of Video Games*. Durham: Duke University Press, 2020.

Sano, Akane, and Rosalind W. Picard. “Stress Recognition Using Wearable Sensors and Mobile Phones,” 671–76. IEEE, 2013. https://doi.org/10.1109/ACII.2013.117.

Schreier, Jason. “Crunch Time: Why Game Developers Work Such Insane Hours.” Kotaku Australia, May 16, 2015. https://www.kotaku.com.au/2015/05/crunch-time-why-game-developers-work-such-insane-hours/.

Sedgwick, Eve Kosofsky, and Adam Frank. “Shame in the Cybernetic Fold: Reading Silvan Tomkins.” *Critical Inquiry* 21, no. 2 (1995): 496–522. https://doi.org/10.1086/448761.

Seigworth, Gregory J., and Melissa Gregg. “An Inventory of Shimmers.” In *The Affect Theory Reader*, edited by Melissa Gregg and Gregory J. Seigworth, 1–26. New York, USA: Duke University Press, 2010. https://doi.org/10.1515/9780822393047-002.

Short, Emily. “Interview with Porpentine, Author of Howling Dogs.” *Emily Short’s Interactive Storytelling* (blog), November 23, 2012. https://emshort.blog/2012/11/23/interview-with-porpentine-author-of-howling-dogs/. Archived at https://web.archive.org/web/20170312075143/https://emshort.blog/2012/11/23/interview-with-porpentine-author-of-howling-dogs/, accessed May 7, 2022.

Somerville, Siobhan. “Queer.” In *Keywords for American Cultural Studies, Second Edition*, edited by Glenn Hendler and Bruce Burgett, 203–7. New York, UNITED STATES: New York University Press, 2014. http://ebookcentral.proquest.com/lib/uci/detail.action?docID=1853975.

Stern, Mark Joseph. “The Biggest Lie About Florida’s ‘Don’t Say Gay’ Bill Erases One Crucial Word.” *Slate*, March 15, 2022. https://slate.com/news-and-politics/2022/03/florida-dont-say-gay-censorship-republican-lies.html.

Tekinbaş, Katie Salen, and Eric Zimmerman. *Rules of Play: Game Design Fundamentals*. Cambridge, Mass: MIT Press, 2003.

Vujic, Angela, Stephanie Tong, Rosalind Picard, and Pattie Maes. “Going with Our Guts: Potentials of Wearable Electrogastrography (EGG) for Affect Detection.” In *Proceedings of the 2020 International Conference on Multimodal Interaction*, 260–68. New York, NY, USA: Association for Computing Machinery, 2020. https://doi.org/10.1145/3382507.3418882.

Wilchins, Riki. *Queer Theory, Gender Theory: An Instant Primer*. New York: Magnus Books, 2014.

Yang, Robert. “KILL UNITY; WE ARE ENGINES.” itch.io, 2018. https://radiatoryang.itch.io/kill-unity.

———. “We Are Drugs; Speculative Dev Tools and Psychedelic Hologram Futures.” *Radiator Design Blog* (blog), February 18, 2015. https://www.blog.radiator.debacle.us/2015/02/we-are-drugs-speculative-dev-tools-and.html.

Yuda, Kaori, Maxim Mozgovoy, and Anna Danielewicz-Betz. “Creating an Affective Fighting Game AI System with Gamygdala,” 1–4. IEEE, 2019. https://doi.org/10.1109/CIG.2019.8848031.