de-generate

de-generate

Kahani Ploessl

ABSTRACT

de-generate is an experimental fighting video game that explores how generative art may permeate as not only a technological aesthetic, but an essential component in gameplay and game evolution. This is primarily realized through generative characters - shape shifting avatars that maximize the capabilities of the digital realm. Characters and their core structural code may be intercepted and facilitated by the player, resulting in new, adaptive, and expressive experiences.

1. INTRODUCTION

Digital reality is shaped by the interface - the nexus between raw computation and human perception. While each may interact and influence the other, their collaboration situates the machine and operator firmly at opposite ends up the screen. The division is explicit, and the resulting interface performs predictably as mediator and translator for user needs.

This pre-established separation of artificial and human processing, restrains digital reality from realizing its full potential. Digital reality and their modes of manifestation are the next dimension - an extension of the physical plane to realize oneself in complete immateriality. As an immensely powerful tool, code capable of instantiating completely self-contained and self-sustaining systems. In pursuit of digital autonomy and expression, redefining interfaces so users may tap into the power of code will play an important role in redefining how digital reality is transversed. Interfaces that push backend capabilities to the forefront will empower users to shape their own digital experiences with flexibility, sovereignty, and longevity.

Considering the enormous task, *de-generate* takes the first step by positing a new and experimental dynamic between code, data, and avatars. As the surrogate for digital existence, the very form and movements of *de-generate's* characters can be computationally disrupted and exploited by real-time events and player controls. Players may harness data and code directly, challenging the interface in unexpected and evolutionary ways.

In realizing this experimental concept, *de-generate* combines generative approaches and video game environments as its basis.

1.1 Generative Art - The Approach

Generative art is a form characterized by experienced emergence. Overtime the piece reveals itself according to a predefined logic system that responds to external factors. While not a practice exclusive to technology, generative art expressed through code is the perfect union. Not only is code explicitly tied to function, but its fundamental components of read, process, and execute are, in essence, what collectively makes generative art. As such, manifestations of generative art assimilate a sense of coded or digital existence. Upon instantiation, generative art unfolds to the combined wills of predestined rules and unpredictable participators. Tandem relationships between the what and the how - results and mode of fruition, machine and operator, backend code and frontend interface - are exposed for observation and experimentation. What emerges is pure evolution, philosophically profound and experimentally informative.

Dissecting these notions further, *de-generate* exhibits principles of generative art in not only the game as a whole, but the individual characters themselves. More specifically, characters are built as contained generative systems. Instead of being coded to operate, characters are coded to exist. The distinction here may seem obscure, but upon starting the game, its ramifications become immediately apparent. *de-generate's* characters are coded so that their fundamental components may be unpredictably altered alongside player input. As the larger game comes in to play, this generativeness is then immediately compounded by inter character dynamics even more so unknown. The characters and their coded constructs are merely capable of setting the scene. What will emerge cannot be predicted, and will be entirely a product of player choices.

1.2 Video Games - The Medium

The purposeful decision to explore generative techniques and innovative digital identities within a video game plays an integral role. Video games, as an interface, exhibits a comprehensive system of interdisciplinary components and real-time participation. As such, the generative themes of *de-generate* can permeate into various aspects of the game environments (aesthetic shaders, maps, character designs) no matter how small or significant.

Interestingly, despite the considerable overlap that exists between generative art and video games, these technological mediums have rarely collaborated. Each involves real-time collaboration between machine and operator actions as a means of progression or evolution. What seems to fundamentally distinguish the two is that while generative art relinquishes control in pursuit of emergence, video games rely on a highly structured frameworks and predefined game beats to tailor specific gaming experiences. Diverging from traditional form, *de-generate* explores how generative discovery may replace simulated gaming dynamics with actual, erratic computation - realtime manipulation of game objects at the structural level.

