Draft 1

Introduction

The work I have reproduced is an animation of broken and restructured artefacts created using TouchDesigner, a node-based visual programming language for real-time interactive multimedia content. TouchDesigner is widely used in dynamic visual effects, interactive art, data visualisation, stage performance, and projection mapping.

Process and challenges

To reproduce this animation, I first learned the basics of TouchDesigner through YouTube tutorials and then found a detailed guide specific to the project. As a beginner with no programming experience, these resources were helpful. However, I faced unexpected challenges during the process. While TouchDesigner's node-based logic is intuitive, even slight parameter changes often led to unforeseen deviations in the visual outcome. These challenges demonstrated the tool's potential for exploration but also revealed the tediousness of fine-tuning. Additionally, my limited understanding of complex nodes required extensive experimentation to approximate the desired effects. This experience made me realise that TouchDesigner's power often relies on mastering its system logic, requiring creators to focus more on the tool's operations than on artistic intuition.

Critical Reflections

What makes TouchDesigner special is its ability to preview and modify creations in real time through visual programming nodes. This seamless and fluid workflow is efficient, but it raises questions about its impact on creativity: Does its intuitive

structure limit creators' thinking, making art creation overly "predictable"? Does its node connectivity and path logic reduce the experimental nature of creation, weaken the creator's subjectivity, and let the tool dominate artistic decisions? Unlike traditional design tools, TouchDesigner prioritises process and system logic over narrative or symbolic representation. This shift redefines the designer's role from "storyteller" to "system architect," challenging traditional design paradigms.

Further Exploration

To explore these issues further, I plan to conduct a series of experimental iterations. One possible direction is to subvert TouchDesigner's reliance on precision and logic by incorporating randomness or unpredictability into the production. This approach will examine the tension between control and chaos in digital design and the relationship between tools and human subjectivity.

Draft 2

Introduction

During the second week of exploration, I gradually turned my focus to ‘crashes’ and ‘errors’ in the TouchDesigner software. As a real-time multimedia authoring tool, TouchDesigner strives to provide a smooth and logical workflow that enables users to create dynamic and interactive multimedia works. However, during the first week of reproducing animations, I noticed that the software crashed and lagged when loading too large a file. This experience prompted me to think about whether the software's ‘crashes’ and ‘bugs’ could be seen as a starting point for creation. I chose the *Conditional design workbook* as an analytical tool to revisit my approach and thinking through its core concepts.

Application of theory

Conditional design workbook emphasises that design should be rule-orientated, generating a dynamic process through the setting of conditions, rather than pursuing a defined end result. The setting of rules provides clear boundaries and structure to the design while allowing for unpredictability in the outcome. This approach pays particular attention to the act of designing itself and how the system generates new possibilities in response to conditions. In addition, the manifesto proposes the idea that ‘constraints inspire creativity,’ meaning that constraints are not obstacles to innovation but rather serve as motivators for experimentation and exploration.

These ideas have provided me with new perspectives on my project, allowing me to reconstruct the experimental framework and further deepen the goals of the project.

For example, I have divided my experiments into three phases: triggering the crash, collecting the crash material, and integrating and editing the material. In the past, I triggered crashes in a random way, but now I started to set more explicit rules, such as adjusting parameters to extreme values or overlaying complex components to actively test the boundary values of the software. Through the application of these rules, moments of crashes are no longer isolated phenomena, but the result of system behaviour. These experiments not only revealed the limitations of the software, but also led me to focus on the whole dynamic process: from the triggering conditions of the crash to the formation of its visual representation.

Deepening inquiry

Based on this, I tried to reintroduce the screenshots and videos of these crash moments into TouchDesigner and use its own functionality to conduct iterative experiments. In this way, the software seems to ‘observe’ its own limitations and presents this process of reflection in a visual form. The ‘bugs’ of the software are here redefined as a creative tool, further exploring its possibilities through rule-driven dynamic generation.

Conclusion

In addition, my project reflects the idea that ‘limitations inspire creativity’ ; TouchDesigner's crashes reveal its systemic fragility, but these limitations also create opportunities to explore the tension between control and chaos. By turning constraints into conditions for experimentation, my project breaks with the traditional view of software as a seamless tool and transforms TouchDesigner into a ‘participant’ in the creative process, a system that both generates and analyses its own visual output.

