UNITED STATES PATENT AND TRADEMARK OFFICE

---

BEFORE THE PATENT TRIAL AND APPEAL BOARD

---

NVIDIA CORPORATION,
Petitioner,

v.

NEURAL AI, LLC,
Patent Owner.

---

Case No. IPR2025-00608
U.S. Patent No. RE49,461

---

**Declaration of Tajana S. Rosing, Ph.D.**
**in Support of Petitioner in the**
**Petition for *Inter Partes* Review of U.S. Patent No. RE49,461**

# TABLE OF CONTENTS

**Page**


I. INTRODUCTION ..... 1
II. BASIS FOR OPINIONS ..... 2
A. Qualifications ..... 2
III. LEGAL UNDERSTANDINGS ..... 5
A. A Person of Ordinary Skill in the Art ..... 5
B. General Claim Construction Principles ..... 8
C. Obviousness ..... 9
IV. BACKGROUND TECHNOLOGY ..... 12
A. GPUs for General Purpose Computing ..... 13
1. History of Graphics Processing Units ..... 14
2. Development of GPUs for General Purpose Computing ..... 16
B. Known Features of GPUs Prior to the '461 Patent ..... 17
1. Prior Art Parallel Computing and "Controllers" ..... 17
2. The Stream Programming Model of GPUs ..... 20
3. APIs and Programming GPUs ..... 21
4. Known Memory Management Techniques ..... 25
5. Artificial Neural Networks and Other Applications of GPGPU ..... 31
6. Real-Time Data Input ..... 34
7. Task and Resource Management on a GPU ..... 35
V. THE '461 PATENT ..... 38
A. Summary of the '461 Patent ..... 38
B. Summary of the Prosecution History of the '461 Patent ..... 45

C. Overview of the Challenged Claims ..... 48
VI. PRIOR ART CONSIDERED ..... 52
A. General Overview of the Primary Prior Art References ..... 53
1. Summary of Buck ..... 53
2. Summary of Wilt ..... 55
3. Summary of nnet ..... 58
4. Summary of ANN ..... 61
5. Summary of GPU Gems ..... 63
VII. ANALYSIS OF THE CHALLENGED CLAIMS AGAINST THE PRIOR ART ..... 66
A. Ground 1: Buck and Wilt Render Obvious Claims 21–41 ..... 67
1. Motivation to Combine Buck and Wilt ..... 67
2. Element-by-Element Analysis ..... 73
a) Claim 21 ..... 73
i. 21[pre]: "A method of executing computations representing an artificial neural network on a computer system comprising at least one central processing unit (CPU), a processing unit, a first memory partition, and a second memory partition, the method comprising:" ..... 73
ii. 21[a]: "executing, by the at least one CPU, a user interaction stream, the user interaction stream controlling transfer of inputs to the artificial neural network to the first memory partition and the second memory partition;" ..... 77
iii. 21[b]: "executing, by the processing unit, a computational stream, the computational stream controlling data exchange between the user interaction stream and the computational stream during execution of the computations representing the artificial neural network;" ..... 80
iv. 21[c]: "shifting control of a data exchange between the user interaction stream and the computational stream to the computational stream in response to starting execution of the computations representing the artificial neural network;" ..... 86

v. 21[d]: “shifting control of the data exchange between the user interaction stream and the computational stream to the user interaction stream in response to completion or interruption of the computations representing the artificial neural network;” ..... 88
vi. 21[e]: “queueing a user command received by the user interaction stream during execution of the computations representing the artificial neural network; and” ..... 91
vii. 21[f]: “executing the user command during execution of the computations representing the artificial neural network at times determined by the computational stream.” ..... 92
b) Claim 22: “The method of claim 21, wherein the user interaction stream controls the data exchange between the user interaction stream and the computational stream outside of execution of the computations representing the artificial neural network.” ..... 95
c) Claim 23: “The method of claim 21, wherein executing the user interaction stream comprises: controlling setting and editing of computational elements of the computations representing the artificial neural network.” ..... 97
d) Claim 24: “The method of claim 21, wherein executing the user interaction stream comprises: controlling setting and editing of parameters of the computations representing the artificial neural network.” ..... 98
e) Claim 25: “The method of claim 21, wherein executing the user interaction stream comprises: controlling setting and editing of parameters of the inputs to the artificial neural network.” ..... 100
f) Claim 26: “The method of claim 21, wherein executing the user interaction stream comprises: specifying an output to be saved to disk and/or displayed on a screen.” ..... 101
g) Claim 27: “The method of claim 21, wherein executing the user interaction stream comprises: parsing elements to be used in the computations representing the artificial neural network.” ..... 104
h) Claim 28: “The method of claim 27, wherein the processing unit comprises a graphics processing unit (GPU) and executing the user interaction stream further comprises: converting the elements into GPU programs.” ..... 105
i) Claim 29: “The method of claim 28, wherein executing the user interaction stream comprises: compiling the GPU programs.” ..... 107

j) Claim 30: “The method of claim 29, wherein executing the user interaction stream comprises: transferring the GPU programs to the second memory partition.”..... 108
k) Claim 31: “The method of claim 21, further comprising: executing, by the at least one CPU, a data output stream, the data output stream controlling transfer of outputs of the computations representing the artificial neural network to disk.”..... 109
l) Claim 32: “The method of claim 21, further comprising: generating the inputs with a video camera during execution of the computations.” 111
m) Claim 38: “The method of claim 21, wherein the user command causes interruption of the computations representing the artificial neural network.” 113
n) Claim 39: “The method of claim 21, wherein the user command causes a change in the inputs to the artificial neural network.” ..... 114
o) Claim 40: “The method of claim 21, wherein the user command causes a change in display properties of an output of the computations representing the artificial neural network.”..... 115
p) Claim 41: “The method of claim 21, wherein, during execution of the computations representing the artificial neural network, the computational stream controls the data exchange between the user interaction stream and the computational stream by requesting the inputs to the artificial neural network.”..... 117
q) Claim 33:..... 118
i. 33[pre]: “A system for executing computations representing an artificial neural network, the system comprising:”..... 118
ii. 33[a]: “a camera to acquire input data for the artificial neural network;”..... 119
iii. 33[b]: “a first memory partition;” ..... 120
iv. 33[c]: “a second memory partition;”..... 121
v. 33[d]: “at least one central processing unit (CPU), operably coupled to the camera, the first memory partition, and the second memory partition, to execute a user interaction stream, the user interaction stream controlling transfer of the input data acquired by the camera to the first memory partition and the second memory partition during execution of the computations representing the artificial neural network;”..... 122

vi. 33[e]: “a processing unit, operably coupled to the first memory partition, the second memory partition, and the at least one CPU, to execute a computational stream, the computational stream controlling transfer of the input data from the first memory partition and the second memory partition during execution of the computations representing the artificial neural network, the execution of the computations representing the artificial neural network occurring while the camera is acquiring the input data; and” ..... 125
vii. 33[f]: “a controller, operably coupled to the at least one CPU and the processing unit, to queue user interactions received by the user interaction stream during the execution of the computations representing the artificial neural network for performance at times selected to avoid data corruption.” ..... 128
r) Claim 34: “The system of claim 33, wherein the user interactions cause interruption of the computations representing the artificial neural network.” ……131
s) Claim 35: “The system of claim 33, wherein the user interactions cause a change in inputs to the artificial neural network.”..... 132
t) Claim 36: “The system of claim 33, wherein the user interactions cause a change in display properties of an output of the computations representing the artificial neural network.”..... 132
u) Claim 37: “The system of claim 33, wherein the controller is configured to request the input data during the execution of the computations representing the artificial neural network.”..... 133

B. Ground 2: Buck in view of Wilt and nnet Render Obvious Claims 23–26, 34–36, and 38–40..... 134

1. Motivation to Combine nnet with Buck and Wilt ..... 134

2. Element-by-Element Analysis ..... 137

a) Element 21[a]: “executing, by the at least one CPU, a user interaction stream, the user interaction stream controlling transfer of inputs to the artificial neural network to the first memory partition and the second memory partition”..... 137
b) Claim 23: “The method of claim 21, wherein executing the user interaction stream comprises: controlling setting and editing of

computational elements of the computations representing the artificial neural network.” ..... 140
c) Claim 24: “The method of claim 21, wherein executing the user interaction stream comprises: controlling setting and editing of parameters of the computations representing the artificial neural network.” ..... 141
d) Claim 25: “The method of claim 21, wherein executing the user interaction stream comprises: controlling setting and editing of parameters of the inputs to the artificial neural network.” ..... 142
e) Claim 26: “The method of claim 21, wherein executing the user interaction stream comprises: specifying an output to be saved to disk and/or displayed on a screen.” ..... 144
f) Claim 38: “The method of claim 21, wherein the user command causes interruption of the computations representing the artificial neural network.” 148
g) Claim 39: “The method of claim 21, wherein the user command causes a change in the inputs to the artificial neural network.” ..... 150
h) Claim 40: “The method of claim 21, wherein the user command causes a change in display properties of an output of the computations representing the artificial neural network.” ..... 151
i) Claim 34: “The system of claim 33, wherein the user interactions cause interruption of the computations representing the artificial neural network.” 152
j) Claim 35: “The system of claim 33, wherein the user interactions cause a change in inputs to the artificial neural network.” ..... 153
k) Claim 36: “The system of claim 33, wherein the user interactions cause a change in display properties of an output of the computations representing the artificial neural network.” ..... 153

C. Grounds 3 and 4: Buck in view of Wilt, and ANN or GPU Gems Render Obvious Claims 32 and 33–37 ..... 154

1. Motivation to Combine ANN or GPU Gems with Buck and Wilt ..... 154

a) ANN ..... 155
b) GPU Gems ..... 156
c) Implementing the Computer Vision Method of ANN or GPU Gems in Buck-Wilt’s System ..... 157

2. Element-by-Element Analysis ..... 160
a) Claim 32: "The method of claim 21, further comprising: generating the inputs with a video camera during execution of the computations." 160
b) Claim 33 ..... 162
i. 33[pre]: "A system for executing computations representing an artificial neural network, the system comprising:" ..... 162
ii. 33[a]: "a camera to acquire input data for the artificial neural network;" ..... 162
iii. 33[b]: "a first memory partition;" ..... 163
iv. 33[c]: "a second memory partition;" ..... 164
v. 33[d]: "at least one central processing unit (CPU), operably coupled to the camera, the first memory partition, and the second memory partition, to execute a user interaction stream, the user interaction stream controlling transfer of the input data acquired by the camera to the first memory partition and the second memory partition during execution of the computations representing the artificial neural network;" ..... 164
vi. 33[e]: "a processing unit, operably coupled to the first memory partition, the second memory partition, and the at least one CPU, to execute a computational stream, the computational stream controlling transfer of the input data from the first memory partition and the second memory partition during execution of the computations representing the artificial neural network, the execution of the computations representing the artificial neural network occurring while the camera is acquiring the input data; and" ..... 166
vii. 33[f]: "a controller, operably coupled to the at least one CPU and the processing unit, to queue user interactions received by the user interaction stream during the execution of the computations representing the artificial neural network for performance at times selected to avoid data corruption." ..... 169
c) Claim 34: "The system of claim 33, wherein the user interactions cause interruption of the computations representing the artificial neural network." 170
d) Claim 35: "The system of claim 33, wherein the user interactions cause a change in inputs to the artificial neural network." ..... 170

e) Claim 36: "The system of claim 33, wherein the user interactions cause a change in display properties of an output of the computations representing the artificial neural network." ..... 170
f) Claim 37: "The system of claim 33, wherein the controller is configured to request the input data during the execution of the computations representing the artificial neural network." ..... 170

D. Grounds 5 and 6: Buck in view of Wilt, ANN or GPU Gems, and nnet Render Obvious Claims 34–36 ..... 171

VIII. SECONDARY CONSIDERATIONS ..... 174

IX. RIGHT TO SUPPLEMENT ..... 174

X. CONCLUSION ..... 175

# EXHIBIT LIST

| Exhibit | Description |
|---|---|
| 1001 | U.S. Patent No. RE49,461 |
| 1002 | Excerpt of File History for U.S. Patent No. RE49,461 (App. No. 17/136,343) |
| 1003 | Declaration of Prof. Tajana Rosing, Ph.D. |
| 1004 | U.S. Patent Application Publication No. 2005/0197977 ("Buck") |
| 1005 | U.S. Patent Application Publication No. 2003/0140179 ("Wilt") |
| 1006 | Neural Network Toolbox User's Guide Version 4 ("nnet") |
| 1007 | Zhongwen Luo, *Artificial Neural Network Computation on Graphic Process Unit* (2004) |
| 1008 | GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation (2005) ("GPU Gems") |
| 1009 | Plaintiff's Disclosure of Asserted Claims and Preliminary Infringement Contentions Pursuant to the Court's Standing Order Governing Patent Proceedings (Jan. 14, 2025) |
| 1010 | Declaration of Dr. Mary Bolin |
| 1011 | Numerical Recipes in C (2d ed. 2002) |
| 1012 | Jeanne Martin, *Fortran 90 Pointers vs. "Cray" Pointers*, 11 ACM SIGPLAN Fortran Forum (1992) |
| 1013 | Arthur Veen, *Dataflow Machine Architecture* (1986) |
| 1014 | Michael Flynn, *Some Computer Organizations and Their Effectiveness* (1972) |
| 1015 | Press Release – NVIDIA Launches the World's First Graphics Processing Unit; GeForce 256 (Aug. 31, 1999) |
| 1016 | Excerpt of OpenGL Shading Language (2004) |
| 1017 | Excerpt of The Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics (2003) |
| 1018 | OpenGL 2.1 Reference Pages |
| 1019 | Advanced Image Processing with DirectX 9 Pixel Shaders (2004) |

| Exhibit | Description |
|---|---|
| 1020 | Dominik Göddeke -- GPGPU::Basic Math Tutorial |
| 1021 | Ian Buck, *Data Parallel Computation on Graphics Hardware* (2003) |
| 1022 | Youquan Liu, *Real-Time 3D Fluid Simulation on GPU with Complex Obstacles* (2004) |
| 1023 | Kyoung-Su Oh, *GPU implementation of neural networks* (2004) |
| 1024 | U.S. Patent No. 7,673,304 ("Gosalia") |
| 1025 | Thomas Rolfes, *Artificial Neural Networks on Programmable Graphics Hardware* in Game Programming Gems 4 (2004) |
| 1026 | P.J.G. Lisboa, *A review of evidence of health benefits from artificial neural networks in medical intervention* (2002) |
| 1027 | U.S. Patent No. 7,418,576 ("Lindholm") |
| 1028 | Bertil Svensson, *SIMD Processor Array Architectures* in PARALLEL PROCESSING IN INDUSTRIAL REAL-TIME APPLICATIONS (1992) |
| 1029 | Michael Glover, *A Massively-Parallel SIMD Processor for Neural Network and Machine Vision Applications* (1993) |
| 1030 | Francisco Mesa-Martinez, *The UCSC Kestrel High Performance SIMD Processor: Present and Future* (2003) |
| 1031 | Getting Started with MATLAB, Version 7 (2005) |
| 1032 | Declaration of Gordon MacPherson |
| 1033 | *The C programming Language* (1988) |
| 1034 | U.S. Patent No. 7,861,060 ("Nickolls") |
| 1035 | U.S. Patent No. 7,139,003 ("Kirk") |
| 1036 | *Sotera* Stipulation (Mar. 25, 2025) |

I, Dr. Tajana S. Rosing, Ph.D., do hereby declare as follows.

## I. INTRODUCTION

1. I am making this Declaration at the request of NVIDIA Corporation, ("Petitioner" or "NVIDIA") in the matter of the *Inter Partes* Review of U.S. Patent No. RE49,461 ("the '461 patent").

2. I am being compensated for my work in this matter at my standard hourly rate. I am also being reimbursed for reasonable and customary expenses associated with my work and testimony in this proceeding. My compensation is not contingent on the outcome of this matter or the specifics of my testimony.

3. I have been asked to provide my opinions regarding whether the subject matter of claims 21–41 ("the Challenged Claims") of the '461 patent would have been obvious to a person having ordinary skill in the art ("POSITA") at the time of the alleged invention, in light of the prior art. It is my opinion that the Challenged Claims would have been obvious to a POSITA.

4. In forming the opinions expressed below and preparing this Declaration, I have considered: the challenged '461 patent; its file history; each of the exhibits identified in the list of exhibits; the relevant legal standards, including the standard for obviousness; and my own knowledge and experience based upon my work related to the field of the '461 patent.

5. Unless otherwise noted, all **emphasis** in any quoted material has been added. Claim terms are *italicized*.

## II. BASIS FOR OPINIONS

### A. Qualifications

6. My complete qualifications and professional experience are described in my Curriculum Vitae, a copy of which can be found as Appendix A. The following is a brief summary of my relevant qualifications and professional experience. Among other things, my CV includes a list of my issued patents and publications for at least the last 30 years.

7. I am a distinguished professor, IEEE and ACM Fellow, and the Fratamico Endowed Chair in the Computer Science and Engineering Department at the University of California, San Diego (UCSD), and adjunct distinguished professor in the Electrical and Computer Engineering Department at UCSD. My research focuses on energy-efficient computing. My place of business is located at 9500 Gilman Drive, La Jolla, California. I am over the age of eighteen, and am a citizen of the United States.

8. I am currently a Director of System Energy Efficiency Lab where I am leading a diverse research team on projects relating to system energy-efficiency. I have over thirty years of academic and industry experience in applying, designing, studying, teaching, and writing about energy-efficient computing, including

datacenters and their interaction with smart grid and renewable energy resources. My experience has spanned both hardware and software, along with measurements and simulations related to datacenters, smart grid, renewable energy resources, power, thermal and reliability management (for which I was nominated both IEEE and ACM Fellow).

9. I received an Electrical Engineering Ph.D. degree in 2001 from Stanford University; my thesis was titled "Energy Efficient System Design and Utilization." Before that, I received an Electrical Engineering B.S. degree in 1992 from Northern Arizona University, an Electrical and Computer Engineering M.S. degree in 1993 from the University of Arizona, and an Engineering Management M.S. degree in 2000 from Stanford University.

10. While at Stanford University, I worked in the same office and on the same machines where Yahoo was started, and just down the hall from the office where Google was started. After completing my M.S. in Electrical and Computer Engineering, I worked at Altera Corporation, which is now Intel Corporation, as a senior design engineer for four years. During and after completing my Ph.D. degree, I worked with Stanford University and Hewlett-Packard Labs leading a team of researchers to develop products for the wireless portable devices market. At HP Labs, my team's efforts also included work related to content delivery networks,

streaming video over wireless, videoconferencing using mobile platforms, and energy efficiency of chips, mobile as well as server systems.

11. Through the research and work experience described above, I have significant experience in the area of computer systems and architecture, including the use of graphic processing units (GPUs), as well as with artificial neural networks. I have many publications related to accelerating machine learning algorithms, including neural networks, using GPUs, FPGAs, ASICs and in/near memory/storage computing devices. This work has led to me winning $56.5M from DARPA and Semiconductor Research Corporation JUMP2.0 center grant that focuses on next generation computing architecture which leverage processing in and near memory for big data workloads such as machine learning. The center I am leading, PRISM, Processing with Intelligent Storage and Memory, has 24 PIs and more than 230 PhD students across 13 top universities in the USA. Furthermore, I am a co-PI of DARPA/SRC JUMP2.0 CoCoSys center which focuses on acceleration and design of next generation cognitive computing architectures, which push the envelope of machine learning into neuromorphic, symbolic, stochastic and hyperdimensional computing. Through my work on Hyperdimensional computing, a new form of machine learning, and its acceleration on GPUs, FPGAs, ASICs and in/near memory/storage, I was awarded a top University Researcher for Design in 2022, an award given by the Semiconductor Industry Association which represents 99% of

the world's semiconductor industry by revenue. This award is given to only one individual per year.