Although video games may appear as rigid structures, their subsequent player behaviour, social dynamics, and cultural relevancy are, by no means, devoid of evolution. This is known as the meta game, a phenomena in which players adopt strategies and tactics outside the core gameplay mechanics. Often involving exploitation of any and all weaknesses in the interface as a means for attaining some goal or advantage, the meta game evolves endlessly alongside continual player engagement. As long as the video game can sustain player enjoyment, a meta game will always emerge. In this sense, *de-generate* incorporates generative techniques by relinquishing controlled game dynamics to the emergence of a meta game. As generative art is intended to be discovered, *de-generate* is designed to fully accept how players embrace the system, even when that entails the exploitation of what may be 'wrong' with the interface.

Combining these perspectives on unpredictable gaming and meta game emergence, *de-generate's* video game interface can be viewed as the optimal testing environment. As a digital construct, the generative experiment may be repeated endlessly in a vacuum. All factors may be fully accounted for, and the resulting data may be precisely recorded for further analysis. Furthermore, as a fighting style game, *de-generate* firmly places generative characters as its sole focus.

2. THEORETICAL CONTEXT

The theoretical context review identifies current arguments concerning generative art, the role of code as a design "contributor", and the importance of real time generation in user interactions.

2.1 Ten Questions Concerning Generative Art

Ten Questions Concerning Generative Art explore several probing questions into the implications of generative art. The most relevant and interesting question concerns "Question 4: What New Kinds of Art Does the Computer Enable" (McCormack et al, 137). Here McCormack et al explains the potential in harnessing computational power in manifesting complex interactions. "Computer simulations allow the building of 'model worlds' that permit the vivid realization and expression of ideas and complex scenarios that are impossible in other media" (McCormack et al, 137). Continuing with this notion, generative art is more than an autonomous process. It is the observation of a wholistic logic system that dynamically adapt to new scenarios and human input.

2.2 Emergence and Generative Art

Gordon Monro's article proposes an evolved definition for generative art as an experience of emergence. Endeavouring to define the complex, Monro describes effective generation as an amalgamation of surprise-wonder-mystery-autonomy. Emergences is a process to be observed, a life of its own that surpasses mere novelty through imaginative contexts and adaptive interactions. This interpretation stresses a distinction between generative art that is simply unknown and what is wondrously enigmatic. Generative art has the potential for real time play, and can be far more creative than algorithmic visualizations.

2.3 The Generative Process, Music Composition and Games

The Generative Process, Music Composition and Games, by Nyssim Lefford, explores how video games provide an excellent environment for analyzing the generative process. Here Lefford argues that video games are capable of establishing a controlled space for user testing, one with clear mechanics, constraints, and user goals.

The context of this article involved generative music games in which user's could compose a series of samples. Interestingly, throughout this analysis Lefford extends the definition of generative processes to that of human perception and decision making. A prominent reference to this human and computational collaboration involves the concept of generative strategy, in which templates guide user experience and expression. Lefford explains that "generative games that make use of a structural template or specify a referent not only make the salience more apparent to both creator and non-creator but do so while preserving a realistic generative scenario" (131, Lefford).

Applying this notion to Generative Character Design, the characters themselves act similarly to templates. Not only does each character employ a different generative approach, but it's playability also provides an interactive guide into how its processes may be utilized. Furthermore, deciding which of these templates to use introduces a strategic element. Players may choose which generative function to wield.

3. METHODOLOGY

With generative art being a highly iterative process, *de-generate* employs this methodology with a cyclical development of prototyping, testing, and refinement. Furthermore, considering the experimental subject matter, iterative monitoring of what manifests within the game environment is essential in understanding how generative techniques may be used to redefine digital interactions.

3.1 Iterative Character Experimentation

While the iterative model typically refers to active user testing, iterative experimentation through the development of generative characters was equally essential. This primarily involved back and forth assessment of a generative techniques and their functional viability at each level of production: abstract ideation, initial 3D formation, base animations, generative moves, and inter character dynamics.