By applying the theory of *Conditional design workbook*, I no longer see crashes as simple technical failures, but rather as dynamically generated results of design rules. This perspective not only helped me dig deeper into the potential and limitations of the TouchDesigner system, but also brought new directions for the project to explore. Next, I plan to further refine the rule set, experiment with different iteration patterns, and observe how the rules shape the behaviour and visualisation of crashes.

Reference

Maurer, L. et al. (2013) *Conditional design workbook*. Amsterdam: Valiz.

Draft 3 (original text)

In this week's research, I continued to focus on exploring the boundaries of crashes in TouchDesigner, and attempted to combine the process of these crashes with the notion of the 'screensaver'. Initially, I viewed crashes as a limitation of the software, but as my research progressed, I began to wonder: what if crashes were not just bugs, but a visual logic that could be designed and exploited?

I tried to capture the moment of a crash and translate it into a visual cyclical process that would echo the logic of a screensaver. Screensavers, by their very nature, are designed to prevent the aging of the screen due to prolonged static display, and so are usually kept dynamic by means of a continuous animation. And the crash behaviour of software itself has cyclic characteristics, such as automatic restart of the program, error windows popping up continuously, and recovery of the interface after it gets stuck. Therefore, with this project, I hope to turn crashes into a continuous visual experience that becomes part of a screensaver.

In the previous week's research, I applied the theory of *Conditional design workbook*, which views design as a rule-driven dynamic process rather than the pursuit of a fixed end result. This week, I further systematised my study of the TouchDesigner crash process and controlled the pattern of its occurrence through rule-based approaches. In order to study TouchDesigner's crash behaviour in a more systematic way, I set up a set of experimental rules to test its different types of crashes in a structured way. I specified several triggering conditions, such as: entering extreme parameters, loading oversized files, and repeatedly overlaying complex components. These approaches

make crashes a systematic behaviour that can be predicted and controlled rather than a random phenomenon. Through my experiments, I recorded patterns of automatic recovery, window pop-up loops, and loading timeouts after software crashes, and translated these phenomena as part of a screensaver. For example, in the experiment of recording the startup time of TouchDesigner, I found that every time I opened the software, it took 51 seconds to load, which is a kind of ‘mandatory wait’ for the user, and even a kind of ‘crash feeling’. Therefore, I recorded and edited this loading process into an infinite loop, making it appear as a startup process that never finishes. In addition, I also tried to set up components and code that would allow the software to trigger its own crashes, creating a ‘crash logic’ controlled by the system itself.

Through this week's experiments, I not only tried to redefine the design process through the lens of a ‘system out of control’, but also explored the technical boundaries and limitations of TouchDesigner. I started to think about how the limitations of the tool can inversely shape the creative process. As a real-time multimedia authoring tool, TouchDesigner aims to provide a fluid and logical workflow that enables users to create dynamic and interactive multimedia works. In practice, however, I have found that TouchDesigner's complexity and instability can be an opportunity to create - computational boundaries, error feedback, crashes, load times, etc., all constitute a ‘language of the tool’ and, to a certain extent, a ‘language of the tool itself’. The computational boundaries, error feedback, crash mechanisms, loading times, etc., all constitute a ‘language of the tool itself’, and to a certain extent determine the final shape of the work. This prompted me to think more deeply about whether creation is entirely creator-driven, or is it partially constrained by the tool itself? Is digital creation always a search for possibilities within the framework and

limitations of the tool? In contemporary digital art and design practice, can we turn the instability, complexity and limitations of tools into a creative method? In the future, I hope to continue to study these questions in depth and explore more ways to transform the ‘limitations of tools’ into creative methods.

Reference

Maurer, L. et al. (2013) *Conditional design workbook*. Amsterdam: Valiz.

Draft 3 (“translated”)

In this week's research, I continued to focus on exploring the boundaries of crashes in TouchDesigner, and [SYSTEM FAILURE]

[REBOOTING SYSTEM...]

[REBOOTING SYSTEM...]