12. My publication history is detailed in my attached CV. I have published over 400 publications and receive a number of best paper awards and nominations. I have also been an invited speaker at numerous academic and industry conferences. I am an inventor on several U.S. and foreign patents related to my work.

13. Based on my experience and education, I believe that I am qualified to opine as to the knowledge and level of skill of a POSITA at the time of the '461 patent (which I further describe below), in addition to the general state of the art during that time.

## III. LEGAL UNDERSTANDINGS

14. In this section, I describe my understanding of certain legal standards that I may have relied upon in forming my opinions set forth in this Declaration. I have been informed of these legal standards by Petitioner's attorneys. Although I am not an attorney and have not researched the law on patent invalidity, I have formed the following understanding about the legal standards.

### A. A Person of Ordinary Skill in the Art

15. I understand there are multiple factors relevant to determining the level or ordinary skill in the pertinent art, including (1) the levels of education and experience of persons working in the field at the time of the invention; (2) the

sophistication of the technology; (3) the types of problems encountered in the field; and (4) the prior art solutions to those problems.

16. I further understand that the hypothetical POSITA is presumed to have knowledge of all references that are sufficiently related to one another and to the pertinent art, and to have knowledge of all arts reasonably pertinent to the particular problem that the claimed invention purports to address.

17. The ’461 patent relates to the field of computer systems and architecture, including the use of GPUs. In my opinion, as it relates to the ’461 patent, a POSITA in and around 2005 and 2006, would have had at least a bachelor’s degree in electrical or computer engineering (or an equivalent discipline) and at least two years’ worth of experience in research, design and/or development of computer systems, including graphical processing units. Additional education could substitute for experience, and vice versa. For example, a master’s degree or higher would have been acceptable in place of the work experience described above.

18. I meet the criteria described above and believe myself to be a person with at least ordinary skill in the art pertaining to the ’461 patent. I would have qualified as such a person by at least September 25, 2006, which I understand is the date the provisional application to which the ’461 patent was filed claims priority. I would also have qualified as such a person by February 2005, which I understand is

when Patent Owner Neural AI alleges the invention in the ’461 patent was conceived.

19. For the purposes of this Declaration, in general, and unless otherwise noted, my statements and opinions, such as those regarding my own experience and what a POSITA would have understood or known generally (and specifically related to the references I consulted herein), reflect the knowledge that existed in the relevant field as of the priority date of the ’461 patent.

20. As I discuss in detail below, the ’461 patent relates to the use of GPUs to accelerate parallel computations, which is relevant to many fields that perform numerical simulations, including artificial neural networks. The ’461 patent explains that “[n]umerical simulations, e.g., finite element analysis, of large systems of similar elements (e.g. neural networks, genetic algorithms, particle systems, mechanical systems) . . . can benefit from GPGPU computation.” Ex1001, 1:59–63. In my opinion, a POSITA in the relevant time frame would have frequently reviewed relevant books and publications, in print and online, and attended conferences for information on relevant technologies. A POSITA interested in this subject matter would have searched for books and publications by keywords such as “GPU,” “neural networks,” or “computer graphics,” alone and with “programming.”

### B. General Claim Construction Principles

21. I understand that there are two types of claims: independent claims and dependent claims. I understand that an independent claim stands alone and includes only the limitations it recites. I understand that a dependent claim is a claim that depends on another claim. I understand that a dependent claim includes all of the limitations recited in the dependent claim as well as any limitations included in the corresponding independent claim.

22. I understand that before determining whether the Challenged Claims are invalid, the claims must be construed on a limitation-by-limitation basis. I understand that claim terms are to be afforded their plain and ordinary meaning as used in the claims and as would be understood by a POSITA at the time of the invention. I understand that this person skilled in the art at the time of the invention is assumed to have read the claim terms in light of the entire patent record, including the other claims, the patent specification, and the patent prosecution history. I have interpreted the claim terms in the ’461 patent in light of this understanding.

23. I understand that to properly understand the meaning of claim terms, one should first consider the intrinsic evidence, which includes the claim language itself, the patent specification, and the patent’s prosecution, as well as any post-issuance history before the Patent Office. For example, the patent specification may show that the inventor used words or terms in a manner inconsistent with their plain

and ordinary meaning. Specifically, I understand that where the specification reveals a special definition given to a claim term by the inventor that differs from the meaning it would otherwise possess, the inventor's lexicography governs. I understand that the prosecution history of the patent may also provide guidance in construing a claim term. For example, the prosecution history may show that the patent applicant might have limited the scope of some or all of the claims during prosecution, either affirmatively or by implication.

24. I understand that if the intrinsic evidence is not conclusive regarding the meaning of a particular claim term, extrinsic evidence may also be used to determine its meaning. I understand that extrinsic evidence may be used, for example, to help determine what a person skilled in the art at the time of the invention would have understood the claim term to mean. Extrinsic evidence may include dictionaries, technical treatises and books, articles, or expert testimony.

**C. Obviousness**

25. I understand that for a claim to be invalid as obvious, the party asserting invalidity must identify prior art references that alone or in combination with other prior art references would have rendered the claim obvious to one of ordinary skill in the art at the time of the invention. I understand that the analysis of obviousness looks at (1) the scope and content of the prior art; (2) the differences between the

claims and the prior art; (3) the level of ordinary skill in the art; and (4) secondary considerations of non-obviousness.

26. I understand that a single prior art reference can render a claim obvious where it would have been obvious to modify that reference to arrive at the patented invention. I understand that it is appropriate to look to the knowledge of a POSITA in assessing whether it would have been obvious to modify a single prior art reference to produce the claimed invention. I also understand that a conclusion of obviousness may be based on a combination of prior art references. I understand that it may be helpful to identify a reason that would have prompted a person of ordinary skill in the relevant field to combine the elements in a way that the claimed invention does.

27. I understand that in determining whether a combination of prior art references renders a claim obvious, it may be helpful to consider whether there is some teaching, suggestion, or motivation to modify or combine the references and whether there would have been a reasonable expectation of success in doing so.

28. I understand that there are several recognized rationales for combining references or modifying a reference to show obviousness. These rationales include: (a) combining prior art elements according to known methods to yield predictable results; (b) simple substitution of one known element for another to obtain predictable results; (c) use of a known technique to improve a similar device

(method, or product) in the same way; (d) applying a known technique to a known device (method, or product) ready for improvement to yield predictable results; (e) choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success; and (f) some teaching, suggestion, or motivation in the prior art that would have led a POSITA to modify the prior art or to combine prior art teachings to arrive at the claimed invention.

29. I understand that in evaluating whether patent claims are invalid as obvious, secondary considerations of non-obviousness are considered, including considerations such as commercial success, long-felt but unsolved needs, copying, praise, teaching away from the claimed invention by the prior art, unexpected results, industry acceptance, failure of others, and skepticism by experts. I understand that the secondary considerations are considered with the balance of obviousness evidence in the record to act as a check against impermissible hindsight bias. I further understand that there must be a nexus between the merits of the claimed invention and the evidence of secondary considerations in order to give such considerations substantial weight in the determination of obviousness.

30. I understand that the burden of presenting evidence of secondary considerations is on the patentee. I also understand that the proponent of the evidence of secondary considerations bears the burden of showing that a nexus exists between

the claimed features of the invention and the evidence offered to show non-obviousness.

## IV. BACKGROUND TECHNOLOGY

31. As I explain in more detail in Section V, the claims of the ’461 patent relate to maintaining separate instruction streams for separate data types and queuing certain tasks based on to “avoid data corruption.” Ex1001, 18:21–26. At a high level, the CPU executes an instruction stream for processing user interactions and the other processing unit executes an instruction stream for processing computations representing an artificial neural network, and interaction and dependencies between the streams is “minimal.” *Id.*, 2:11–19. The claims recite shifting “control” over whether data can be exchanged between the two streams and queuing a user command received while the computations are being performed for execution at a time determined by the computational stream. To put in context my opinions on the invalidity of the claims of the ’461 patent, in this section I provide a brief discussion of the relevant technological background.

32. I begin by discussing the history of GPUs and their use for numerical computations (such as for artificial neural networks), an area in which Petitioner NVIDIA and others were at the forefront long before the ’461 patent. I then describe the ideas and methodologies claimed in the ’461 patent for performing numerical

simulations and computations, such as simulating artificial neural networks, on GPUs that were well known and practiced in the art before the ’461 patent.

### A. GPUs for General Purpose Computing

33. Fundamentally, computers are machines that perform mathematical operations on input data to create output data. Every task a computer performs—whether it’s processing text, rendering an image, or running complex simulations—boils down to executing calculations based on binary logic. This fundamental capability makes computers inherently suited for mathematical and scientific computations. Thus, from the earliest days of computers to now, scientists and engineers have relied on computers’ speed and accuracy to solve difficult problems.

34. Originally, numerical computations were handled by central processing units (CPUs), which excel at executing instructions sequentially, *i.e.*, one step at a time. While they were highly effective for many applications, CPUs’ ability to perform large-scale numerical simulations was limited by their sequential nature, leading researchers to explore alternative hardware solutions for faster and more efficient computation.

35. Parallel processing concepts and systems for parallel processing, including for both graphics and general-purpose computing, were well known for decades before the ’461 patent. As discussed below, in the late 1990s, NVIDIA developed the first consumer-level graphics card that could offload essentially the

entire graphics pipeline from the CPU, dubbing it a "Graphics Processing Unit," or "GPU." While CPUs excel at handling a few tasks sequentially, as noted above, GPUs are optimized for parallel processing, making them ideal for the thousands of simultaneous calculations required to render graphics.

### 1. History of Graphics Processing Units

36. A graphics processing unit (GPU) is an integrated circuit with computer hardware designed for graphics. GPUs were developed in the 1990s by NVIDIA to accelerate graphics rendering. The '461 patent recognizes that "[v]arious brand[s] of GPU [we]re relevant" at the time, including "GPU's based on the GeForce series from NVIDIA Corporation." Ex1001, 4:44–66. NVIDIA released the world's first programmable GPU, the GeForce 256, in 1999. Ex1015, 1. At the time of release, the GeForce 256 had the equivalent parallel processing performance of a "maximum configuration 256-processor Cray T3D," one of the leading parallel processing systems as the time. *Id.* NVIDIA developed and released several other GPUs prior to 2005, including the GeForce 2 (2000), GeForce 3 (2001), GeForce 4 (2002), GeForce FX (2003), and GeForce 6 (2004) GPUs. *E.g.*, Ex1008, 471–91 ("GeForce 6 Series GPUs provide the GPU programmer with unparalleled flexibility and performance . . . ."); *see also* Ex1017, 11–14 (describing the "generations" of GPUs).

37. Unlike CPUs, which were optimized to handle tasks sequentially, GPUs were designed with parallel processing in mind. GPUs were originally

intended to leverage their parallel processing architecture to accelerate graphics rendering in desktop computers, particularly real-time rendering of three-dimensional environments.

38. At a high level, traditional GPUs processed "textures" (stored as data arrays) by using "shaders" (programs) to modify pixel data and create graphics for display. *See* Ex1008, 500–01. Shaders operated in a highly parallel manner, executing mathematical operations on individual texture elements (sometimes called texels or pixels) to perform tasks like shading, filtering, lighting computations, and blending. These operations were ideal for parallelism because the computations for each element were typically independent of others, allowing GPUs to process thousands of elements simultaneously.

39. After their release, GPUs quickly evolved to include programmable shaders, replacing the older fixed-function pipeline. This advancement gave developers the ability to write custom shader programs, enabling them to manipulate graphics at a much more granular level. For example, programmable shaders allowed for more complex visual effects, such as per-pixel lighting, procedural textures, and realistic surface details, revolutionizing graphics rendering in video games, simulations, and visual effects. These innovations also laid the groundwork for general-purpose computing on GPUs (GPGPU), discussed in more detail below,

enabling GPUs to handle non-graphics tasks like scientific and numerical simulations and machine learning.

### 2. Development of GPUs for General Purpose Computing

40. Not long after GPUs were released for graphics processing, some in the industry—including many at and supported by NVIDIA—realized the power of GPUs as versatile, low-cost parallel computing platforms. Because graphics processing is an inherently parallelizable task, GPUs were designed with parallelism baked into their structure. *See* Ex1008, 457–70. Additionally, GPUs did not need to support as many different sorts of functions and datatypes as general-purpose CPUs; by omitting the circuity needed to support these features, GPU circuits were able to fit more processing power onto a single chip die than an equivalent CPU.

41. GPUs were used almost immediately to perform parallelized computations for non-graphics applications, such as numerical simulations. In the early 2000s, this type of computing was often referred to as "general-purpose computing on GPUs," sometimes referred to as "GPGPU."

42. GPGPU works by using "textures" to store data and repurposing "shaders" to perform computations on the data. Ex1008, 497–502. The techniques enabled GPUs to handle tasks like scientific computing, numerical simulations, and neural network training. NVIDIA and others led GPGPU research and development. For example, NVIDIA not only improved hardware and software for programmable

GPUs, but also actively supported the GPGPU community through educational efforts. Ex1034; Ex1035. NVIDIA published books like GPU Gems, which provided practical guidance and techniques for using GPUs in general-purpose applications, helping developers harness the parallel processing power of GPUs for a wide range of fields. *See* Ex1008, 451 ("General-Purpose Computation on GPUs: A Primer"). One recognized benefit of GPGPU applications was that, with the architecture described above, all calculations occurring in parallel on the GPU, they "free up the CPU for other tasks." *Id.*, 650.

### B. Known Features of GPUs Prior to the '461 Patent

43. The '461 patent describes several features and functions of GPGPU, some of which are recited in the claims. As explained below, these features and functions were all well known to those skilled in the art prior to the '461 patent.

#### 1. Prior Art Parallel Computing and "Controllers"

44. The principles of parallel processing were known prior to the '461 patent, and researchers and engineers in the field had applied these parallelization techniques to GPUs, which are particularly suited to perform parallel computations with their parallel processing engines. Researchers and engineers had also settled on effective designs for parallel processing systems, including a "controller" for managing the parallel computing process.

45. Parallel computing involves dividing a computational task among more than one processor. When appropriately configured, a system using many processors can perform challenging calculations more quickly than a single-processor system by instructing these many processors to work on different parts of the problem simultaneously.

46. Decades before the ’461 patent, parallel computing was of particular interest to the scientific computing community, who expected that speed gains through parallelization would make new classes of experiments and computations possible. Ex1008, 365–67. It was found, however, that parallel processes lose efficiency when individual processors sit idle or when they compete for memory resources.

47. Thus, parallel computing system designers proposed architectures to address such bottlenecks that may arise in such systems. A common solution involved using one set of computer hardware (*e.g.*, a controller) to coordinate the function and results of a second set of hardware that performs the core computations (*e.g.*, processors). By having the controller manage the computations of the parallel processors, as opposed to only a CPU, the CPU is free and available to perform other tasks.

48. For example, one class of control architecture for parallel computation is Single-Instruction-Multiple-Data (“SIMD”), which has been known in the art

since at least the 1970s. Ex1014, 1, 13. There are many examples in the prior art of SIMD architectures (including for neural networks) with a host computer, controller, and parallel processors. Ex1013, 375–89; Ex1028, 1–2; Ex1029, 845; Ex1030, 2.

49. Another class of control architecture for parallel processing is the Multiple Instruction Multiple Data ("MIMD") architecture. Ex1014, 957–58. In such a system, operations are split into units whose outputs do not depend on each other, and which might require different calculations. *Id.*; Ex1008, 547–48 ("[I]n a MIMD-parallel architecture, different processors may simultaneously execute different instructions."). Like SIMD, prior art MIMD systems used a controller. *See* Ex1014, 957–58.

50. Beginning in the late 1990s, those skilled in the art recognized that the existing GPU pipeline could be adapted to known parallel processing paradigms, like SIMD and MIMD. Ex1008, 547–49. Texture memory, originally designed to store a matrix of pixels with float values at each location, closely resembled the vector structures commonly used in numerical analysis. By reimagining its GPU architecture in the context of established parallel computing models, NVIDIA made its GPUs highly suitable for these frameworks. By early 2005, NVIDIA GPUs supported features like "MIMD branching, SIMD branching, and condition codes," all of which were well-understood architectures in the art of parallel computing. *Id.*, 547–49.

51. Companies like NVIDIA had also applied the well-known "controller" framework for directing parallel computations. For example, NVIDIA patents disclose improved GPU architectures with a "controller" to direct the distribution and execution of instructions in the processing cores of the GPU. Ex1034; Ex1035.

**2. The Stream Programming Model of GPUs**

52. Prior to the '461 patent, it was known in the art that GPUs operated based on the "stream programming model," where data is represented as a *stream*—an ordered set of elements of the same data type. Ex1008, 464 ("In the stream programming model, all data is represented as a *stream*—which we define as an ordered set of data of the same data type. That data type can be simple (a stream of integers or floating-point numbers) or complex (a stream of points or triangles or transformation matrices)."). Streams are processed by *kernels*, which is a program or function that "tak[es] one or more streams as inputs and produc[es] one or more streams as outputs." Ex1008, 464.

53. A key feature of this stream programming model is that kernels—the computational units in the pipeline—process entire streams of data rather than individual elements. Ex1008, 464. This design ensures that computations on individual stream elements are entirely independent of one another, enabling what seems like a serial calculation to be executed efficiently on data-parallel hardware. *Id.* Moreover, because the outputs of a kernel depend solely on its inputs, all the data

required for execution is known upfront, allowing for highly optimized data storage, access, and management. This architecture aligned perfectly with the existing graphics pipeline in GPUs (including NVIDIA's), which relied on data parallelism and element independence. *Id.*

54. In the stream programming model, applications are built by chaining multiple kernels together. *Id.*, 465. By the time of the '461 patent, GPUs featured high-performance data-parallel processors capable of implementing two essential kernels in the graphics pipeline: the vertex program and the fragment program. *Id.*, 467. These programs could sustain high computation rates on user-defined tasks with the precision necessary for solving general-purpose computing challenges. In effect, they transformed GPUs into programmable stream processors, making them suitable for far more than graphics rendering. *Id.* Their architecture excelled in tasks with "high arithmetic intensity," such as solving systems of linear equations, making GPUs an attractive solution for a wide range of computationally demanding applications beyond their traditional graphics domain. *Id.*, 467, 495.

### 3. APIs and Programming GPUs

55. Prior to the '461 patent, as adoption of GPGPU spread and demand grew, graphics hardware manufacturers such as NVIDIA developed improved software tools for programmable GPUs, such as advanced application programming interfaces (APIs) and programming languages. These tools facilitated interactions

with GPU pipelines, including programmable shaders for GPGPU tasks. Graphics APIs act as a bridge between software applications and the graphics hardware by providing high-level, simple-to-use functions for developers while handling low-level communication with the hardware.

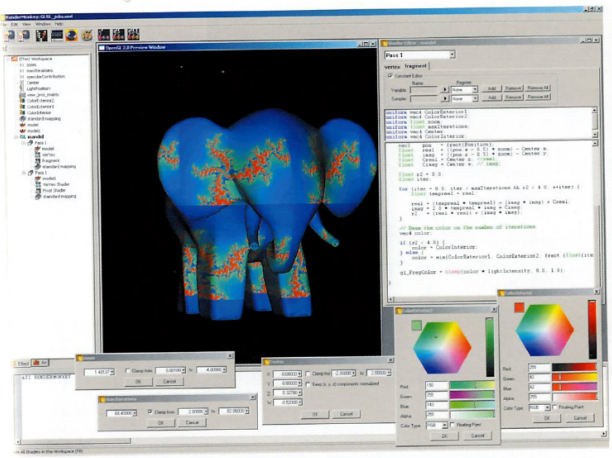
56. For example, programmers made use of NVIDIA's GPUs through graphics Application Programing Interfaces (APIs), such as Direct3D and OpenGL, as well as GPU languages such as HLSL, GLSL, and Cg. Ex1008, xxxii; *see also* Ex1001, 2:20–22 ("A GPU on a conventional video card is usually controlled through OpenGL, DirectX, or similar graphic application programming interfaces (APIs)."). On release, the GeForce 256 GPU used Microsoft's DirectX 7.0 API. Ex1015, 2. By 2004, NVIDIA's GPUs supported OpenGL 2.1, then developed by Silicon Graphics, Inc. NVIDIA worked in tandem with API designers to enable programmers to make better use of their GPUs.