Currently, *de-generate* has completed phase one development of its generative characters. At this stage, each character embody employs a conceptual own unique take on generative , and are realized through a collaboration between simulated and computational effect. Characters have been coded with a mixture of standard and generative moves that dynamically react to one another.

3.2 User Testing

User testing will begin at GradEx 2023. Taking advantage of the opportunity to have many people with varying gaming familiarity, *de-generate's* effectiveness in delivering generative experiences and player enjoyment may be thoroughly gauged. Considering that GradEx is an event, user testing will be conducted causally, and will follow a line of questioning indicated on the following page.

1. How do players interpret generative motion?

   - Are players adapting to gameplay?
   - What characters are players choosing?
     - Does a trend emerge?
   - Are players having fun? Do they want to replay?

2. Are the controls adequately animating the character?

   - What is being fulfilled, what isn't?
   - Can the generative functions go further?

3. What meta game emerge between characters?

   - How are the characters interacting?
   - Do glitches or disparities occur?
   - Are certain characters vulnerable to others?
     - Should this be compensate for, or adopted as strategy?
   - Are shortcoming in the game environment being exploited by players?

4. PROJECT DEVELOPMENT

4.1 Game Environment and Character Conceptualization

The core game environment was created with Unity and fulfils the classic fighting game archetype with a character selection menu, playable opponents, fighting maps, reactive health bars, and an end game state.

Characters were first conceptualized through abstract ideations and sketch work. These initial sketches visualized potential motions or states with vague ideas on how they may generatively perform. Early generative explorations had inspired enough confidence for ideas to ruminate in the project's central theme of new digital emergence. Moving on, characters were then formed and animated in Blender with base movements including: idle, mobility (jump, run), attacks, and damage states. Once uploaded to unity, each character followed relatively standard frameworks for input controls, animation controllers, and collider events. Establishing the game's interconnected environment of objects, colliders, and controlling scripts early on allowed subsequent experimentations to undergo immediate implementation and continuous testing.

It was within this constructed environment where generative deep dives could take place. When approaching each character, the goal was to discover how their initial conceptualization may be expressed by generative motions. Furthermore, how could the pre-established template be disrupted while still maintaining a functional role within *de-generate's* framework.

4.2 Dynamic Shaders

Dynamic shaders in Unity were the first method explored due to its clearest visual connection to generative art. This highly parallelized rendering tool allows object properties such as colour, texture, and vertex distribution to be exaggerated and triggered by in game events. Coded in HLSL, shaders are generally comprised of vertex shaders, fragment shaders, or some combination of the two. The former manipulates shape properties by applying algorithmic motion to all vertices in a mesh. Fragment shaders, on the other hand, apply surface texture and patterns. These varying effects are illustrated by *de-generate's* initial explorations in Figure 2.

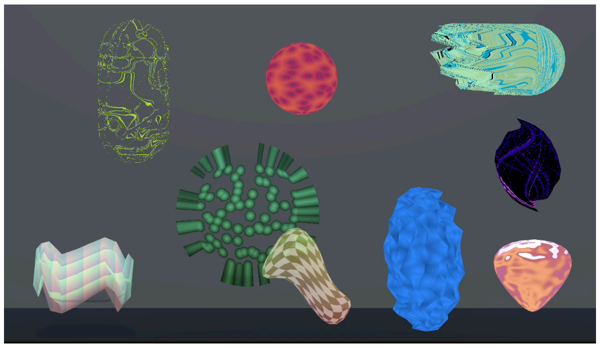


Figure 2. Initial Shader Experiments

While dynamic shaders can respond to input controls and in game events, their generative influence stop at surface level aesthetics. Since shader are optimized for displaying graphics efficiently across a material, building functions in relation to quantifiable data is not possible. For example, colliders cannot be updated or made to respond to shaders. Therefore, shaders were used for amplifying generative themes and other techniques in backgrounds, fighting maps (Figure 3), and character materials (Figure 4 and 5). These shaders are vital in conveying a sense of fluidity between digital objects and character identities.