[REBOOTING SYSTEM...]

[REBOOTING SYSTEM...]

[REBOOTING SYSTEM...]

Rebooting in 5...4...3...2...1...

---

attempted to combine the process of these crashes with the notion of the ‘screensaver’. Initially, I viewed crashes as a limitation of the software, but ** m$ research p@!gres3**, I began to wo@!**: what if crashes were not just bugs, but a visu** log!$ that could be designed and ex@!*****?

I tried to T0uchDe$ign3r Err0r: **Sy$t3m F@!lur3**

L0ad!ng....L0ad!ng....L0ad!ng....

15%

35%

65%

... **99%**

... **99%**

... **99%**

... **99%**

... **99%**

[redacted]... **99%**

[redacted]... **99%**

[redacted]... **99%**

[redacted]... **99%**

[redacted]... **99%**

ERROR CODE: 0x0007F1 capture the moment of a crash and translate it into a visual cyclical process that would echo the logic of a screensaver. Screensavers, by their very nature, are designed to prevent the aging of the screen due to prolonged static display, and so are usually kept dynamic by means of a continuous animation. And the crash behaviour of software itself has cyclic characteristics, such as automatic restart of the program, error windows popping up continuously, and recovery of the interface after it gets stuck. Therefore, with this project, I hope to **rn crashes in!! a continuous visual experien** t!** becomes p@$! of a scr**n!av!!.

** ¥!! previous week's resear@@, I ap@$!e!

L0ad!ng....L0ad!ng....L0ad!ng....

L0ad!ng....L0ad!ng....L0ad!ng....

L0ad!ng....L0ad!ng....L0ad!ng....the theory of *Conditional design workbook*, which views design as a rule-driven dynamic process rather than the pursuit of a fixed end result. This week, I further systematised my study of the TouchDesigner crash process and controlled the pattern of its occurrence through rule-based approaches. In order to study TouchDesigner's crash behaviour in a more sy[illegible]

[illegible] outs after software crashES= AND TRANSLATED THESE PHENOMENA AS PART OF A SCREENSAVER. FOR EXAMPLE= IN THE EXPERIMENT OF RECORDING THE STARTUP TIME OF TOUCHDESIGNER= I FOUND THAT EVERY TIME I OPENED THE SOFTWARE= IT TOOK 51 SECONDS TO load, which is a kind of ‘mandatory wait’ for the user, and even a kind of ‘crash feeling’. Therefore, I recorded and edited this loading process into an infinite loop, making it appear as a startup process that never finishes. In addition, I also tried to set up components and code that would allow the software to trigger its own crashes, creating a ‘crash logic’ con@!!lle* ** [illegible] SYSTEM FAILURE]

[REBOOTING SYSTEM...]

[REBOOTING SYSTEM...]

Rebooting in 5...4...3...2...1...

Through this week's experiments, I not only tried to redefine the design process through the lens of a ‘system out of control’, but also explored the technical boundaries and limitations of TouchDesigner. I started to think about how the limitations of the tool can inversely shape the creative process. As a real-time multimedia authoring tool, TouchDesigner aims to provide a fluid and logical workflow that enables users to create dynamic and interactive multimedia works. In practice, however, I have found that TouchDesigner's complexity and instability can be an opportunity to create - computational boundaries, error feedback, crashes, load times, etc., all constitute a ‘language of the tool’ and, to a certain extent, a ‘language of the tool itself’. The computational boundaries, error feedback, crash mechanisms, loading times, etc., all constitute a ‘language of the tool itself’, and to a certain extent

determine the final shape of the work. This prompted me to think more deeply about whether creation is entirely creator-driven, or is it pa[illegible]porary digital arT AND DESIGN PRACTICE CAN WE turn the instab[illegible]ons in depth and explore more ways to transform the 'limitations of tools' **** !r$!!@ve meth!!s.SYSTEM FAILURE]

[REBOOTING SYSTEM...]

[REBOOTING SYSTEM...]

ERROR: UNABLE TO REBOOT.

```
def crash():
    print("SYSTEM ERROR: RECURSION LIMIT REACHED")
    return crash()


crash()
```