57. OpenGL, short for Open Graphics Library, is a cross-platform API developed in 1992 by Silicon Graphics to standardize the way developers program hardware to render graphics. It provided a hardware-agnostic interface, allowing developers to create complex graphical applications without worrying about the specifics of the underlying hardware. Initially designed for high-performance workstations used in industries like CAD, scientific visualization, and video production, OpenGL soon became a key tool in game development and real-time

rendering. With the introduction of programmable shaders in GPUs during the late 1990s and early 2000s (discussed in further detail below), OpenGL evolved to support these features, enabling developers to write custom shader programs in GLSL (OpenGL Shading Language). Ex1017, 26–29.

58. Other high-level programming languages prior to the ’461 patent included HLSL (High-Level Shader Language) and Cg (C for Graphics). HLSL, developed by Microsoft, was introduced as part of DirectX to provide developers with a tool for creating custom shaders on Windows platforms. Cg, developed by NVIDIA, aimed to simplify GPU programming by providing a C-like syntax, making it accessible to a broader range of developers. Ex1017, 4–5 (“Cg program can perform physical simulation, compositing, and other nonshading tasks.”).

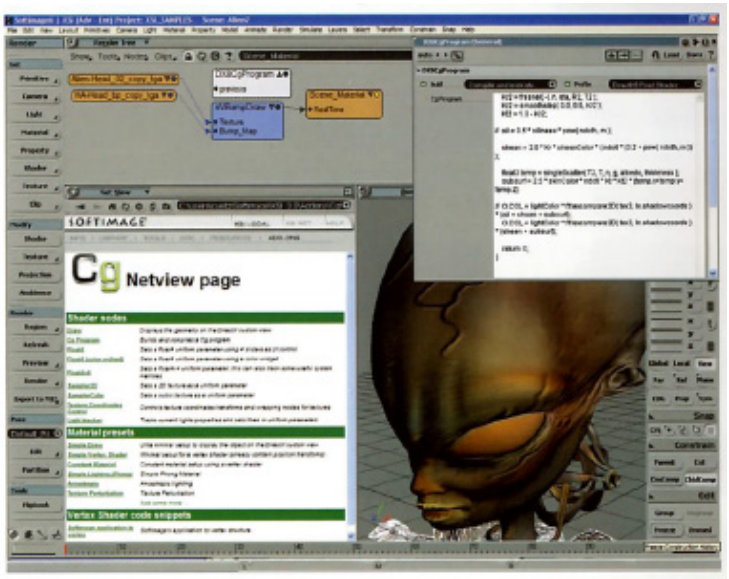
59. Visual tools were also created to facilitate GPU programming and interacting with programs such as modeling and simulation programs. For example, the book *OpenGL Shading Language* shows a graphical user interface (GUI) that allowed a user to interactively adjust parameters used by the GPU for shading an object.



**olor Plate 1** Screen shot of the RenderMonkey IDE user interface. The shader that is procedurally nerating a Julia set on the surface of the elephant is shown in the source code window. Color selection ols and user interface elements for manipulating user-defined uniform variables are also shown.

Ex1016, 27.

60. Similarly, the book *Cg: The Cg Tutorial* describes and shows various Cg viewers that can be used to facilitate user interaction.




Ex1017, 41; *see also id.*, 39–40, 44.

### 4. Known Memory Management Techniques

61. As GPUs became more powerful, with increasing clock speeds and chip sizes, the amount of time it took to transfer data (cost of communication) increased compared to the amount of time it took to perform computations (cost of computation). Ex1008, 459–60. Thus, memory management techniques were already an area of focus for developers.

#### a) Using Pointers for Memory Management

62. Computers store information in memory. Memory is not an infinite resource, so programmers are careful to avoid storing data that will not be needed later. Further, memory is more limited the closer it is to the processor. A processor generally has a local memory cache with less space, and a larger memory (such as DRAM, which is main memory, SSD, which is Flash-based storage, or a hard drive) with more space. But transferring data to and from main memory or storage can be much slower than moving data in the local cache, and so programmers avoid unnecessary such transfers. Reading and writing data takes time, even in the processor cache, so programmers avoid unnecessary copying (which both reads and writes data).

63. The C programming language, first released in 1977, has long offered sophisticated tools to handle memory use and access. Data used during the operation of a program is called runtime data, and these are generally stored in memory cache.

The C programming language includes high-level functions to allocate and deallocate memory in the cache for variables, and it uses automatic processes that deallocate local variables when a function closes. *See*, *e.g.*, Ex1033, 86–116. To pass an array (i.e., data) to a function, C instructs programmers to create a pointer variable which stores the address of the array in the memory, and then pass that address to the function. *Id.*, 88–90. This approach allows the function to read data from the space in memory directly and to write data back to that same space directly. Without using pointers, a system would have to copy into a memory location accessed by the function and then copy the resulting data back at the end of the function, which would require a lot of time and system resources. *Id.*, 89. These design features reduce the number of unnecessary read/write cycles and reduce the risk that the processor cache will run out of memory.

64. Similar memory management features quickly became common among computer programming languages, particularly those developed for numerical simulation and computation. *See*, *e.g.*, Ex1012, 17–20.

**b) <u>Using Outputs as Inputs and Pointer Swapping</u>**

65. It was well known that in many numerical simulations and computations, the output of one computational step is used as the input for the next step. For example, in a simulation of a physical system over time, the result (output) of one time step is fed into the simulation as an input for the next time step, creating

a feedback loop that allows the model to evolve accurately over time. Prior to the ’461 patent, those skilled in the art had already developed effective solutions for implementing these kinds of computations in computers.

66. A simple approach is to copy the output data into a new array for use in the subsequent step, but this copying process incurs a performance cost due to the time and memory bandwidth required to duplicate potentially large datasets. A more efficient alternative—which was well known and used in the art prior to the ’461 patent—is to avoid copying altogether by swapping the pointers of the input and output data arrays. By simply redirecting the pointer that indicates which array to use as input, the simulation can proceed without duplicating data, significantly reducing overhead. This approach not only saves time but also minimizes memory usage, making it especially valuable in high-performance computing applications where memory and processing power are critical resources.

67. Programming techniques for swapping pointers existed in computer programming long before the ’461 patent. For example, one textbook relating to the C programming language teaches a “swap” function, which swaps *pointers* to values, rather than swapping the values themselves, for efficiency.

The way to obtain the desired effect is for the calling program to pass *pointers* to the values to be changed:

```
swap(
```

Since the operator & produces the address of a variable, `&a` is a pointer to `a`. In `swap` itself, the parameters are declared as pointers, and the operands are accessed indirectly through them.

```
void swap(int *px, int *py)  /* interchange *px and *py */
{
    int temp;

    temp = *px;
    *px = *py;
    *py = temp;
}
```

Ex1033, 88.

68. Prior to the ’461 patent, GPUs had been designed in ways to easily allow the above principles to be applied. For example, GPUs (such as NVIDIA’s) allowed shaders to render results to a buffer (memory) rather than a display. It was known to use these buffers to perform iterative computations, using the output of one rendering step as the input to the next, until a final frame was ready to render to the display (in the case of graphics) or final result was passed back to a CPU and/or system memory (in the case of GPGPU). It was well understood that programmers could use these buffers to solve iterative numerical simulations with limited read-write cycles between CPU and GPU. Ex1006, 501–08. One known way to accomplish this was through “render-to-texture,” through which the results of a program (shader) are written to local texture memory “so they can be used as input to future programs.” *Id.*, 501.

69. The concept of “pointer swapping” was well known in the graphics and GPGPU space prior to the ’461 patent as a way to leverage the output of one

computation as the input to the next. For example, in the book GPU Gems (which I understand is prior art to the '461 patent), NVIDIA employee and author Mark Harris provides C++ code for simulating a reaction-diffusion system on a GPU that includes a "swap()" function at the end of each iteration that allows the output of one step of the simulation to become the input of the next step. The code defines one buffer (*i.e.*, portion of memory) as the input (currentSource) and another as the output (currentTarget). That is, the input data for a step is stored in a memory location pointed to by the variable currentSource and the output data is saved in a memory location pointed to by the variable currentTarget. At the end of computational cycle, the function swap() swaps the pointers for the currentSource buffer with that of the currentTarget buffer, so that the data pointed to by currentTarget becomes the input pointed to by currentSource for the next cycle.

**Listing 31-2 *(continued)*. C++ Code to Set Up and Run a Reaction-Diffusion Simulation on the GPU**

```
    // Initialize the state of the simulation with values in array
    pugInitBuffer(rdBuffer, array);
}

void update_rd(){ // Call this every iteration
    pugBindStream(rdProgram, "concentration", rdBuffer, currentSource);
    PUGRect range(0, width, 0, height);
    pugRunProgram(rdProgram, rdBuffer, range, currentTarget);

    std::swap(currentSource, currentTarget);
}
```

Ex1006, 508 (annotated). GPU Gems contains several other examples of pointer swapping during numerical computations. *E.g.*, *id.*, 44 ("Then, we swap the roles of the two textures and iterate the process . . . ."), 713 ("swap rho and newrho pointers"

at the conclusion of each step of a "Conjugate Gradient Solver"), 743 ("To sort the entire field, we just call the next list and swap the buffers until we're done.").

70. In this art, these techniques were sometimes known as "double buffering" or "ping-pong rendering." For example, OpenGL's "double buffering" used two framebuffers (which were dedicated areas of memory)—one for rendering and one for display—and swaps pointers between them at the end of each frame. In fact, by 2004, OpenGL 2.1 supported quadruple buffering. Ex1018, 1. DirectX 9 likewise supported double buffering at least by 2003. Ex1019, 1, 18, 22–23; *see also* Ex1020, 17 (describing the "ping pong technique," in which the user "alternately use[s] the output of a given rendering pass as an input in the next one"); Ex1023, 323–24.

**c) <u>Transferring Data Between System Memories</u>**

71. Prior to the '461 patent, it was known that in numerical simulations and other computer applications, a user did not need to preserve all information from each time step. Rather, a user could elect when and how to store output data, including intermediate results of calculations.

72. A user running a numerical simulation would understand that runtime data is primarily stored in the processor's cache as long as it fits within the available cache size. If the cache capacity is exceeded, data is transferred to other levels of memory, such as DRAM, which offers larger storage but with increased access

latency. Since transferring data between different memory hierarchies incurs computational overhead in terms of time and resource management, the user must balance the need for storing output data with the cost of moving it. For example, the user might only require a steady-state solution at the final step of the simulation, minimizing storage needs. Alternatively, he or she might need to track the evolution of a concentration profile over time, choosing an appropriate step size to achieve a desired accuracy. In some cases, a variable step-size method might be employed to maintain numerical stability, while still ensuring output is recorded at regular intervals. In each scenario, the user configures the simulation to store output selectively, discarding or overwriting intermediate results as necessary. Ultimately, the user must strike an appropriate balance between data retention, computational efficiency, and storage constraints, based on the specific objectives of the simulation. *See, e.g.*, Ex1011, 139–43 (showing a selection of step size according to tolerance for error and demand for speed).

### 5. Artificial Neural Networks and Other Applications of GPGPU

73. By early 2005, GPUs had already been used for a variety of traditional scientific computing applications. This included, for example, solving systems of linear equations (Ex1008, 703–18); reaction-diffusion simulations (Ex1008, 505–08); protein folding and structure prediction (Ex1021, 8; Ex1008, 695–702); and

fluid dynamics and flow simulation (Ex1022, 1; Ex1008, 747–63), among many others. *See generally* Ex1008 ("Part VI: Simulation and Numerical Algorithms").

74. One significant application that arose from early GPGPU advancements was the ability to simulate artificial neural networks on GPUs. *See, e.g.*, Ex1023, 322–24; Ex1007, 1. An artificial neural network is a computer-based model designed to mimic the way the human brain processes information. In the human brain, neurons are organized into layers that work together to process information. Neurons communicate by sending signals to each other, forming pathways that support thinking, learning, and memory. Information flows through layers of neurons, with each layer refining and interpreting the data. As pathways are used more frequently, they strengthen, allowing the brain to recognize patterns. The collection of layers of neurons is referred to as a "neural network."

75. An "artificial neural network" mimics a natural neural network in a computer. The artificial neural network is made up of layers of nodes—each acting like a neuron—that receive data, perform calculations, and pass the results to the next layer of nodes. Every node is connected to nodes in the next layer with parameters called weights. By adjusting these weights, or connections, between these layers of nodes during training, the network gradually learns to recognize patterns and make decisions, much like how repeated experiences in the human brain strengthen neural pathways.

76. In one simple illustrative example, the inputs to a neural network are represented as a vector or matrix of values. Each input is then multiplied by a corresponding weight (representing the connection between layers of nodes). The sum of these weighted inputs is then calculated and that resulting vector is passed through an activation function that determines whether a node (neuron) should be activated (fire) based on the weighted sum of its inputs. Activation functions introduce non-linearity to the system, and enable the system to learn complex patterns. A bias term can be added to the weighted sum of inputs prior to passing through the activation function to shift the results to better fit the data. These computations are computed for all nodes in the layer, and the results are passed as inputs to the next layer.

77. An artificial neural network is usually trained by feeding a model large amounts of labelled data. For example, to identify handwritten numbers, the network would be provided with a large number of labelled images of numbers (positive examples) and other objects that are not numbers (negative examples). The computer simulating the neural network would then make predictions (*i.e.*, what number is represented in the image) by running the training data through its layers of nodes. The predictions are compared to the labels (the correct answers) to calculate an error. This error is propagated backward through the network—a process called backpropagation—which adjusts the internal weights to improve accuracy. Because

this process requires performing millions of complex calculations (like matrix multiplication) across many layers and iterations, it is computationally intensive.

78. Prior to the ’461 patent, it was known that the parallel processing power of GPUs was crucial for efficiently training these networks, as GPUs are particularly well-suited for the matrix operations and numerical calculations required for neural network training. Ex1023, 322–23 (“[M]any inner product operations can be replaced with matrix multiplication, which is more appropriate for GPU implementation.”). For example, before the ’461 patent, GPUs were already being utilized to power artificial neural networks in a variety of applications, including tasks such as image and pattern recognition (Ex1023), object detection (Ex1007, 1), medical diagnostics (Ex1026), and checkers (Ex1025, 6). The rapid progress in this field owed much to the computational capabilities provided by early GPGPU innovations, including innovations from NVIDIA and others. *See generally* Ex1008.

**6. Real-Time Data Input**

79. Prior to the ’461 patent, the idea of using GPUs to process real-time streams of input, such as video data from a camera, was being explored in numerous research and development contexts. For example, GPU-based computer vision was well developed, with “[t]he single-instruction, multiple-data (SIMD) capability of the GPU mak[ing] it suitable for running computer vision tasks.” Ex1008, 649.

80. It was thus well understood at the time that the parallel processing capabilities of GPUs made them particularly suitable for real-time tasks. Researchers and engineers had been experimenting with receiving real-time inputs, like live video streams from cameras, and sending them to GPUs for tasks such as image processing, object tracking, and motion detection. Ex1023; Ex1007. As one example, researchers had used artificial neural networks in a GPU to track a soccer ball in real time for robot soccer. Ex1007, 1.

### 7. Task and Resource Management on a GPU

81. GPGPU involves running many tasks concurrently within a single stage of processing and coordinating dependencies between stages of processing. Thus, efficient management of tasks assigned to the GPU and of system resources is critical for optimal utilization of a GPU's parallel processing capabilities. The operating system is the central software that manages hardware components in a computer, including CPUs and GPUs, ensuring that they work together smoothly and efficiently. One key part of this management is handled by the "scheduler," a mechanism built into the operating system that decides which tasks—whether they are individual threads or complete processes—get to use the computer's resources at any given time. *E.g.*, Ex1007, Ex1024, Ex1027. The scheduler works by assigning priorities and time slices to tasks based on factors like urgency, resource demands,

and overall system load, ensuring that critical operations run promptly while also balancing the needs of less urgent tasks.

82. Schedulers enable shared resource utilization by determining which tasks should run at any given moment and making necessary computational resources available when required. For instance, once all the resources needed for a specific task are ready, the scheduler can prioritize and select that task for execution. The scheduler also shifts access to and control over particular resources based on the task. Scheduling policies govern how these decisions are made and take into account factors like task priority, resource demands, and timing. These policies are designed to meet various objectives, such as minimizing latency or response time, maximizing throughput, or meeting deadlines in real-time systems.

83. Prior to the ’461 patent, one of the most widely used scheduling strategies was priority scheduling, where each task was assigned a priority value. Ex1027, 4:11–5:62. In the situation that a set of two or more tasks have the same priority value, the scheduling policy can shift to using round robin scheduling for that set of tasks, assigning time slices to each task in equal portions and in circular order so that each task has an equal chance of getting resources. *Id.* The scheduler would then use these priority levels to allocate resources, giving preference to higher-priority tasks. In systems that supported what is known as “preemption,” higher-priority tasks could interrupt lower-priority ones, temporarily suspending

their execution. When a task is preempted, the resources it was using are freed up for the higher-priority task. This dynamic allocation ensures that critical tasks are executed promptly while maintaining system responsiveness.

84. The concept of queuing tasks for executing was fundamental to task scheduling. When a resource required for a particular task is currently in use, the scheduler will queue the task for later execution. Schedulers queue tasks as they come in by assigning each task a position in the queue based on a scheduling algorithm. For example, if a GPU is currently executing a computation, the scheduler can queue the task by placing it in a specific position in the queue based on factors such as task priority and resource availability. The queuing of tasks allowed operating systems to handle multitasking effectively, balancing the load and ensuring that high-priority tasks received timely attention. By managing tasks that require access to the same data, computational resources currently in-use, or have data dependencies, schedulers optimize efficiency, balance computational loads, and improve data coherency in a system.

* * *

85. In sum, the concepts discussed in the ’461 patent are not new. Prior to the ’461 patent, the industry had already advanced and transformed GPUs into a general-purpose computing platform using well-known techniques such as SIMD parallelization, memory management, and scheduling. These techniques,

individually and collectively, allowed GPUs to perform computationally intensive tasks, like large scale numerical simulations such as artificial neural networks. NVIDIA and others led this effort, not Patent Owner. And NVIDIA published its methods (both in books and in patents) before Patent Owner filed its patent. Ex1008; Ex1017; Ex1027; Ex1034; Ex1035.

## V. THE '461 PATENT

### A. Summary of the '461 Patent

86. The '461 patent, titled "Graphic Processor Based Accelerator System and Method," was filed on December 29, 2020, and issued on March 14, 2023. U.S. Patent No. RE49,461 (Ex1001), Cover. It claims priority to a provisional application that was filed on September 25, 2006. *Id.*, p. 2.

87. The '461 patent is the last patent in a family of patents directed to an optimized system for performing general-purpose computations on graphics processing units, which are traditionally special-purpose processors. The '461 patent family relates to an "accelerator" for performing computations in a computer system, such as numerical simulations. *Id.*, 2:31–64. In the Abstract, the '461 patent is summarized as follows:

> An accelerator system is implemented on an expansion card comprising a printed circuit board having (a) one or more graphics processing units (GPUs), (b) two or more associated memory banks (logically or physically partitioned), (c) a specialized controller, and (d) a local bus providing signal coupling compatible with the PCI industry standards. The controller handles most of the primitive operations to set up and

> control GPU computation. Thus, the computer's central processing unit (CPU) can be dedicated to other tasks. In this case a few controls (simulation start and stop signals from the CPU and the simulation completion signal back to CPU), GPU programs and input/output data are exchanged between CPU and the expansion card. Moreover, since on every time step of the simulation the results from the previous time step are used but not changed, the results are preferably transferred back to CPU in parallel with the computation.