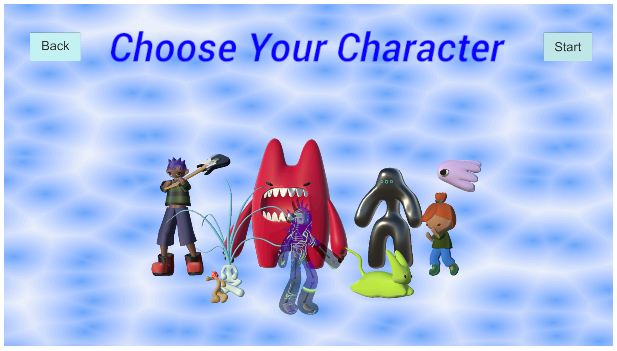


Figure 1. Character Selection Menu. Left to right: rockerChic, mushGang, hellBear, plasMan, slimeCat, metalMan, tulip, swarmOfSquids

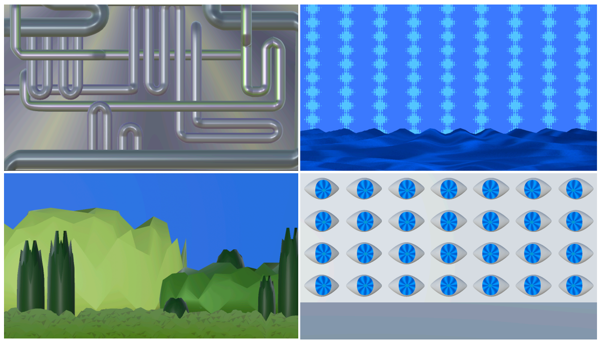


Figure 3. Map backgrounds with shaders, titled renderPipeline, waterFall, shaderGarden, and watchfulEyes