*Id.*, Abstract.

88. In the Background of the Invention, the ’461 patent explains that “Graphics Processing Units (GPUs) are found in video adapters (video cards) of most personal computers (PCs), video game consoles, workstations, etc. and are considered highly parallel processors dedicated to fast computation of graphical content.” *Id.*, 1:38–50. The patent further acknowledges that “manufacturers of GPUs have included general purpose programmability into the GPU architecture leading to the increased popularity of using GPUs for highly parallelizable and computationally expensive algorithms outside of the computer graphics domain.” *Id.*, 1:46–55. As an example, the ’461 patent says that “[n]umerical simulations, e.g., finite element analysis, of large systems of similar elements (e.g. neural networks, genetic algorithms, particle systems, mechanical systems) . . . can benefit from GPGPU computation.” *Id.*, 1:59–63.

89. The ’461 patent specification does not claim to have invented the performance of general-purpose computing on GPUs. *Id.*, 1:46–55. In fact, the ’461 patent states that GPGPU applications have “increased [in] popularity.” *Id.* The

patent, however, claims that GPGPU applications "are not able to achieve optimal performance" because "[t]here is overhead for graphics-related features and algorithms that are not necessary for these non-video applications." *Id.*, 1:53–55. To overcome this drawback, the '461 patent purports to invent a "hardware implementation" having "(a) one or more graphics processing units, (b) two or more associated memory banks that are logically or physically partitioned, (c) a specialized controller, and (d) a local bus providing signal coupling." *Id.*, 2:31–64. The '461 patent does not purport to have invented any specialized hardware. *Id.*, 14:31–35. Instead, the '461 patent describes a purportedly novel architecture. *Id.*, 2:31–64. This hardware implementation of "the computational stream of execution" replaces "thread and context initialization" with "hardware initialization." *Id.* This arrangement allegedly improves overall system performance because, rather than the system CPU, "[t]he controller handles most of the primitive operations needed to set up and control GPU computation." *Id.* As a result, "the CPU is freed from this function and is dedicated to other tasks." *Id.*

90. Figure 2 (below) is illustrative of the allegedly inventive "GPU accelerator system." *Id.*, 5:8–11. On the left-hand side of the system are "one or more CPU's 120," "main, or system memory 130," and "mass/non volatile data storage 140." *Id.*, 4:17–32. Those components connect to "expansion card 180" over one or more buses. *Id.* Expansion card 180 includes one GPU 240, memories 210 (shader

memory bank) and 250 (texture memory bank), and "accelerator controller 220." *Id.*, 5:8–24. In the embodiment of Figure 2, "one memory bank, the shader memory bank 210, is used to store the shaders needed for the execution of the required computations and the parameters that are common for all computational elements," and "[t]he second memory, the texture memory bank 250, is used to store all the necessary data that are specific for every computational element (including, but not limited to, input data, output data, intermediate results, and parameters)." *Id.*, 5:55–64. The '461 patent explains that "the term 'texture' in this document refers to a data array unless specific otherwise and each element of the texture is a pixel of color information" and "the term 'shader' in this document refers to a GPU program unless specified otherwise." *Id.*, 5:25–36.

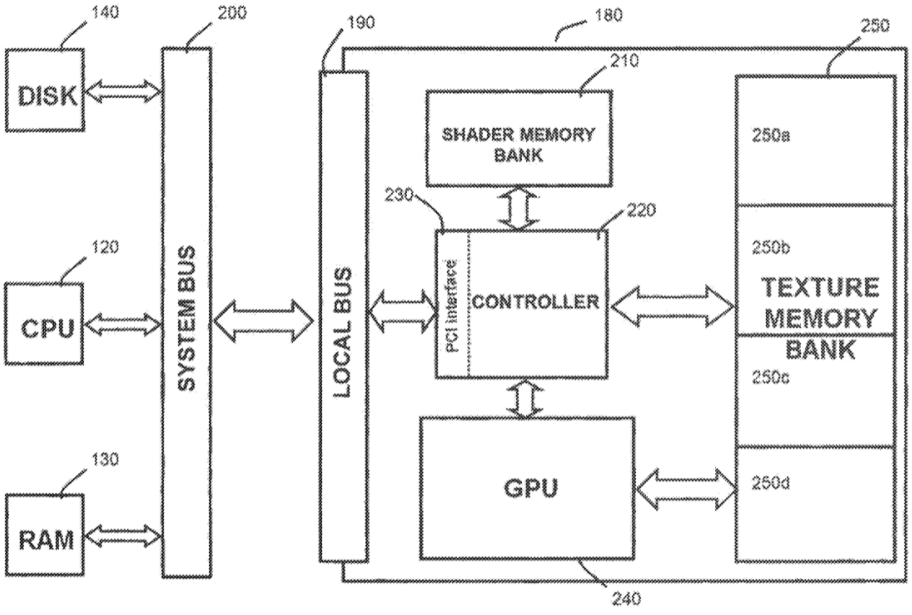



FIG. 2

*Id.*, Fig. 2.

91. Accelerator controller 220 is described as "specifically designed" and can be implemented as "field programmable gate array (FPGA) logic, or custom built application-specific (ASIC)," or "even fully implemented in software." *Id.*, 5:16–24. The controller "commands the storage and retrieval of arrays of data," execution of GPU programs," and "data transfer between the system bus 200 and the expansion card 180." *Id.*, 5:25–36.

92. The "system comprises three 'peripheral' components: user interactive hardware, disk hardware, and computational hardware." *Id.*, 2:11–21. The CPU establishes communication and synchronization between the peripheral components, and "[e]ach of the peripherals is preferably controlled by a dedicated thread that is executed in parallel with minimal interactions and dependencies on the other threads." *Id.* These interactions and dependencies are controlled and minimized by scheduling tasks (e.g., for simulating artificial neural networks) for execution at certain times.

93. The '461 patent claims, however, recite maintaining separate instruction streams (or threads) for separate data types, which the patent refers to as "user interaction" and "computational" streams. *Id.*, 2:11–19, 7:60–8:6. Stream processing, which is taught in Buck and GPU Gems, was a basic, fundamental concept underlying GPU programming in the prior art.

94. The claims also recite the implementation of artificial neural network computations on a GPU. *Id.*, 1:59–63. Long before the '461 patent, researchers and companies such as NVIDIA and Microsoft and had established techniques for performing general-purpose computing on GPU-accelerated systems, including artificial neural networks, and developed specialized hardware to manage and control the processes. For example, Buck—a patent application publication assigned to Microsoft and not considered during prosecution—describes the training and simulation of artificial neural networks on a GPU-accelerated system. Buck also explains that the GPU and CPU in GPU-accelerated systems maintain independent instruction streams, data paths, and dedicated memory.

95. The claims further recite the basic, fundamental concept of task scheduling and resource management in a GPU-accelerated system: prioritizing currently executing computations over user commands that may affect the computations to "avoid data corruption" and prioritizing the execution of user commands when no computations are being performed. Ex1001, 7:60–9:52. The claims ensure that the CPU cannot corrupt the data being used by the GPU for its calculations by "shifting control of a data exchange" to the GPU while it is executing a calculation and then back to the CPU after the calculation is done or interrupted. Similarly, the claims recite "queueing" commands intended for the GPU while the

GPU is performing other calculations so that they are ready when the GPU is next available.

96. In other words, when the system is not running an artificial neural network simulation, a user can freely provide input by setting and editing parameters and elements of the artificial neural network and sources of external inputs, specifying and changing display or storage parameters, and starting and stopping a simulation. *Id.* Once the user "starts the simulation" and the processing unit begins executing computations, the computational stream "takes the priority." *Id.* Although the user "retains the ability to interrupt the simulation, change the input, or change the display properties of the framework," these "interactions are queued to be performed at times determined" by the computational stream to "avoid the corruption of data." *Id.* When execution of the computations is complete or interrupted, control shifts from the computational stream to the user interaction stream. *Id.* For example, when no simulation is running, the user can set and edit parameters and elements of the artificial neural network, change input sources, and edit display or storage parameters. *Id.*

97. These concepts, however, were well known and used long before the '461 patent and disclosed in patents and publications from NVIDIA, Microsoft, and others. By the time of the '461 patent, resource and data management in a GPU-accelerated system were universal. Wilt, which was not cited or discussed during

prosecution of the ’461 patent, describes managing computational resources and data access in a GPU-accelerated system by scheduling tasks based on factors such as task priority and resource availability. In particular, Wilt teaches that commands received while the GPU executes computations are processed and queued for execution at a later time.

98. The dependent claims recite only basic elements like deciding which equations to use, setting the parameters for the equations, or deciding which outputs to save to disk or display on a screen.

99. As I discuss in detail below, in my opinion, the combinations of references raised in the Petition render claims 21–41 obvious.

### B. Summary of the Prosecution History of the ’461 Patent

100. The ’461 patent originated from reissue Application No. 17/136,343 (“the ’343 application”) filed on December 29, 2020. Ex1002, 1–22. The ’461 patent claims priority from Provisional Application No. 60/826,892 filed on September 25, 2006, which is nearly 14 years prior to the filing of the ’343 application. *Id.*, 109.

101. The applicant originally filed a number of preliminary amendments adding claims to the issued claims of U.S. Patent No. 9,189,828 and U.S. Patent No. RE48,438. *Id.*, 39–65, 142–158. The applicant also included reissue application declarations that characterized the claims as “correct[ing] Applicant’s error by claiming the ‘user interaction stream’” that “has not been claimed to date.” *Id.*, 54–

64. Eventually, in response to the Examiner's objections, the applicant cancelled all issued claims of U.S. Patent No. 9,189,828 and included an entirely different set of claims. *Id.*, 289–298.

102. At the time of the first Office Action, representative claim 58 (eventually renumbered to claim 21) recited:

> A method of executing computations representing an artificial neural network on a computer system comprising at least one central processing unit (CPU), a processing unit, a first memory partition, and a second memory partition, the method comprising:
>
> executing, by the at least one CPU, a user interaction stream, the user interaction stream controlling transfer of inputs to the artificial neural network to the first memory partition and the second memory partition;
>
> executing, by the processing unit, a computational stream, the computational stream controlling data exchange between the user interaction stream and the computational stream during execution of the computations representing the artificial neural network;
>
> shifting priority from the user interaction stream to the computational stream in response to starting execution of the computations representing the artificial neural network; and
>
> shifting priority from the computational stream to the user interaction stream in response to completion or interruption of the computations representing the artificial neural network.

*Id.*, 153.

103. In the first, non-final Office Action, the Examiner rejected this new set of claims as obvious over U.S. Patent No. 7,219,085 to Buck in view of U.S. Patent No. 7,477,256 to Johnson, and U.S. Patent No. 7,525,547 to Diard. *Id.*, 207–223. The Office Action included limited remarks.

104. In an email confirming an interview with the Examiner, the applicant stated that "We see that the background of Buck mentions that 'GPUs are essentially stream processors' and that CPUs and GPUs have instruction streams, but we don't see anything about user interaction streams or computational streams.[] It is also not clear to us which portions of Buck, Johnson, and Diard disclose or suggest shifting priority between user interaction and computational streams." *Id.*, 287.

105. In its response to the first Office Action, the applicant noted in a summary of the telephonic interview with the Examiner, that they "discussed differences between the claimed technology and the cited references and possible directions for claim amendments," and that "[n]o agreement was reached." *Id.*, 295.

106. The applicant also amended what issued as claim 21 by (1) replacing the language "shifting priority" with "shifting control of data exchange between the user interaction stream and the computational stream," and (2) adding "queueing a user command received by the user interaction stream during execution of the computations representing the artificial neural network." *Id.*, 294. The applicant also amended what issued as claim 33, adding "a controller, operably coupled to the at least one CPU and the processing unit, to queue user interactions received by the user interaction stream during execution of the computations representing the artificial neural network for performance at times selected to avoid data corruption." *Id.*

107. The applicant filed new reissue declarations stating that "Applicant erred by claiming less than it had a right to claim, and that claim 21, for example, corrects this error because it does not recite 'an accelerator' as in claim 12" of parent US Patent No. 9,189,828. *Id.*, 345–354. The issued claims of the '461 patent remove not only any reference to "an accelerator," but also to "a graphics processing unit" in all but three dependent claims. *Id.*, claims 21–27, 31–41.

108. The Examiner allowed the claims on November 8, 2022. *Id.*, 359–367. In the reasons for allowance, the Examiner simply stated that the prior art did not "teach or fairly suggest," the claimed combination of limitations. *Id.* The '461 patent thus issued on March 14, 2023, with 21 claims. *Id.*, 392.

### C. Overview of the Challenged Claims

109. There are two independent claims in the '461 patent: claims 21 and 33. Claim 21 is a method apparatus claim and claim 33 is a system claim.

110. I understand that Petitioners are challenging all of the claims in the '461 patent (the "Challenged Claims"). I have analyzed each of these claims.

| | |
|---|---|
| 21[pre] | A method of executing computations representing an artificial neural network on a computer system comprising at least one central processing unit (CPU), a processing unit, a first memory partition, and a second memory partition, the method comprising: |
| 21[a] | executing, by the at least one CPU, a user interaction stream, the user interaction stream controlling transfer of inputs to the artificial neural network to the first memory partition and the second memory partition; |

| | |
|---|---|
| 21[b] | executing, by the processing unit, a computational stream, the computational stream controlling data exchange between the user interaction stream and the computational stream during execution of the computations representing the artificial neural network; |
| 21[c] | shifting control of a data exchange between the user interaction stream and the computational stream to the computational stream in response to starting execution of the computations representing the artificial neural network; |
| 21[d] | shifting control of the data exchange between the user interaction stream and the computational stream to the user interaction stream in response to completion or interruption of the computations representing the artificial neural network; |
| 21[e] | queueing a user command received by the user interaction stream during execution of the computations representing the artificial neural network; and |
| 21[f] | executing the user command during execution of the computations representing the artificial neural network at times determined by the computational stream. |
| 22 | The method of claim 21, wherein the user interaction stream controls the data exchange between the user interaction stream and the computational stream outside of execution of the computations representing the artificial neural network. |
| 23 | The method of claim 21, wherein executing the user interaction stream comprises: controlling setting and editing of computational elements of the computations representing the artificial neural network. |
| 24 | The method of claim 21, wherein executing the user interaction stream comprises: controlling setting and editing of parameters of the computations representing the artificial neural network. |
| 25 | The method of claim 21, wherein executing the user interaction stream comprises: controlling setting and editing of parameters of the inputs to the artificial neural network. |
| 26 | The method of claim 21, wherein executing the user interaction stream comprises: specifying an output to be saved to disk and/or displayed on a screen. |

| | |
|---|---|
| 27 | The method of claim 21, wherein executing the user interaction stream comprises: parsing elements to be used in the computations representing the artificial neural network. |
| 28 | The method of claim 27, wherein the processing unit comprises a graphics processing unit (GPU) and executing the user interaction stream further comprises: converting the elements into GPU programs. |
| 29 | The method of claim 28, wherein executing the user interaction stream comprises: compiling the GPU programs. |
| 30 | The method of claim 29, wherein executing the user interaction stream comprises: transferring the GPU programs to the second memory partition. |
| 31 | The method of claim 21, further comprising: executing, by the at least one CPU, a data output stream, the data output stream controlling transfer of outputs of the computations representing the artificial neural network to disk. |
| 32 | The method of claim 21, further comprising: generating the inputs with a video camera during execution of the computations. |
| 33[pre] | A system for executing computations representing an artificial neural network, the system comprising: |
| 33[a] | a camera to acquire input data for the artificial neural network; |
| 33[b] | a first memory partition; |
| 33[c] | a second memory partition; |
| 33[d] | at least one central processing unit (CPU), operably coupled to the camera, the first memory partition, and the second memory partition, to execute a user interaction stream, the user interaction stream controlling transfer of the input data acquired by the camera to the first memory partition and the second memory partition during execution of the computations representing the artificial neural network; |

| | |
|---|---|
| 33[e] | a processing unit, operably coupled to the first memory partition, the second memory partition, and the at least one CPU, to execute a computational stream, the computational stream controlling transfer of the input data from the first memory partition and the second memory partition during execution of the computations representing the artificial neural network, the execution of the computations representing the artificial neural network occurring while the camera is acquiring the input data; and |
| 33[f] | a controller, operably coupled to the at least one CPU and the processing unit, to queue user interactions received by the user interaction stream during the execution of the computations representing the artificial neural network for performance at times selected to avoid data corruption. |
| 34 | The system of claim 33, wherein the user interactions cause interruption of the computations representing the artificial neural network. |
| 35 | The system of claim 33, wherein the user interactions cause a change in inputs to the artificial neural network. |
| 36 | The system of claim 33, wherein the user interactions cause a change in display properties of an output of the computations representing the artificial neural network. |
| 37 | The system of claim 33, wherein the controller is configured to request the input data during the execution of the computations representing the artificial neural network. |
| 38 | The method of claim 21, wherein the user command causes interruption of the computations representing the artificial neural network. |
| 39 | The method of claim 21, wherein the user command causes a change in the inputs to the artificial neural network. |
| 40 | The method of claim 21, wherein the user command causes a change in display properties of an output of the computations representing the artificial neural network. |

| | |
|---|---|
| 41 | The method of claim 21, wherein, during execution of the computations representing the artificial neural network, the computational stream controls the data exchange between the user interaction stream and the computational stream by requesting the inputs to the artificial neural network. |

111. For the reasons I detail below, it is my opinion that each of these claims is unpatentable over the prior art discussed below.

## VI. PRIOR ART CONSIDERED

112. In this Declaration, I primarily discuss five prior art references with exemplary citations identifying the relevant features related to the claim language of the Challenged Claims. I describe combinations of these prior art references that, when combined, render the Challenged Claims obvious. I also discuss other documents and references to help establish what was known to a POSITA prior to the ’461 patent. It is my opinion that the Challenged Claims are rendered obvious in light of the prior art specifically discussed in this Declaration, in view of the knowledge of a POSITA at the time of the ’461 patent.

113. I understand that Patent Owner Neural AI, LLC (“Patent Owner”) asserts that it conceived of the purported inventions claimed in the ’461 patent at least as early as February 25, 2005. I have been asked to assume that the references discussed in this Declaration are prior art to the ’461 patent. In this Declaration I offer no opinions regarding whether Patent Owner is in fact entitled to a conception date prior to the filing date of the ’461 patent.

### A. General Overview of the Primary Prior Art References

114. The following sections provide a brief description of the anticipating prior art systems and software that invalidate the '461 patent. This discussion is not meant to be exhaustive, and my full element-by-element analysis is provided in more detail below in the analysis section. I cite representative portions of the prior art references to make my point below, though in many instances there are other portions of the prior art references that reinforce the same points. I have personally reviewed the references cited in this Declaration.