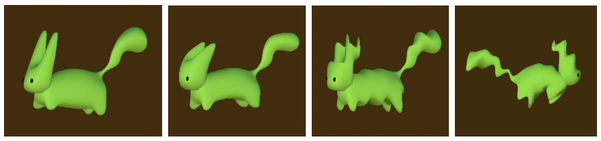


Figure 4. slimeCat vertex shader

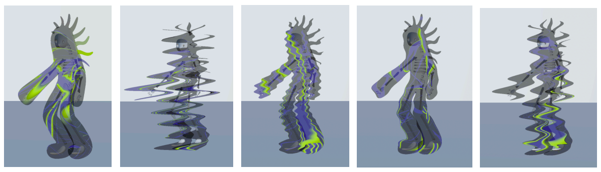


Figure 5. plasMan vertex and fragment shader

4.3 Mesh Deformations

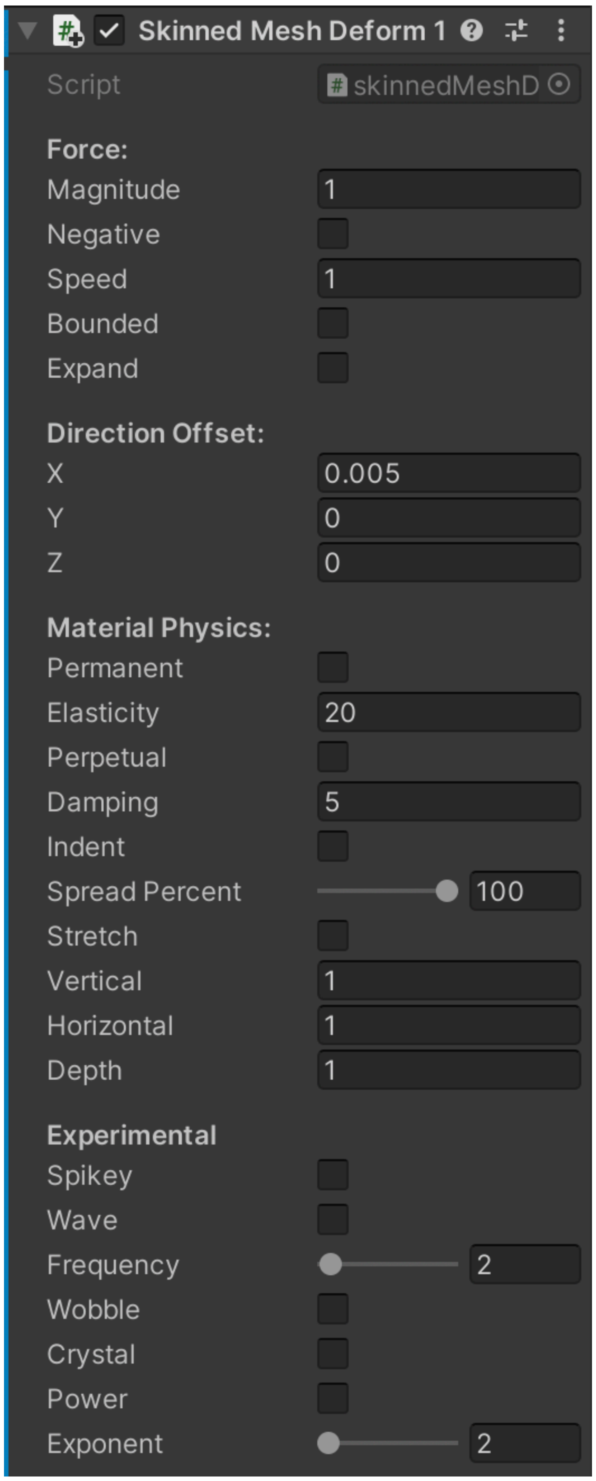
The most successful generative approach involved mesh deformations, a process in which an artificially instantiated force is algorithmically applied to each and every vertex in a mesh and thereby, the mesh collider. Applications of this generative technique have considerable variability and can range from simulating material physics to embodying entirely experimental generative motions. A template C# script and its public variables, as seen in Figure 6, was coded to accommodate flexible usage and explorations. Here variables fall into three main categories: force properties, material physics, and experimental.

Force:
- Vector data: Magnitude, Direction, Speed
- Negative: sets wether the force attracts or repels vertices
- Bounded: sets deformation limit
- Expand: set wether the force affects furtherest or nearest vertices

MateriaL Physics:
- Elasticity: how quickly vertices return to their origin
- Permanent: sets elasticity to 0, the vertices do not return
- Dampening: how quickly the velocity returns to 0
- Perpetual: sets dampening to 0, the velocity continues
- Spread Percent: how localized the deformation is
- Indent: sets the material as brittle
- Stretch: additional material stretching or compressing

Experimental:
- Spikey: effect 0-50% of vertices with even distribution
- Wave: oscillate displacement across the object's surface
- Wobble: oscillates velocity across the object's surface
- Crystal: experimental effect transforming mesh into appearing geometric
- Power: exponential displacement

When developing characters with mesh deformations, this template script was adapted to feature some combination of the above variables in conjunction with controller inputs and collider events. Furthermore, deformations were designed as thoughtfully extensions to the character's core generative concepts with complimentary controller facilitation.


Skinned Mesh Deform 1

Script	skinnedMeshD
Force:	
Magnitude	1
Negative	☐
Speed	1
Bounded	☐
Expand	☐
Direction Offset:	
X	0.005
Y	0
Z	0
Material Physics:	
Permanent	☐
Elasticity	20
Perpetual	☐
Damping	5
Indent	☐
Spread Percent	100
Stretch	☐
Vertical	1
Horizontal	1
Depth	1
Experimental	
Spikey	☐
Wave	☐
Frequency	2
Wobble	☐
Crystal	☐
Power	☐
Exponent	2

Figure 6. Mesh deformation template script's public variables

4.3.1 metalMan

As the name implies, metalMan's mesh deformations are elementally inspired by the malleability of metal. This is indicated by three main state features: sharp attacks, defence armour, and reactive denting. Sharp attacks enabled by Spikey at a randomly generated frequencies and force vectors erupt from metalMan's surface. Defensive armour, rapidly expands a Bound enabled mesh, increasing defence but drastically decreasing speed. And lastly, localized and semi-permanent deformations achieved with a low Spread Percent and Elasticity set to zero, occur in reaction to received attacks. These new deformations may be utilized in regular attack moves, but will disappear when the player switches to another state.

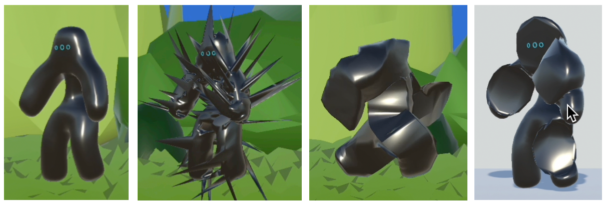


Figure 7. metalMan

4.3.2 plasMan

plasMan uses experimental mesh deformation variables for visualizing chaotic plasmic energy. This effect, applied abstractly to the outer body, is juxtaposed overtop a rigid skeleton. Here generativeness is used for charging and expelling bursts of energy. Player inputs may be stockpiled into perpetual mesh deformations reused in energy attacks, such as shooting plasma balls or energy bursts deformations.

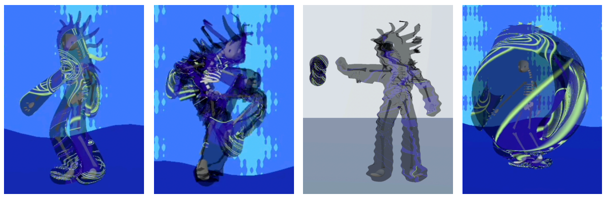


Figure 8. plasMan

4.3.3 rockerChic

rockerChic is the simplest generative character, with their mesh deformation, the smashing of their guitar, performing in reaction to the player's own destruction. Upon entering the opponent's collider, the guitar breaks with a low Spread Percentage, Elasticity set to zero, and Indent enabled, meaning deformations are localized, permanent, and brittle. The guitar's acquired damage can then be reutilized in a "rock out" move, in which rockerChic plays the guitar to project a generative music attack.

4.3.4 slimeCat

slimeCat introduces generative authorship as a moldable character. Here slimeCat's shape can be squished, expanded, and puddled to player liking, strategy, random generative elements, or in reaction to in game events. Squishing applies

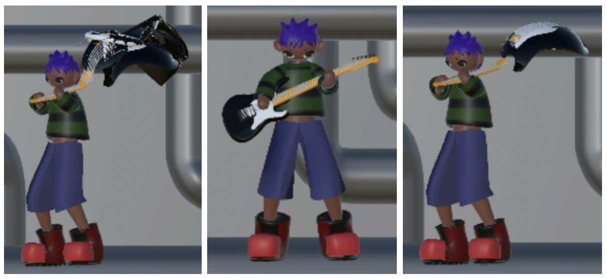


Figure 9. rockerChic

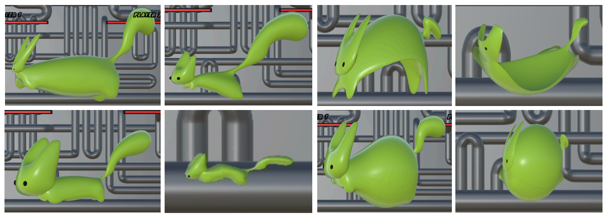


Figure 10. slimeCat

convex of concave deformations (depending on vector direction). Expansion scales up slimeCat with ballooning deformation. And puddle enables Wobble to melt slimeCat into a hard to hit state. These deformations alter the backend mesh and collider, enabling each transformation to redefine how slimeCat interacts with other characters. For example, a larger slimeCat may be more powerful and likely to land an attack, however at this size, they'd also make an easier target.