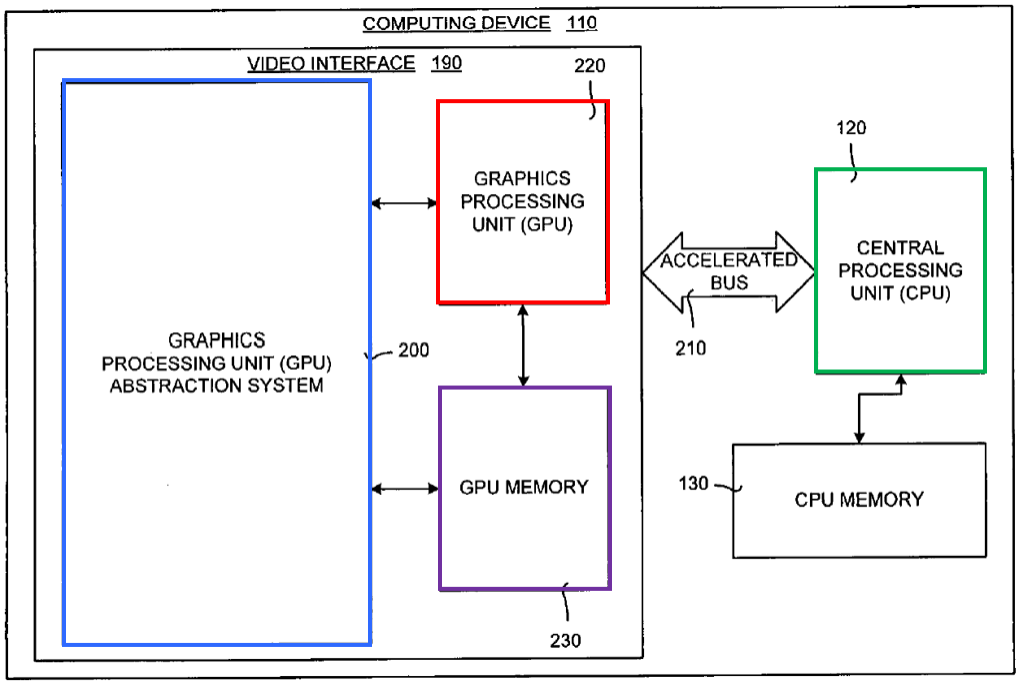
#### 1. Summary of Buck

115. U.S. Patent Application Publication No. 2005/0197977 ("Buck") is titled "Optimizing Performance of Graphics Processing Unit for Efficient Execution of General Matrix Operations" and published on September 8, 2005 from a non-provisional application filed on June 25, 2004. Ex1004, Cover. Buck claims priority from a provisional application filed December 9, 2003. *Id.* Buck is assigned to Microsoft Corporation. *Id.*

116. Buck explains that GPUs were already widely used for non-graphics processing like artificial neural network as a result of their "increased processing power and programmability" as of its filing in 2004. Ex1004, [0034]. Buck discloses a computer system for optimizing matrix operations on a GPU that allows a GPU to process and execute "operations traditionally handled by the CPU," such as the

training and testing, or simulation, of a "pair of layers in a neural network." *Id.*, [0011], [0054]; *see also* [0029]–[0031].

117. As I have illustrated below in Figure 2 (annotated below), Buck's computing device 110 includes CPU 120 (green), GPU 220 (red), GPU optimization system 200 (blue), and GPU memory 230 (purple). *Id.*, [0052]–[0057].




*Id.*, Fig. 2 (annotated).

118. Buck describes the GPU as "a prominent component of the PC with its own dedicated path to main CPU memory," "their own instruction streams, datapaths, and dedicated memory." *Id.*, [0003], [0008]. Buck's system "take[s] full advantage of the new flexibility introduced by programmable pixel shaders of GPUs" and "parallel architecture of the GPU" to train and use machine learning algorithms, such as neural networks, "an order of magnitude faster on GPU than on

CPU." *Id.*, [0011]. The pixel shaders can be programmed with "user defined function[s]" to "implement the various primitive operations used in machine learning algorithms and other non-graphics applications." *Id.*, [0011], [0059], [0067].

119. In my opinion, Buck is enabled, in that a POSITA exercising reasonable diligence could make and use the invention described in Buck without undue experimentation. For example, Buck provides 16 pages of disclosure and several figures that a POSITA could have relied on to develop an operational system.

**2. Summary of Wilt**

120. U.S. Patent Application Publication No. 2003/0140179 ("Wilt") is titled "Methods and System for Managing Computational Resources of a Coprocessor in a Computing System" and published on July 24, 2003 from a non-provisional application filed on January 4, 2002. Wilt is assigned to Microsoft.

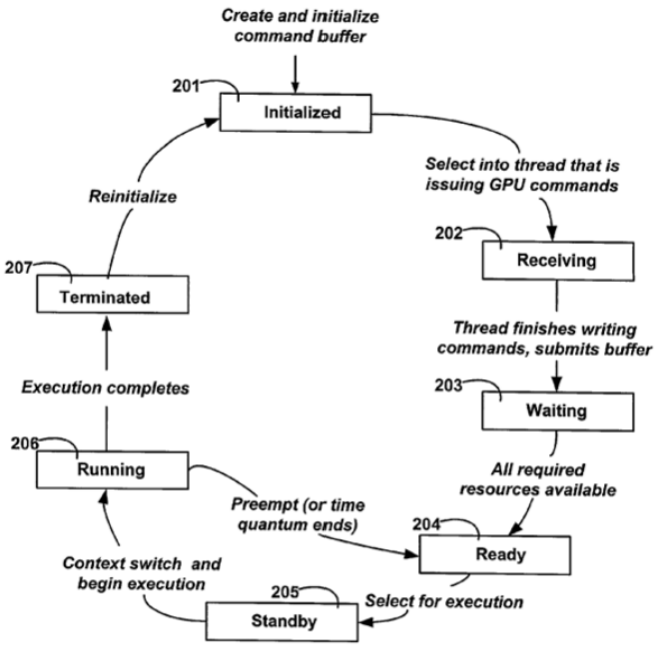
121. Wilt explains that, because "graphics processors have become significantly more functional," there is a need for "careful management" of the computational resources of the graphics processor to "deliver a high quality end user experience." Ex1005, [0005], [0009]. To address this need, Wilt discloses systems and methods for "efficient and effective management of computational resources of graphics processor(s)," in a computing system having co-processors (*e.g.*, a host processor CPU and a co-processor GPU). *Id.*, [0009].

122. Wilt's system "enables multiple threads of execution" to "efficiently share the computational resources available in the system." *Id.*, [0010]. In particular, Wilt discloses a system for constructing multiple hardware-specific commands, or "command buffers," in parallel and scheduling and queuing the command buffers for dispatch through the use of a scheduler. *Id.*, [0010], [0107], [0126]. The scheduler queues command buffers for execution according to resource availability, command buffer priority, and real-time requirements of the system. *Id.*

123. Wilt's GPU is controlled by command buffers sent by its host CPU. *Id.*, [0095]. For example, "a process, or a thread in a process, that implements some specific graphical application" in Wilt's computer system can use an application programming interface (API) call to request an action from the GPU. *Id.*, [0098]. That API call can be translated into a hardware-specific command that is written into a command buffer. *Id.*, [0097]–[0105]. Once the command buffer is full or the command has been completely written into the command buffer, the command buffer enters a waiting state until various waiting criteria are fulfilled. *Id.*, [0106]–[0107]. For example, if the command buffer requires a number of resources to be executed, the command buffer remains in the waiting state until all of those resource are available, at which time the command buffer enters a ready state and can be submitted to the GPU. *Id.* A component of Wilt's operating system, the scheduler, "decides which command buffer should be dispatched" and "has a great deal of

flexibility in deciding," considering factors such as "the priority, if any, of the command buffer, the priority of the requesting thread, the real-time requirements of the system, and so on." *Id.*, [0107], [0146].

**FIG. 2**




Ex1005, Fig. 2 (the "life cycle of a command buffer").

124. The command buffer "remains in the running state 206 until some event preempts execution of the command buffer or the entire contents of the command buffer have been read and executed." *Id.*, [0108]. If the entire command buffer has been executed, it enters the "terminated state" and "becomes available for the system to reinitialize and make available to another thread in the initialized state." *Id.*, [0109]. If the command buffer is, instead, preempted, "it is assumed that the hardware has the capability to save the hardware context in such a way that execution

of the command buffer can be resumed where the preemption occurred," and the buffer can "be recycled into the ready state 204 and be re-submitted to the hardware to resume from where it stopped." *Id.*, [0110].

125. Wilt's scheduling and resource management system prevents its GPU from "wast[ing] valuable time waiting for another command buffer to be submitted," by queueing the next command buffer for dispatch while the GPU is executing a command buffer. *Id.*, [0126]. When the command buffer is ready for execution, the scheduler "submits the command buffer to the hardware." *Id.*, [0108].

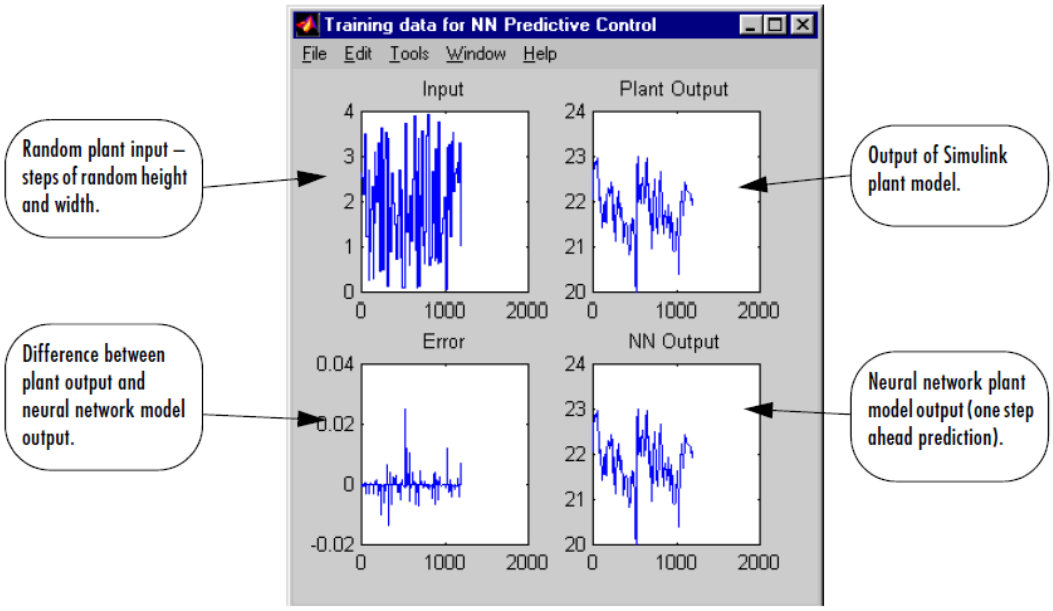
126. In my opinion, Wilt is enabled, in that a POSITA exercising reasonable diligence could make and use the invention described in Wilt without undue experimentation. For example, Wilt provides 18 pages of disclosure and several figures that a POSITA could have relied on to develop an operational system.

**3. Summary of nnet**

127. "Neural Network Toolbox: For Use with MATLAB" ("nnet") was published by The MathWorks, Inc. in 2004.

128. nnet is a user manual for MathWorks' Neural Network Toolbox, a collection of functions for creating and interacting with artificial neural networks in MATLAB. Ex1006. In addition to describing the functions available for implementing neural networks within MATLAB, nnet also explains how to use its graphical user interface to perform standard user interface functions, such as

"creat[ing] a network, view[ing] it, train[ing] it, simulat[ing] it and export[ing] the final results," and "import[ing] data…for use in the GUI." *Id.*, 91–100, 237–244.




*Id.*, 218.

129. MATLAB is a well-known, widely used platform that "integrates computation, visualization, and programming in an easy-to-use environment." Ex1031, 1–2. Typical uses for MATLAB include modeling, simulation, and scientific and engineering graphics. *Id.* The MATLAB system includes "five main parts": the development environment, the mathematical function library, the language itself, graphics, and the API. *Id.*, 1–3.

130. Well before the '461 patent, The MathWorks had established itself as a well-known and reputable company within the engineering and scientific communities, particularly for its MATLAB and Simulink products. Professionals in

the field were likely familiar with The MathWorks's online resources and would have instinctively checked their website for relevant documentation. Additionally, by 2004, the internet had become a primary resource for information retrieval, with search engines like Google making it straightforward to find specific documents. Given the increasing reliance on digital documentation, it was common practice for researchers in the field to seek out user manuals, software guides, and technical support online, knowing that official websites like The MathWorks's would be reliable sources for accurate and up-to-date information.

131. In my opinion, a POSITA looking to implement an interactive GUI for simulating artificial neural networks could have readily located nnet. The POSITA would have been familiar with MATLAB and would have been aware that, prior to the '461 patent, that a Neural Network Toolbox with ready-made functions for training, testing, and simulating artificial neural networks was available for use with MATLAB for years. Indeed, this Neural Network Toolbox was one of a few commercial solutions available for implementing artificial neural networks, and the POSITA would have been motivated to search for or directly navigate to The MathWorks's website to review documentation of the Neural Network Toolbox for use with MATLAB. Once on The MathWorks's website, the POSITA could search for keywords such as "neural network" or browse the products and services that The MathWorks offered to find the Neural Network Toolbox. Documentation for the

Neural Network Toolbox on The MathWorks's website includes the User Guide (Ex1006) I discuss.

132. Additionally, in my opinion, a POSITA exercising reasonable diligence that was searching for references regarding neural networks would have located nnet no later than around October 2004 by keyword searching for terms found in the full text of nnet, such as "neural network." A POSITA in the late 2004–2005 timeframe could have also used search terms, alone or in certain combinations, such as the indexing terms for nnet, including keywords in the title of the document, such as "neural," "network," and "MATLAB" and keywords describing the contents of the document, such as "neural networks."

133. In my opinion, nnet is enabled, in that a POSITA exercising reasonable diligence could make and use the invention described in nnet without undue experimentation. For example, nnet provides more than 800 pages of disclosure and several figures, including example code, that a POSITA could have relied on to develop an operational system.

**4. Summary of ANN**

134. "Artificial Neural Network Computation on Graphic Process Unit" ("ANN") was published in the Proceedings of the IEEE International Joint Conference on Neural Networks (2005).

135. ANN explores the application of GPUs for artificial neural network computation. *Id.* ANN begins by acknowledging that artificial neural networks are "widely used in pattern recognition," and are particularly beneficial where "computational load is very heavy" or where "real time process[ing] is required." *Id.* ANN explains the significant potential for using GPUs for artificial neural networks due to their inherently parallel and repetitive nature. *Id.* ("The computation result shows that ANN computing on GPU is much faster than on standard CPU when the neural network is large.").

136. In one example, ANN applies these principles by implementing a "[t]hree layer MLP neural network" with an "input layer consist[ing] of seven nodes," a "hidden layer consist[ing] of three nodes," and an "output layer consist[ing] of just one node" to recognize a soccer ball (object recognition) and "trace the ball in real time" (feature tracking). Ex1007, 2–5. The "trained MLP" can then be used to assist soccer-playing robots. Ex1007, 2–5. There are "two main computation steps for" the MLP, the first of which is "matrix multiplication," which is a key mathematical operation in any artificial neural network, and the second of which is a "sigmoid function calculation." *Id.*, 3. Video data of the soccer ball gathered in real-time would be transferred by the CPU to GPU memory for processing, and in other implementations, the GPU may have a "video in" function

such that video data is "retrieved directly from [the] camera and store[d]" in GPU memory. *Id.*, 5.

137. In my opinion, ANN is enabled, in that a POSITA exercising reasonable diligence could make and perform the computer vision algorithm implemented on a GPU as disclosed in ANN without undue experimentation. For example, ANN provides implementation details and identifies specific hardware that a POSITA could have relied on to develop an operational system.

138. Additionally, in my opinion, a POSITA, exercising reasonable diligence and searching for references regarding neural networks would have located ANN around January 4, 2006. For example, a POSITA would have located ANN in the library by searching for the conference specifically or a term such as "neural networks."

**5. Summary of GPU Gems**

139. GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation ("GPU Gems") was published in March 2005 by Petitioner NVIDIA.

140. GPU Gems contains 48 chapters written by experts in the field that describe the state of the art in GPU programming. Ex1008, xxxi–xxxii. Most relevant here, Part IV is a "Primer" on "General-Purpose Computation on GPUs," which "aims to provide a gentle introduction to the world of general-purpose

computation on graphics processing units, or 'GPGPU.'" *Id.*, xx, 453. The chapters of Part IV include "details of how to solve general-purpose problems using the GPU." *Id.*, 454.

141. Much of my analysis in this declaration relates to Chapter 40, titled "Computer Vision on the GPU," which explores "methods of efficiently mapping common mathematical operations of computer vision onto modern computer graphics architecture." *Id.*, 649. As explained below, the GPU programming techniques in GPU Gems explicitly include the concept of receiving and processing "real-time" input from a video camera, as claimed in the '461 patent. *Id.*, 649–66.

142. Chapter 40 explains that the "single-instruction, multiple-data (SIMD) capability of the GPU makes it suitable for running computer vision tasks, which often involve similar calculations operating on an entire image." *Id.*, 649–50. In this way, GPUs can be used "to accelerate computer vision computation and free up the CPU for other tasks." *Id.*, 650. Each of the computer vision algorithms discussed can be used to "create a fully interactive, real-time computer-mediated reality, in which the environment of a user appears to be augmented, diminished, or otherwise altered." *Id.*, 651. Additionally, the chapter noted that all of the methods discussed were publicly available in "the OpenVIDIA GPU computer vision library, which can be used to create real-time computer vision programs." *Id.*, 650–51.

143. One example application "which can be mapped effectively onto graphics hardware" is "feature tracking." *Id.*, 650–51. A "feature" in an image is an area around a point with "some higher-than-average amount of 'uniqueness'" that makes "the point easier to recognize in subsequent frames of video." *Id.*, 661. Feature vectors are used to track the "same point in different images and can be extended to more generalized object recognition techniques." *Id.* Chapter 40 explains how "[t]he GPU is used to detect and track distinctive points (features) in real time as the camera pans," *id.*, 661, and even included code for a "fragment program for calculating feature vectors," *id.*, 663.

144. In my opinion, GPU Gems is enabled, in that a POSITA exercising reasonable diligence could make and perform the GPGPU computing algorithm disclosed in GPU Gems (including the computer vision algorithms of Chapter 40) on GPU-accelerated hardware. GPU Gems includes an entire section on GPU-accelerated computations, with specific examples and code that could be used and notes that fully functional code was publicly available. A POSITA exercising reasonable diligence would have been able to perform the examples disclosed in GPU Gems, including on the hardware of Buck and Wilt as described herein.

145. Additionally, in my opinion, a POSITA, exercising reasonable diligence and searching for references regarding programming GPUs, would have located GPU Gems in July 2005. For example, a POSITA would have located GPU

Gems by searching for keywords like "GPU" and "programming," which are contained in the searchable title, or "[c]omputer graphics" or "[r]eal-time programming," which are contained in the searchable subject headings for GPU Gems.

## VII. ANALYSIS OF THE CHALLENGED CLAIMS AGAINST THE PRIOR ART

146. Based upon my review of the prior art, particularly the references detailed in this declaration, it is my opinion that the subject matter of each of the Challenged Claims would have been obvious to one of ordinary skill in the art in view of the known prior art references.

147. Based upon my analysis, it is my opinion that the Challenged Claims of the '461 patent are rendered obvious by the following prior art grounds. All obviousness grounds include the knowledge of a POSITA:

| Ground | Claims | Basis for Rejection |
|---|---|---|
| 1 | 21–41 | Obviousness over Buck in view of Wilt |
| 2 | 23–26, 34–36, 38–40 | Obviousness over Buck in view of Wilt and nnet |
| 3 | 32–37 | Obviousness over Buck in view of Wilt and ANN |
| 4 | 32–37 | Obviousness over Buck in view of Wilt and GPU Gems |
| 5 | 34–36 | Obviousness over Buck in view of Wilt, ANN and nnet |
| 6 | 34–36 | Obviousness over Buck in view of Wilt, GPU Gems, and nnet |

### A. Ground 1: Buck and Wilt Render Obvious Claims 21–41

148. In my opinion, claims 21–41 of the ’461 patent would have been obvious over Buck and Wilt.

#### 1. Motivation to Combine Buck and Wilt

149. In my opinion, a POSITA would have been motivated to combine Buck with Wilt. Before the ’461 patent, it was well known that resource and task management methods could be implemented in GPU-accelerated systems to optimize efficiency, balance computational loads, and manage data dependencies. Section IV.B. These concepts had already been applied by companies in the industry, including Microsoft, to which both of Buck and Wilt are assigned. A POSITA reviewing Wilt would have been motivated to implement Wilt’s scheduling and task

management system in Buck's GPU-accelerated system for implementing neural networks to optimize the performance of the system.