4.4 Generative Spawns

Generative spawns use a series of real-time object instantiations with algorithmic movements. Through the totality of these objects, fluctuating surfaces emerged as enigmatic structures to be utilized. Such movements are autonomous to player controls, and therefor force the player to adopt new strategies.

Character using generative spawns are still in development, and will be included in hellBear, mushGang, and

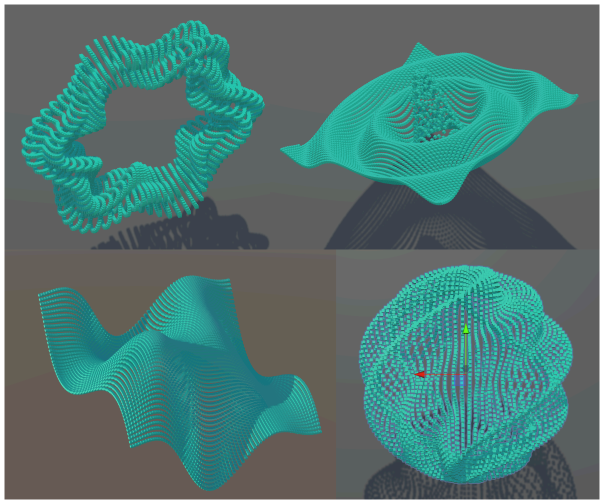


Figure 11. Mathematical Surfaces

swarmOfSquids. hellBear will spawn generative fire in the form mathematical surfaces like Figure 11. mushGang will spawn autonomous mushMinions. And swarmOfSquids generates a mathematical surface populated with squids, in which player controls involve toggling the algorithm's variables.

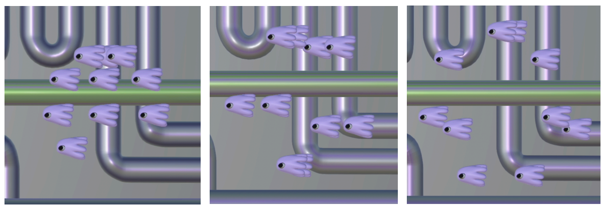


Figure 12. swarmOrSquids

REFLECTION

At this stage, I am elated with *de-generate's* project process. Each character examines a new generative technique that, for the most part, exemplifies *de-generate's* themes well while simultaneously functioning as needed. Mesh deformations were also an especially fruitful exploration. metalMan, plasMan, rockerChic, and slimeCat each embody a distinct interpretation, and thereby present a good range of how generativeness can inform digital bodies. Even still, these characters are only just beginning to scratch the surface.

Next steps will involve finishing the generative spawn characters. I am optimistic they'll be up for GradEx, especially given their fundamentally different approaches would be really interesting in comparison and opposition. Beyond this, the next technique explored will be a combination of generative spawn and mesh deformation principles. Here the idea is that instantiated spawns will be at each vertex position of another mesh. This way, when mesh deformations occur, vertex displacement can be clearly registered.

6. REFERENCES

Flick, Jasper. "C# And Shader Tutorials for the Unity Engine." *Catlike Coding*, Catlike Coding, https://catlikecoding.com/unity/tutorials/.

McCormack, Jon, et al. "Ten Questions Concerning Generative Computer Art." *Leonardo*, vol. 47, no. 2, 2014, pp. 135–41. *JSTOR*, http://www.jstor.org/stable/43834149. Accessed 9 Dec. 2022.

Monro, Gordon. "Emergence and Generative Art." *Leonardo*, vol. 42, no. 5, 2009, pp. 476–77. *JSTOR*, http://www.jstor.org/stable/40540082. Accessed 9 Dec. 2022.

Lefford, Nyssim. "The Generative Process, Music Composition and Games." *Leonardo*, vol. 40, no. 2, 2007, pp. 129–35. *JSTOR*, http://www.jstor.org/stable/20206374. Accessed 9 Dec. 2022.