150. As I have discussed above, Wilt describes managing the computational resources of a GPU to facilitate multitasking and the execution of multiple instruction streams. To the extent that Buck does not focus on how its system schedules and manages tasks on the GPU, in my opinion, a POSITA would have been motivated to implement Wilt's resource management method within Buck's co-processor system.

151. Buck discloses a computer system that uses GPU-acceleration for the training and testing of an artificial neural network. Ex1004, [0011], [0054], [0029]–[0031]. Buck explores "performance optimization" of general matrix operations on a GPU. Ex1004, [0035]. Certain "machine learning algorithms and other non-graphics applications" that require "repeated operations on large" matrices and vectors "greatly benefit" from migration to a GPU. *Id.*, [0010]–[0011]. A GPU's specialized parallel architecture allows these algorithms to "run an order of magnitude faster on GPU than on CPU," and Buck's system "port[s] some or all of the CPU processing to the GPU." *Id.* Buck does not focus on how its system schedules and manages tasks on the GPU. Wilt does.

152. In particular, Wilt describes managing the computational resources of a GPU to facilitate multitasking and the execution of multiple instruction streams as

claimed. Ex1005, [0009]–[0010], [0107], [0126]. In my opinion, a POSITA would have been motivated to implement Wilt's resource management method within Buck's co-processor system and would have reasonably expected that the combination would enable multiple tasks to "efficiently share the computational resources available in the system," as Wilt teaches. Ex1005, [0010].

153. A POSITA reviewing Buck's CPU and optimized GPU computing system would have looked to other art describing the resource management operations of GPUs and CPUs. It was common practice for a patent assignee to focus on describing one set of features in one patent application and another set of features in a different patent application, and, in my opinion, a POSITA would have expected to look to different patent applications to find different disclosures related to the same, or a similar, system. Therefore, another Microsoft patent application publication, such as Wilt, would have been a natural reference for a POSITA to consult. Buck focuses on the use of the GPU for complex computations instead of the CPU, Wilt focuses on the details of managing resources between the CPU and the GPU. Both Buck and Wilt are Microsoft patent application publications. Both references disclose a system having a host CPU and a co-processor GPU. And, as shown below, Buck's and Wilt's example computing environment are nearly identical.

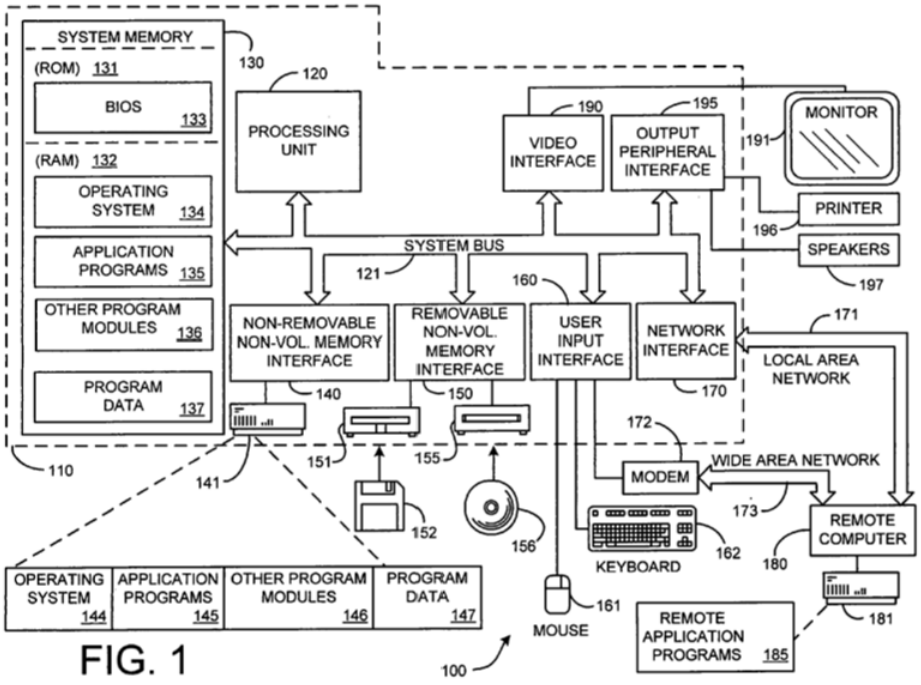



FIG. 1

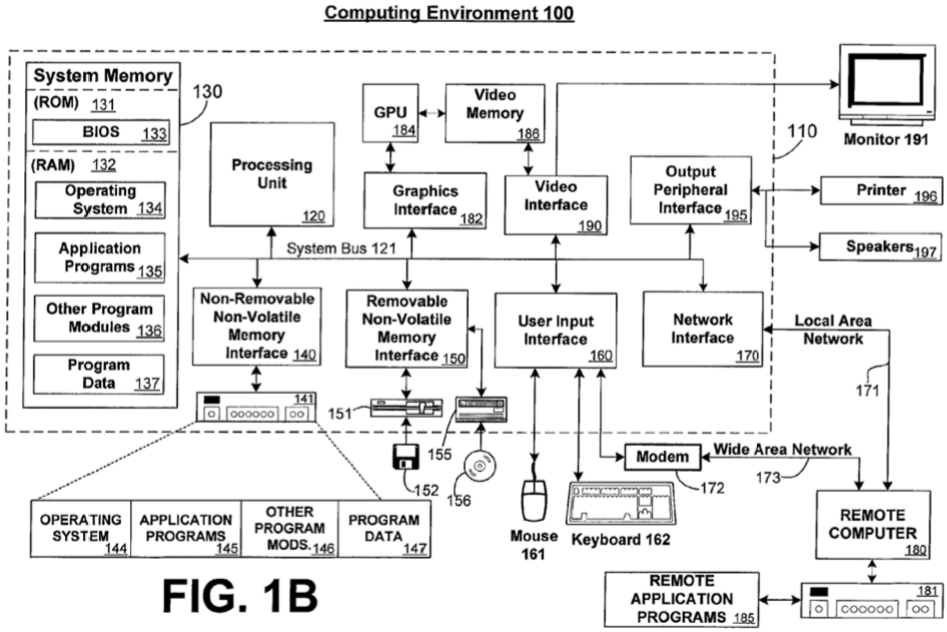



FIG. 1B

Compare Ex1004, Fig. 1 with Ex1005, Fig. 1B.

154. In my opinion, a POSITA reviewing Wilt would have understood there to be multiple advantages of "careful management" of co-processor computational

resources. Ex1005, [0007]. Such careful management would be particularly important in computationally intensive applications, such as simulating an artificial neural network as described by Buck. Wilt explains that, "[a]s graphics chips become more functional, it is reasonable to expect the number of active applications that demand significant computational resources from them to increase, and for the granularity of switching between these applications to become finer." *Id.*, [0008]. By providing "various ways to manage the scheduling and dispatch of graphics processing requests," Wilt addresses "the need for efficient and effective management of computational resources of graphics processor(s) in a computing system." *Id.*, [0009]. And Buck's computing system is specifically designed to perform the "more sophisticated computations" that Wilt contemplates a GPU performing. In my opinion, a POSITA would have looked to the art for solutions to efficiently manage the complex computing tasks delegated to Buck's GPU. Ex1004, [0010]–[0012], Ex1005, [0005].

155. Additionally, Wilt's task scheduling and resource management system allows the system to maintain data coherency and manage data dependencies between the GPU and CPU. By efficiently allocating and coordinating tasks between the CPU and GPU, schedulers ensure that data remains consistent and up-to-date across all processing units. This prevents conflicts and ensures that each task has access to the most recent and accurate data. Schedulers also manage

interdependencies by facilitating proper synchronization and communication between the CPU and GPU.

156. In my opinion, a POSITA would have been motivated to implement Wilt's GPU resource management system and method in Buck's GPU-accelerated parallel processing system and would have understood that Wilt's resource and task management system and method would efficiently manage the complex non-graphics computing tasks delegated to Buck's GPU. Ex1005, [0005]. Furthermore, Buck and Wilt have complementary disclosures: whereas Buck focuses on the interfacing of the hardware and applications of its GPU-accelerated system, Wilt provides details regarding operating system requirements and behind-the-scenes task and resource management. In my opinion, a POSITA would have understood that, when combined, the disclosures of Buck and Wilt provide a fulsome picture of the optimized GPU-accelerated system proposed.

157. Furthermore, in my opinion, a POSITA would have had a reasonable expectation of success when implementing Wilt's teachings in Buck's system. Wilt's system efficiently distributes a GPU's computational resources and accommodates various synchronization requirements in co-processor system that with significant computational demands on the co-processor—like Buck's system. Buck's optimized GPU-accelerated system is a co-processor system with complex synchronization and resource management requirements as contemplated by Wilt.

Indeed, in my opinion, a POSITA would have understood that Wilt's teachings would be applicable to Buck's GPU-accelerated system and would have provided benefits such as optimal resource allocation.

158. Therefore, it is my opinion that a POSITA would have understood that the combination of Buck and Wilt would have been an implementation of a known technique (e.g., Wilt's resource and task management) within a known system (e.g., Buck's optimized GPU-accelerated system) to yield predictable results (e.g., fast and efficient computation of complex numerical simulations).

**2. Element-by-Element Analysis**

159. I begin by analyzing independent claim 21 and its dependents, including claims 38–41. I then proceed to address claims 33–37, which recite similar concepts to claim 21 with the addition of trivial features and steps, all of which are disclosed in the prior art.

**a) Claim 21**

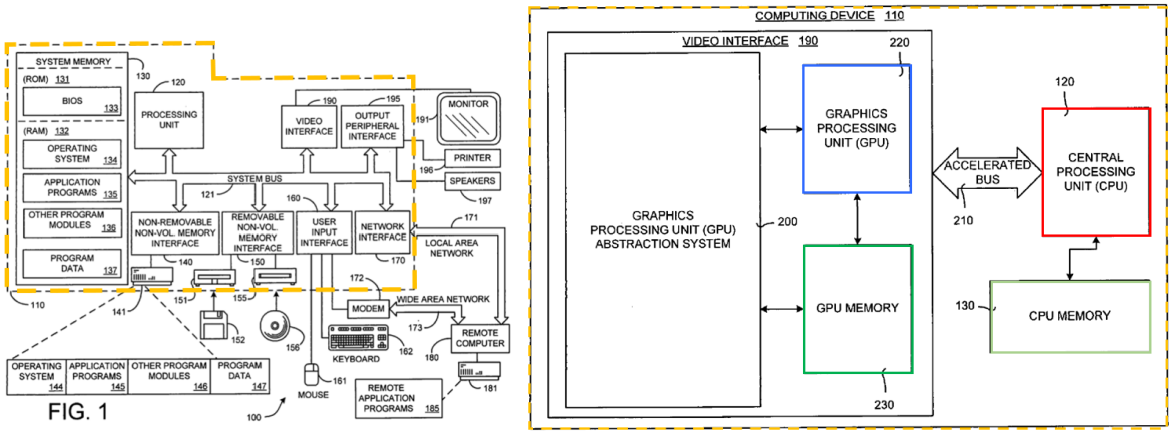
**i. 21[pre]: "A method of executing computations representing an artificial neural network on a computer system comprising at least one central processing unit (CPU), a processing unit, a first memory partition, and a second memory partition, the method comprising:"**

160. In my opinion, to the extent the preamble is limiting, the combination of Buck in view of Wilt discloses or teaches it.

161. Buck discloses "systems and methods" for "processing and execution of general matrix operations." Ex1004, Abstract. These operations can be used for

machine learning applications, such as "neural network[s]." *Id.*, [0029]–[0031], [0055], [0076]–[0083], [0129]–[0132], [0157]–[0158], [0210], [0211]–[0228], [0229]–[0232], [0233]–[0245], [0246]–[0250], Figs. 14–16, cls. 22, 26, 28. Buck walks through working examples of "a training session" and "a testing session" of a "pair of layers in a neural network." *Id.*, [0029]–[0031], [0229]–[0250]. Thus, Buck discloses a method of executing computations representing an artificial neural network as claimed.

162. As I have illustrated below, Buck's computer system (computing environment 100) includes a computer 110 (annotated in orange) having a central processing unit (CPU 120, annotated in red), a processing unit (GPU 220, annotated in blue), a CPU memory 130 (light green), and a GPU memory 230 (annotated in green). *Id.*, [0037]–[0052], [0053]–[0057].




Ex1004, Figs. 1, 2 (annotated).

163. Buck's CPU memory teaches at least a "first memory partition." Buck's GPU memory is segmented and additionally teaches at least a "first memory

partition" and a "second memory partition." Buck's GPU memory includes a workspace and "read only buffers (segments) outside [the] workspace [that] may be used as input data." *Id.*, [0123], Fig. 9. The workspace includes scratch space having a "temporary buffer" and "a permanent buffer" and contains "all operands as well as a destination for computation." *Id.*, [0070], [0124]. GPU memory 230 stores various types of data, including shader "programs P," "initial weights," "learning parameters," "input data," "target data," "intermediate results," and "trainable parameters." *Id.*, [0122], [0235]–[0236], Figs. 9, 15, 16. For example, as illustrated below in Fig. 15, the CPU performs the steps (annotated in red) of loading "GPU programs (P), learning parameter[s] (L), initial trainable parameter[s] (W)," training input data (X′) and target training data (T) into GPU memory. The segmented GPU memory stores (annotated in green) the initial trainable parameter[s] (W), and training input data (X′), and target training data (T), in addition to the shader programs (P) (not shown) and maintaining a temporary workspace (also annotated in green).

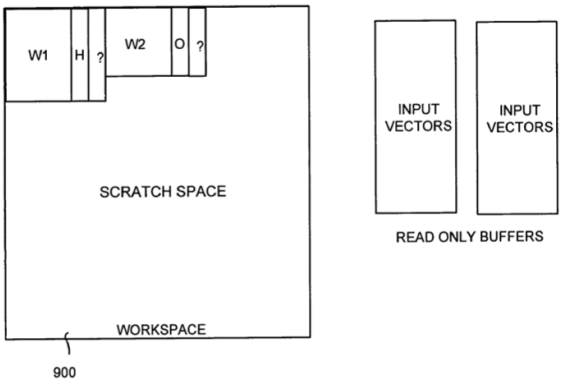



FIG. 9

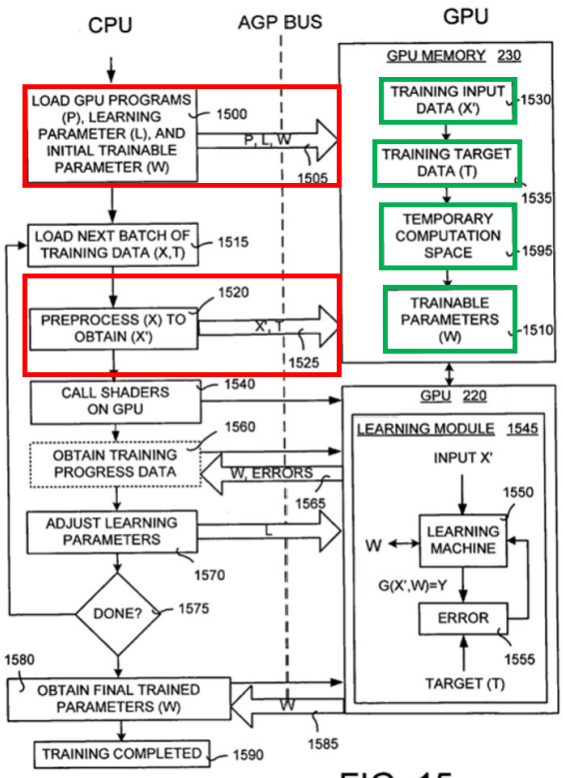



FIG. 15

Ex1004, Figs. 9, 15.

164. Buck's description of its segmented GPU memory teaches different partitions of memory, and thus teaches at least a "first memory partition" and a "second memory partition." For example, texture data can be stored in a first segment or a "first memory partition" and shader programs can be stored in a second segment or a "second memory partition." The '461 patent explains that "partitioning" can "be done logically," and that different portions of a single memory bank can be used to store different types of data, such as "input data, output data, intermediate results, and parameters." Ex1001, 5:65–6:11. Therefore, Buck, alone or in view of Wilt, teaches a method of executing computations representing an

artificial neural network on a computer system comprising at least one central processing unit (CPU), a processing unit, a first memory partition, and a second memory partition.

**ii. 21[a]: "executing, by the at least one CPU, a user interaction stream, the user interaction stream controlling transfer of inputs to the artificial neural network to the first memory partition and the second memory partition;"**

165. In my opinion, the combination of Buck in view of Wilt discloses or renders Element 21[a] obvious.

166. A user "may enter commands and information into" Buck's computer 110 "through input devices such as a keyboard 162, and pointing device 161, commonly referred to as a mouse, trackball[,] or touchpad." Ex1004, [0049]. These input devices are connected to Buck's CPU (processing unit 120) "through a user input interface 160 that is coupled to the system bus 121" or "by other interface and bus structures," including "a parallel port, game port[,] or a universal serial bus (USB)." *Id.*, [0049]–[0050]. Other input devices, including a microphone, satellite dish, or a television or broadcast video receiver, "or the like" can be connected to the CPU through the user input interface 160 as well. *Id.*

167. Buck discloses that its CPU processes these user interactions and commands by executing a user interaction stream. Processing unit 120 (CPU) maintains its own "stream[s]" of instructions and data. Ex1004, [0008]. As the "host" processor Buck's CPU receives and interprets the "commands and

information" from user input interface 160. *Id.* The CPU then translates these commands and information into instructions and data to be transmitted to GPU 220. Ex1004, [0008].

168. The CPU stores these commands and information in its CPU memory 130 (a first memory partition). Ex1004, [0046]. CPU memory 130 includes RAM 132, which "typically contains data and/or program modules that are immediately accessible to and/or presently being operated on by" CPU 120. *Id.*

169. As I have explained above, GUIs for interacting with models and simulations were well-known and widely used prior to the '461 patent. Section IV.B.2. Such user interfaces allowed users to control and interact with applications at a high level through simple visual representations and elements that perform more complex actions and analysis without requiring the user to understand the implementation of those actions and analysis.

170. Buck's system is designed to be used by a user for simulating artificial neural networks, and requires input from a user to set-up and run networks. Of course, the inputs to Buck's CPU would include instructions defining, setting, or editing functions (such as activation, learning, and training functions of the artificial neural network), parameters, or variables of an artificial neural network. The information would also include, for example, the memory locations from which data should be loaded, user-defined arrays of data, or live streams of audio, video, or text

data. Thus, these "commands and information" provided through Buck's user input interface would include inputs to the artificial neural network that specify how the artificial neural network should process information, and what information to process.

171. In a GPU-accelerated system, such as Buck's, the CPU processes data received from user interactions (or through input devices connected to the user input interface) for use in operations performed by the GPU and prepares that data as graphics data (e.g. a texture map). Ex1004, [0010]–[0012], [0055], [0058]–[0068], [0069]–[0210], [0211]–[0250]; Figs. 9, 15, 16.

172. The CPU then transfers this texture data to its segmented GPU memory, storing input data in a texture in one segment (a first memory partition) and other input data in a texture in another segment (a second memory partition). *Id.* For example, one memory partition may contain "initial weights" and/or "learning parameters," while a second memory partition may contain other input data such as "trainable parameters." Ex1004, [0106]–[0120], [0235]–[0250].

173. The CPU also processes instructions received from user interactions (or through input devices connected to the user input interface) for execution by the GPU and prepares those instructions as fragment shaders (e.g., shader programs). Ex1004, [0069]–[0075]; *see also id.* at [0010]–[0012], [0055], [0058]–[0068], [0069]–[0210], [0211]–[0250]; Figs. 9, 15, 16. The CPU then transfers the shader

programs to its segmented GPU memory in a yet another segment for shader programs, which can also be the claimed "second memory partition.

174. Thus, Buck's discussion of its CPU processing the commands and information, which results in inputs to the artificial neural network being transferred to segments of GPU memory discloses that the user interaction stream controls transfer of inputs to the artificial neural network to the first memory partition and the second memory partition.

175. Thus, in my opinion, the combination of Buck in view of Wilt discloses or renders obvious that the CPU executes a user interaction stream that controls transfer of inputs to the artificial neural network to the first memory partition and the second memory partition because Buck's CPU's processing of commands and information causes inputs to the artificial neural network to be transferred to the CPU memory and/or segments of GPU memory.

**iii. 21[b]: "executing, by the processing unit, a computational stream, the computational stream controlling data exchange between the user interaction stream and the computational stream during execution of the computations representing the artificial neural network;"**

176. In my opinion, the combination of Buck in view of Wilt discloses or renders Element 21[b] obvious.

177. Buck's "GPU optimization system and method" ports "some or all of the CPU processing," particularly "general matrix operations," which are "well-

suited for processing on the GPU," to the GPU. Ex1004, [0035]. These operations include computations to train and test a two-layer neural network. *Id.*, [0211]–[0250]. Buck describes its GPU (a "processing unit") as an autonomous processor with its "own instruction stream[], datapaths, and dedicated memory." *Id.*, [0008]. Thus, Buck discloses a processing unit executing a computational stream including computations representing the artificial neural network. *Id.*

178. As I have explained above, the stream programming model laid the groundwork for the development of much of modern GPU architecture and programming environments prior to the '461 patent. Section IV.B.2. Indeed, it was well-known at the time to use separate streams for different types of data in the stream programming model. This approach enabled parallel processing of different data streams for optimized performance and resource allocation. Buck's system "alleviates the computational limitations of CPUs by porting some or all of the CPU processing to the GPU." Ex1004, [0035]. By executing the easily parallelized computations over large sets of data on the GPU, Buck's system offloads not only the graphics pipeline from the CPU (the traditional purpose of a special-purpose GPU) but also the computations representing the artificial neural network. Thus, in the combined Buck and Wilt system, while the CPU executes a user interaction stream by processing user commands and information, the GPU executes a

computational stream by executing computations, including those representing an artificial neural network.

179. Buck discloses that the computational stream controls the data exchange between the user interaction stream and the computational stream. Just as in the ’461 patent, Buck’s computational stream “interacts with the User Interaction Stream…through initialization routines and data exchange in between simulations.” Ex1001, 7:60–9:52. “After the user starts the simulation,” although Buck’s CPU can continue to receive and process information and commands for the GPU while the GPU executes computations representing an artificial neural network, the GPU does not receive any information or commands “until the simulation is finished or interrupted” and thus Buck’s computational stream controls data exchange in the way described in the ’461 patent. Ex1001, 8:14–9:52; Ex1004, [0008], [0055], [0106]–[0120], [0235], [0211]–[0250]. In the ’461 patent, the “user can perform operations” to direct the CPU to parse new input textures for the GPU or convert computational elements to GPU programs “as many times as necessary prior to starting the simulation,” but once the simulation has been initialized, the user’s commands “to interrupt the simulation, change the input, or to change the display properties…are queued to be performed at times determined by the controller-driven exchange.” Ex1001, 8:14–9:52.

180. Similarly, once Buck's GPU begins executing a stream of instructions for the computations representing the artificial neural network, Buck's CPU (or a particular thread of the CPU) "blocks" and remains in a state of suspension until the GPU completes executing the simulation. Ex1004, [0235], [0211]–[0250]. Buck's blocked CPU does not provide instructions or data to the GPU, perform synchronization with the GPU, or otherwise exchange data with the GPU. *Id.* Therefore, the CPU is prevented from exchanging data with the GPU. The CPU (or CPU thread) remains blocked until the GPU completes executing a set of instructions. *Id.*

181. In other words, when Buck's GPU begins computations, the computational stream controls data exchange between the user interaction stream executed by the CPU and the computational stream executed by the GPU while the computations are executed by causing suspension of Buck's CPU (or CPU thread). Buck thus discloses that the computational stream controls the data exchange between the user interaction stream and the computational stream because no data exchange can take place between the user interaction stream and the computational stream while the computational stream is being executed. *Id.*

182. Wilt also discloses a GPU that executes a stream of hardware-specific commands. Ex1005, [0028], [0046]. Wilt explains that its GPU is "controlled by sending command buffers from the CPU." *Id.*, [0095]. When Wilt's CPU receives a

command from an application to perform a graphics operation, that command is usually received as an application programming interface (API) call. That call is typically at a high level of abstraction, meaning that it is more abstract and closer to natural human language, and can be translated into a lower level "token stream," which is less abstract and closer to machine language. *Id.*, [0046]. A "token stream," or "a stream of hardware-independent tokens," describes a series of operations and can be further translated by a hardware-specific software component into a "command buffer." A "command buffer" is "a stream of hardware-specific" commands that are "ready for consumption by graphics hardware." *Id.*, [0028], [0046].

183. The CPU controls the GPU by sending these "command buffers" for the GPU to execute. Ex1005, [0095]. Thus, Wilt further discloses that the GPU is executing streams of hardware-specific commands that together constitute a "computational stream" that performs the requested computations. As I have explained above with respect to Element 21[pre], in the combination of Buck and Wilt, these computations represent an artificial neural network. Ex1004, [0055], [0230], [0233]–[0250].

184. Furthermore, Wilt discloses that a "graphics processor can only execute one command buffer at a time," and that, once running, "the command buffer remains in the running state…until some event preempts execution of the command

buffer or the entire contents of the command buffer have been read and executed." Ex1005, [0086], [0108]. Therefore, Wilt teaches that while the GPU is executing a command buffer (a portion of the computational stream), the GPU does not execute any other command buffer. In other words, Wilt teaches that while the GPU is executing the computational stream, the GPU cannot change the instructions it executes or the data on which it operates. Like the "computational stream" in the '461 patent, Wilt's command buffer controls whether data exchange takes place between the user interaction stream executed by the CPU and the computational stream executed by the GPU because it prevents data exchange while the computation is being executed. Ex1001, 8:14–9:52. In the combined Buck-Wilt system, the instruction stream executed by the GPU, which includes the currently executing command buffer, controls the exchange of data between any stream and the instruction stream, because no data exchange can take place between any stream and the computational stream while the computational stream is being executed.

185. Wilt additionally teaches that its computational stream controls whether data exchange takes places through its discussion of preemption. Ex1005, [0108], [0127]. Preemption is the reclaiming of a resource from a process before the process has finished using it. When a currently executing task is preempted, it is temporarily interrupted so that another task, typically with a higher priority, can be executed. *Id.* The priorities of various command buffers, which are portions of the computational

stream, are attributes of the computational stream. *Id.* Based on these attributes of the computational stream, Wilt's GPU may halt execution of the preempted command buffer to execute the preempting command buffer and change the instructions it executes or the data on which it operates. *Id.* In this additional way, Wilt teaches that the computational stream controls whether data exchange takes place between the user interaction stream executed by the CPU and the computational stream executed by the GPU because it allows data exchange while the computation is being executed if the computation is preempted.

186. Thus, in my opinion, the combination of Buck in view of Wilt discloses or renders obvious a processing unit executing a computational stream that controls data exchange between the user interaction stream and the computational stream during execution of the computations representing the artificial neural network.

**iv. 21[c]: "shifting control of a data exchange between the user interaction stream and the computational stream to the computational stream in response to starting execution of the computations representing the artificial neural network;"**

187. In my opinion, the combination of Buck in view of Wilt discloses or renders Element 21[c] obvious.

188. As I have explained above with respect to Element 21[a], the combined Buck-Wilt system discloses that its CPU executes a user interaction stream that can

provide instructions and information to its GPU. Ex1004, [0008], [0049]–[0050], [0069]–[0075].

189. As I have explained above with respect to Element 21[b], in the combined Buck-Wilt optimized GPU-accelerated system, once the GPU receives an instruction and begins performing the computation (i.e., executing the computations representing the artificial neural network), it continues performing that computation until it is complete or preempted. Ex1004, [0235]; *see also id.*, [0035], [0211]–[0250]; Ex1005, [0086], [0108]. During this time, the GPU cannot execute any other instructions. Additionally, the CPU (or a particular thread of the CPU) blocks until the GPU completes the computation or the computation is preempted. Ex1004, [0235]; *see also id.*, [0035], [0211]–[0250]; Ex1005, [0086], [0108]. Thus, the combination of Buck and Wilt discloses that its GPU's execution of computations (i.e., simulating an artificial neural network) prevents the CPU from exchanging data with the GPU until it the computations are complete or preempted, and therefore that the computational stream has control over data exchange between the user interaction stream executing on the CPU and the computational stream executing on the GPU while the GPU is executing the computations. Ex1004, [0035], [0233]–[0250]; Ex1005, [0086], [0108].

190. This suspension of the data exchange between the user interaction stream and the computational stream once the GPU begins executing computations

teaches that control of the data exchange shifts between the user interaction stream and the computational stream to the computational stream in response to starting execution of the computations representing the artificial neural network as claimed.

**v. 21[d]: "shifting control of the data exchange between the user interaction stream and the computational stream to the user interaction stream in response to completion or interruption of the computations representing the artificial neural network;"**

191. In my opinion, the combination of Buck in view of Wilt discloses or renders Element 21[d] obvious.

192. As I have explained above with respect to Elements 21[b] and 21[c], in the combined Buck and Wilt system, the computational stream has control over data exchange between the user interaction stream executing on the CPU and the computational stream executing on the GPU while the GPU is executing the computations. Ex1004, [0035], [0211]–[0250]; Ex1005, [0086], [0108]. The GPU continues executing a computation until it is complete or preempted. Ex1004, [0235]; *see also id.*, [0035], [0233]–[0250]; Ex1005, [0086], [0108]. Preemption is the reclaiming of a resource from a process before the process has finished using it. When a currently running task is preempted, it is temporarily interrupted so that another task that requires a resource being used by the currently running task, and typically has a higher priority, can be executed instead. *Id.* Thus, Buck in view of

Wilt discloses the completion or interruption of computations representing an artificial neural network.

193. Once a command buffer has finished running or is preempted, Wilt's system "release[s] all of the resources that were needed for the command buffer and unblocks pending command buffers that need those resources." Ex1005, [0115], [0151]. For example, segments of GPU memory storing internal variables, functions, and other parameters of the artificial neural network can be updated once they are released. Buck also discloses that the CPU (or a particular thread of the CPU) blocks until the GPU completes the computation, and therefore that the CPU (or particular CPU thread) is no longer blocked once the GPU completes the computation. Ex1004, [0235]. When Buck's CPU (or particular CPU thread) unblocks, it can resume providing instructions or data to the GPU, performing synchronization with the GPU, or otherwise exchanging data with the GPU. Therefore, the combined Buck-Wilt system discloses shifting control of the data exchange between the computational stream (the instruction stream executed by the GPU) and the user interaction stream (the instruction stream related to the user interactions provided by the CPU through the user input interface) back to the user interaction stream in response to completion or interruption of the computations. This resumption of the data exchange between the user interaction stream and the computational stream once the GPU's execution of the computations is complete or preempted teaches that

control of the data exchange shifts between the user interaction stream and the computational stream back to the user interaction stream in response to starting execution of the computations representing the artificial neural network.

194. Additionally, Wilt explains that the scheduler accounts for variables such as the "priority" of the command buffer itself and of a requesting thread when scheduling tasks. Ex1005, [0107]. Scheduling algorithms including priority scheduling and round robin scheduling were well-known at the time of the '461 patent, and it would have been obvious that different threads, such as threads representing a computation or threads requiring data transfer, could have different priority values. Section IV.B.7. It would have been obvious to a POSITA, in my opinion, for example, that a new instruction to display the results of a computation received through the user input interface could be assigned a lower priority value than the execution of a computation for the neural networks, while a new instruction to halt the computation received through the user input interface could be assigned a higher priority value than the execution of the computation.

195. Thus, in my opinion, the combination of Buck in view of Wilt discloses or renders obvious shifting control of the data exchange between the user interaction stream and the computational stream to the user interaction stream in response to completion or interruption of the computations representing the artificial neural network.

### vi. 21[e]: "queueing a user command received by the user interaction stream during execution of the computations representing the artificial neural network; and"

196. In my opinion, the combination of Buck in view of Wilt discloses and teaches Element 21[e].

197. As I have explained above with respect to Element 21[a], Buck discloses that user commands received through its user input interface coupled to the CPU are received by the user interaction stream executed by the CPU. Ex1004, [0049]–[0050].

198. When a command for Wilt's GPU is received, that command is written into a command buffer. Once the command buffer is full or the command has been completely written into the command buffer, the command buffer enters a waiting state until various waiting criteria are fulfilled. Ex1005, [0106]–[0107].

199. The command buffer "scheduler," a component of Wilt's operating system, then "decides which command buffer should be dispatched." Id., [0024], [0146]–[0154]. Wilt's scheduler minimizes latency between the completion of one command buffer and the submission of another command buffer to the GPU by "queu[ing]" command buffers for execution "before the current command buffer has finished executing." *Id.*, [0093], [0126].

200. In particular, Wilt's resource management system "maintain[s] multiple DMA buffers and 'ping-pong[s]' between them, writing to one while the

graphics processor reads and executes another." *Id.*, [0085]–[0086], [0093]. In this way, although the GPU can only execute one command buffer at a time, the CPU can continue to prepare command buffers for the GPU—based on commands received while the GPU is executing a different command buffer—to be executed by the GPU at a later time. *Id.*, [0086]. The command buffers in the "ready" state can be "queue[d] up" so that the GPU can then "immediately begin execution of another command buffer when the current command buffer has completed." *Id.*, [0126]. The combined Buck-Wilt system thus minimizes latency between the completion of one command buffer and the submission of another command buffer to the GPU by queuing command buffers for execution "before the current command buffer has finished executing." *Id.*, [0093], [0126].

201. Thus, in my opinion, the combination of Buck in view of Wilt discloses or renders obvious queueing a user command received by the user interaction stream during execution of the computations representing the artificial neural network. Ex1004, [0069]–[0075]; *see also id.* at [0059], [0061], [0067], [0069]–[0083], [0084]–[0087], [0235]–[0250], Figs. 9, 15, 16; Ex1005, [0086].

#### vii. 21[f]: "executing the user command during execution of the computations representing the artificial neural network at times determined by the computational stream."

202. In my opinion, the combination of Buck in view of Wilt discloses and teaches Element 21[f].

203. As I have explained above with respect to Elements 21[pre], 21[b], and 21[e], the combined Buck and Wilt system discloses performing computations representing an artificial neural network and scheduling commands for the GPU to execute based on a computational stream. Section VII.A.2.a.i, VII.A.2.a.iii, VII.A.2.a.vi; Ex1004, [0035], [0211]–[0250]; Ex1005, [0086], [0108].

204. Wilt's scheduler "has a great deal of flexibility in deciding which command buffer should be submitted to the hardware." Ex1005, [0107]. The scheduler considers attributes of the computational stream, such as whether a command buffer (part of the computational stream executed by the GPU) is currently executing, its "priority, if any," the priority of other command buffers, and the "priority of the requesting thread" when deciding whether and when to schedule the command buffer generated from the user command. *Id.*, [0107]. The scheduler then submits the command buffer for execution at a particular point within the computational stream. *Id.*, [0107].

205. Thus, in the Buck-Wilt system, a command buffer generated from a user command received while the GPU is executing computations would be submitted by the scheduler for execution at a time determined by the computational stream because the scheduler's decision is based on the status and priority of the other command buffers. Furthermore, because Buck-Wilt teaches that the command buffer can be scheduled for execution during the stream of computations that

represent the artificial neural network, the combination discloses that the user command is executed during execution of the computations.

206. To the extent that each command buffer that is currently being executed by the GPU, is interpreted as a computational stream, and not the set of all command buffers, Wilt also explains that "a variety of events could potentially cause preemption of a command buffer during execution." Ex1005, [0127]. In my opinion, it would have been obvious that certain commands, such as halting the simulation or changing the input data for the simulation, can be assigned a higher priority value than the currently executing computation, and therefore preempt the currently executing computation. Because the combined Buck-Wilt system supports preemption, if an event preempts the execution of the currently executing computation, the computation can be temporarily suspended and a new command buffer can be submitted to the GPU. Ex1005, [0108]–[0110], [0127], [0152]–[0153]. Once the preempting command buffer has been executed, the execution of the preempted command buffer can "be resumed where the preemption occurred." *Id.*, [0110] [0152]–[0153].

207. In this additional, alternative way, the Buck and Wilt combination teaches that the user command is executed during execution of the computations representing the artificial neural network at a time determined by the computational

stream because the currently executing command buffer must either be preempted or complete before a new command buffer is executed.

208. It is therefore my opinion that the combination of Buck in view of Wilt discloses or renders obvious executing the user command during execution of the computations representing the artificial neural network at times determined by the computational stream. Therefore, Buck in view of Wilt renders obvious the entirety of Claim 21.

**b) Claim 22: "The method of claim 21, wherein the user interaction stream controls the data exchange between the user interaction stream and the computational stream outside of execution of the computations representing the artificial neural network."**

209. In my opinion, the combination of Buck in view of Wilt discloses or renders obvious Claim 22.

210. As I have explained above with respect to Claim 21, the combined Buck-Wilt system's GPU is controlled by command buffers sent by the CPU and the Buck/Wilt GPU's computational stream and the CPU's user interaction stream can exchange data. Ex1005, [0095]–[0143]; Section VII.A.2.a. Once a command buffer or instruction has finished running or is preempted, the Buck-Wilt system "release[s] all of the resources that were needed for the command buffer and unblocks pending command buffers that need those resources." Ex1005, [0115], [0151].

211. Although the Buck-Wilt GPU's computational stream controls the data exchange while the GPU is executing computations representing the artificial neural network, when the GPU of the combined Buck-Wilt system is not executing any command buffers, it is idle and can receive different command buffers specifying different instructions and data on which it should operate. Ex1004, [0211]–[0250]; Ex1005, [0086], [0108].

212. As I have explained above with respect to Element 21[d], upon completion or interruption of the GPU's execution of computations, control over the data exchange between the user interaction stream and the computational stream shifts to the user interaction stream. For example, resources such as GPU memory segments can be synchronized, edited, or otherwise changed. The CPU can also resume providing instructions or data to the GPU, performing synchronization with the GPU, or otherwise exchanging data with the GPU. Ex1004, [0211]–[0250]; Ex1005, [0086], [0108]. Therefore, the Buck-Wilt system discloses that the user interaction stream controls the data exchange between the computational stream (the instruction stream executed by the GPU) and the user interaction stream (the instruction stream related to the user interactions provided by the CPU through the user input interface) when the GPU is not performing computations.

213. Thus, it is my opinion that the combination of Buck in view of Wilt discloses or renders obvious that the user interaction stream controls the data

exchange between the user interaction stream and the computational stream outside of execution of the computations representing the artificial neural network.

**c) Claim 23: "The method of claim 21, wherein executing the user interaction stream comprises: controlling setting and editing of computational elements of the computations representing the artificial neural network."**

214. In my opinion, the combination of Buck in view of Wilt teaches or renders Claim 23 obvious.

215. Buck and Wilt each disclose a user input interface through which a "user may enter commands and information into the computer." Ex1004, [0049]–[0050]; Ex1005, [0071]. The "commands and information" that a user provides to Buck-Wilt's computer allow a user to program the "programmable pixel shade[r]s" to "implement the various primitive operations" used in the artificial neural networks, define variables, implemented as textures, and to define neural networks to train or simulate using these shaders and textures. Ex1004, [0011]–[0012]. The '461 patent explains that "computational elements" refers to the system of equations defining the artificial neural network and includes "textures that track variables plus shaders that define processing algorithms." Ex1001, 5:42–47, 8:55–58..

216. At a minimum, in my opinion, it would have been obvious to a POSITA that a user can define and edit the shaders and textures used to "implement the various primitive operations used in" the computations representing an artificial neural network. Ex1004, [0011]. As the '461 patent acknowledges, editing an

artificial neural network "is a user-interactive process." Ex1001, 8:63–65. And, as I have explained above with respect to Element 21[a], it would have been obvious that Buck's user input interface allows a user to control and interact with Buck's GPU. Section IV.B.2; Section VII.A.2.a.ii.

217. Simple user interfaces for programming, such as command line interfaces and GUIs were well-known and in-use for decades at the time of the '461 patent. Buck's discussion of shaders "programmable" for a variety of applications, including running simulations, discloses that a user could program the shaders or update variables, stored as textures, as appropriate for their use case. Ex1004, [0007], [0011], [0055]. Because there was a finite number of known, predictable user interface options, including a GUI or command line interface, it is my opinion that a POSITA would have found it obvious to implement an interface through which a user could create or edit the shaders and textures, and would have had a reasonable expectation of success in doing so. Section IV.B.3; Ex1016; Ex1017.

218. Thus, because there was a finite number of known, predictable user interface options, including a GUI or command line interface, in my opinion, a POSITA would have found it obvious to implement an interface through which a user could create or edit the shaders and textures, and would have had a reasonable expectation of success in doing so.

**d) Claim 24: "The method of claim 21, wherein executing the user interaction stream comprises: controlling setting and editing of**

**parameters of the computations representing the artificial neural network."**

219. In my opinion, the combination of Buck in view of Wilt discloses or renders obvious Claim 24.

220. As I have explained above with respect to Claim 23, the combined Buck-Wilt system allows a user to input commands and information through a user input interface. Ex1004, [0049]–[0050]; Ex1005, [0071]. Those inputs would include setting or editing the parameters of the artificial neural network algorithms when using Buck's system to process an artificial neural network. For example, Buck discusses loading "initial weights" and "other learning parameters" during training and testing. Ex1004, [0235]–[0249]. Thus, Buck discloses that a user could enter or import parameters of the computations to set up and run the networks for training or testing. In my opinion, it would have been obvious that a user can define and edit the parameters of the computations representing an artificial neural network as appropriate for their use case.

221. Thus, it is my opinion that the combination of Buck in view of Wilt teaches executing user commands controlling setting and editing of parameters of the computations representing the artificial neural network and discloses or renders obvious this limitation.

**e) Claim 25: "The method of claim 21, wherein executing the user interaction stream comprises: controlling setting and editing of parameters of the inputs to the artificial neural network."**

222. In my opinion, the combination of Buck in view of Wilt teaches or renders obvious Claim 25.

223. As I have explained above with respect to Claims 23 and 24, the combined Buck-Wilt system allows a user to provide commands and information to its computer to set and edit parameters of the computations representing the artificial neural network through a user input interface. Ex1004, [0049]–[0050]; Ex1005, [0071]. That information would include the sources of input data to the artificial neural network, as well as "initial weights, and other learning parameters." Ex1004, [0236]. For example, Buck discusses loading training data and query data during training and testing, along with parameters used for the inputs. Ex1004, [0235]–[0249]. Thus, Buck discloses that a user could set or edit parameters of the inputs to the artificial neural network, such as the source of inputs, weights, and learning parameters. Additionally, in my opinion, it would have been obvious that a user can define and edit the parameters of the inputs to an artificial neural network, because artificial neural networks operate on changing inputs to yield different outputs.

224. Thus, it is my opinion that the combination of Buck in view of Wilt teaches executing user commands controlling setting and editing of parameters of

the inputs to the artificial neural network and discloses or renders obvious this limitation.

**f) Claim 26: "The method of claim 21, wherein executing the user interaction stream comprises: specifying an output to be saved to disk and/or displayed on a screen."**

225. In my opinion, the combination of Buck in view of Wilt discloses or renders obvious Claim 26.

226. Buck discloses that the output of a computation performed by its GPU-accelerated system "is displayed on the display device 191," (annotated in pink). Ex1004, [0060]. Buck's system also stores the output of a computation using a number of storage media, such as a "hard disk drive 141 that reads from or writes to non-removable, nonvolatile magnetic media," a "magnetic disk drive 151 that reads from or writes to a removable nonvolatile magnetic disk 152," and an "optical disk drive 155 that reads from or writes to a removable, nonvolatile optical disk 156" (annotated in dark green). *Id.*, [0046]–[0047].

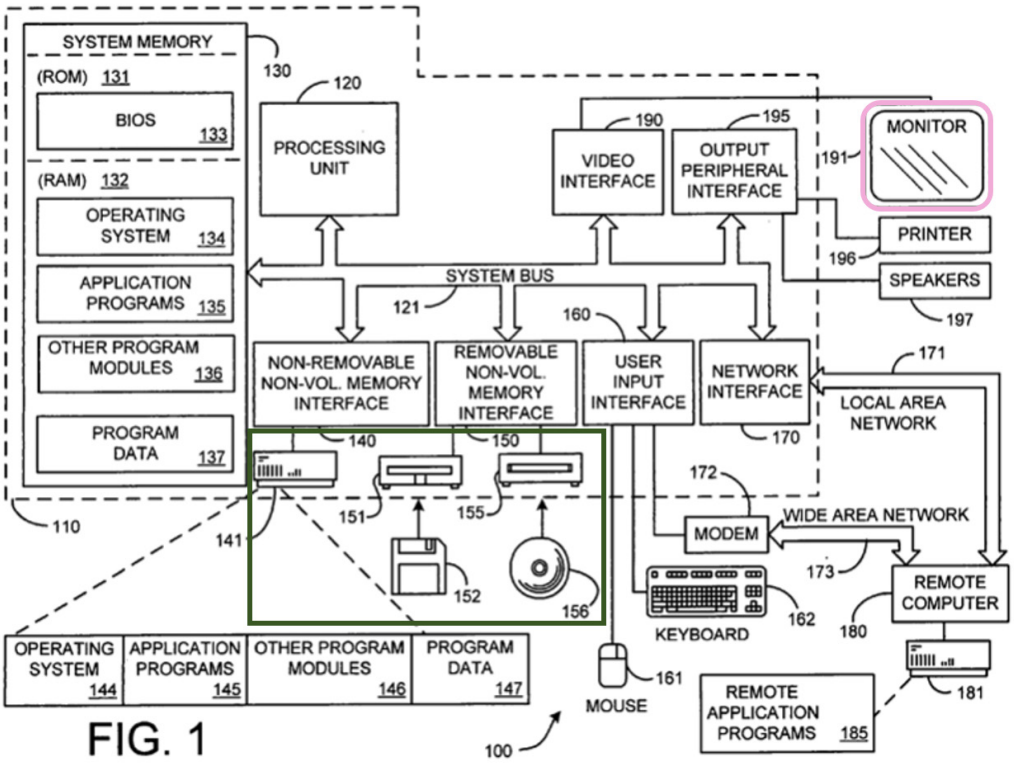



FIG. 1

Ex1004, Fig. 1 (annotated) reference characters 141, 151, 152, 155, 156, 191.

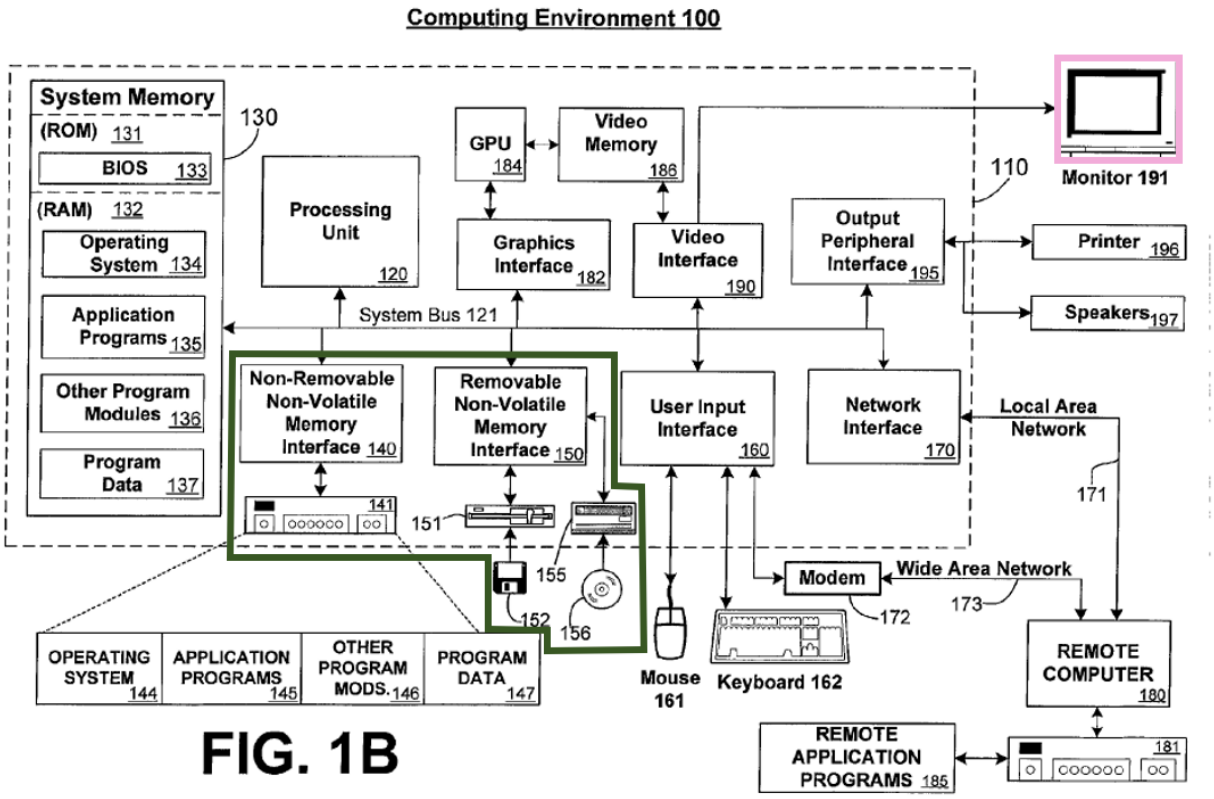



FIG. 1B

Ex1005, Fig. 1B (annotated) reference characters 141, 151, 152, 155, 156, 191.

227. As I have explained above with respect to Claims 23 and 24, a user can provide to commands and information to the combined Buck-Wilt system's computer through a user input interface. Ex1004, [0049]–[0050]; Ex1005, [0071]. Those commands would include specifying an output to be saved to disk and/or displayed on a screen to view or save the results of training or testing the artificial neural networks. Thus, Buck in view of Wilt discloses that a user could specify an output to be saved to disk and/or displayed on a screen. *Id.* At a minimum, it is my opinion that it would have been obvious that a user can choose to save to disk or display the output of a set of computations representing an artificial neural network.

*Id.* A user of a system simulating an artificial neural network, such as the Buck-Wilt system, would want to be able to review the output of the simulation to use, modify, or perform further analysis on the results.

228. Thus, in my opinion, the combination of Buck in view of Wilt teaches executing user commands specifying an output to be saved to disk and/or displayed on a screen.

**g) Claim 27: "The method of claim 21, wherein executing the user interaction stream comprises: parsing elements to be used in the computations representing the artificial neural network."**

229. In my opinion, the combination of Buck in view of Wilt discloses or renders obvious Claim 27.

230. Buck explains that "[f]unctions operating on the inputs are expressed as sequences of instructions," and that a "complete program is broken into successive passes of these functions over the data," where each pass represents "a single data parallel operation in the program." Ex1004, [0068], [0099]–[0101], [0107], Fig. 8. Complex functions may "be divided up into further passes" due to architecture limitations such as instruction length, storage availability, and number of outputs permitted. *Id.* at [0068]. Buck's discussion of breaking an application or function into a series of passes teaches parsing elements to be used in the computations representing the artificial neural network. Ex1004, [0069]–[0075], *see also id.*, [0076]–[0210], Fig. 8.

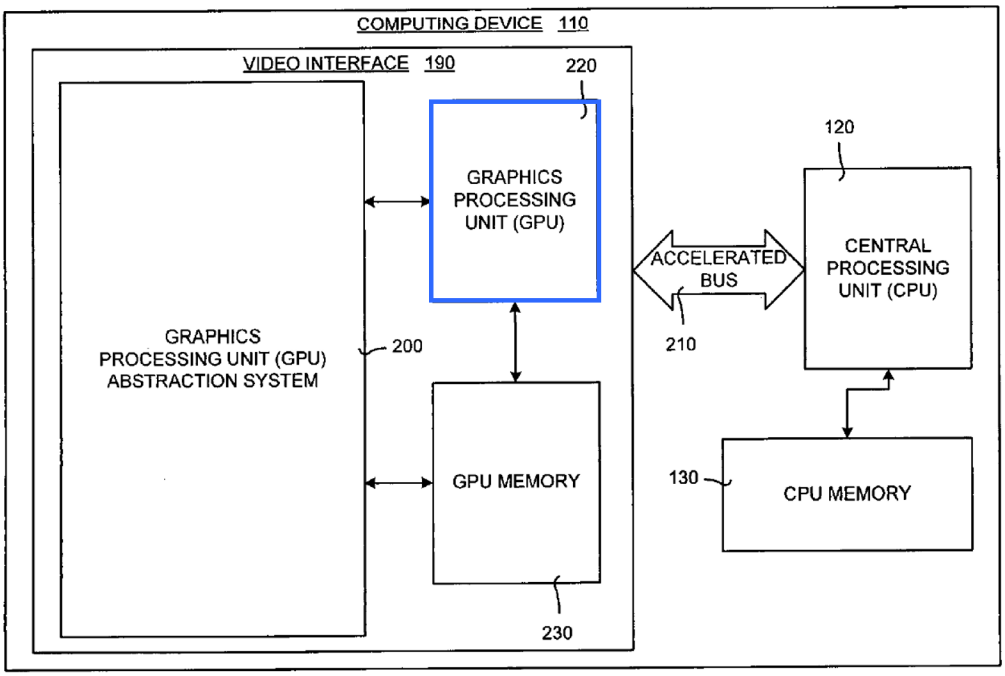
231. And, as I have explained above with respect to Claims 23 and 24, the combined Buck-Wilt system allows a user to input commands and information, such as defining, setting, or editing of computational elements through a user input interface. Ex1004, [0049]–[0050]; Ex1005, [0071]. Those inputs would include commands parsing elements to be used in the artificial neural network computations.

232. Thus, it is my opinion that the combination of Buck in view of Wilt teaches that executing the user interaction stream includes parsing elements to be used in the computations representing the artificial neural network and discloses or renders obvious this limitation.

**h) Claim 28: "The method of claim 27, wherein the processing unit comprises a graphics processing unit (GPU) and executing the user interaction stream further comprises: converting the elements into GPU programs."**

233. In my opinion, the combination of Buck in view of Wilt discloses or renders obvious Claim 28.

234. As I have explained above with respect to Element 21[b] and illustrated below, the combined Buck-Wilt system's processing unit is a GPU (GPU 220, annotated in blue).




Ex1004, Fig. 1 (annotated).

235. Additionally, Buck describes how fragment, or shader, processors that are running shader programs are used for general purpose computation on GPUs. Ex1004, [0069]–[0075]. Buck also describes how various operations, including those for implementing the neural networks it describes, can be "implemented using one or more pixel shaders." *Id.* at [0076]–[0087].

236. As I have explained above with respect to Claim 27, the combined Buck-Wilt system parses machine learning algorithms into "successive passes," each representing "a single data parallel operation in the program." Ex1004, [0076]–[0087]. These passes are implemented with "simple primitives (or building blocks)" that are each in turn implemented "using one or more pixel shaders." *Id.* Each pixel

shader is a program expressed as a "series of instructions." *Id.* Thus, the Buck-Wilt system teaches converting the elements into GPU programs.

237. As I have explained above with respect to Claims 23 and 24, the combined Buck-Wilt system allows a user to provide commands and information to its computer to define, set, and edit the computations representing the artificial neural network through a user input interface. Ex1004, [0049]–[0050]; Ex1005, [0071]. Those inputs would include commands to convert the successive passes (elements) into GPU programs. It was well-known before the '461 patent that GPGPU applications were implemented on GPUs through programmable shaders, and thus it have been obvious that a user instruction to implement a GPGPU algorithm would result in converting elements of a computation in the GPGPU algorithm into GPU programs. Section IV.A.

238. Thus, it is my opinion that the combination of Buck in view of Wilt teaches that executing the user interaction stream includes converting parsed computational elements into GPU program and discloses or renders obvious this limitation.

**i) Claim 29: "The method of claim 28, wherein executing the user interaction stream comprises: compiling the GPU programs."**

239. In my opinion, the combination of Buck in view of Wilt discloses or renders obvious Claim 29.

240. Buck explains that "[a]ctual execution" of algorithms "in the GPU requires that the list of passes be created in a preprocess," and that the "individual functions must be compiled and downloaded to the GPU." Ex1004, [0228]. And as I have discussed above with respect to claims 23 and 24, the combined Buck-Wilt system discloses a user interface through which a user can enter commands and information into the computer to define, set, and edit the computations representing the artificial neural network through a user input interface. Ex1004, [0049]–[0050]; Ex1005, [0071]. Those commands can include instructions to execute a function, which would initiate compilation of the GPU programs.

241. Thus, it is my opinion that the combination of Buck-Wilt teaches that executing the user interaction stream includes compiling the GPU programs and discloses or renders obvious this limitation.

**j) Claim 30: "The method of claim 29, wherein executing the user interaction stream comprises: transferring the GPU programs to the second memory partition."**

242. It is my opinion that the combination of Buck in view of Wilt discloses or renders obvious Claim 30.

243. Buck describes loading the "programs P" for the different shaders into GPU memory as part of the training and testing process for an artificial neural network. Ex1004, [0236], [0246]; *see also* Figs. 15, 16. As I have discussed above, the various segments of GPU memory 230 disclose multiple memory partitions. *Id.*,

[0070], [0122]–[0124], [0235]–[0236], Figs. 9, 15, 16. Thus, Buck discloses loading the shader programs P to a particular segment of GPU memory, which is a transfer of the program into the second memory partition.

244. As I have discussed above with respect to Claims 23 and 24, the combined Buck-Wilt system discloses a user interface through which a user can enter commands and information into the computer. Ex1004, [0049]–[0050]; Ex1005, [0071]. Those commands would include commands to transfer the GPU programs to a segment of GPU memory (a second memory partition) so they can be executed. Ex1004, [0011]–[0012].

245. Thus, it is my opinion that the combination of Buck in view of Wilt teaches that executing the user interaction stream includes transferring the GPU programs to the second memory partition and discloses or renders obvious this limitation.

**k) Claim 31: "The method of claim 21, further comprising: executing, by the at least one CPU, a data output stream, the data output stream controlling transfer of outputs of the computations representing the artificial neural network to disk."**

246. It is my opinion that the combination of Buck in view of Wilt discloses or renders obvious Claim 31.

247. As I have explained above with respect to Element 21[a], the combined Buck-Wilt system's CPU, as the host processor, maintains its own "stream[s]" of

instructions and data. Ex1004, [0008]. And as I have explained above with respect to Claim 26, Buck-Wilt's CPU is connected to a number of storage media (annotated below in dark green), such as hard drive disk, magnetic disk drive, and optical disk drive through various memory interfaces that are "[t]ypically connected to the system bus 121." *Id.*, [0043]–[0049]; Ex1005, [0068]–[0070]. These storage media "can be used to store the desired information and . . . can be accessed by the computer 110." Ex1004, [0044].

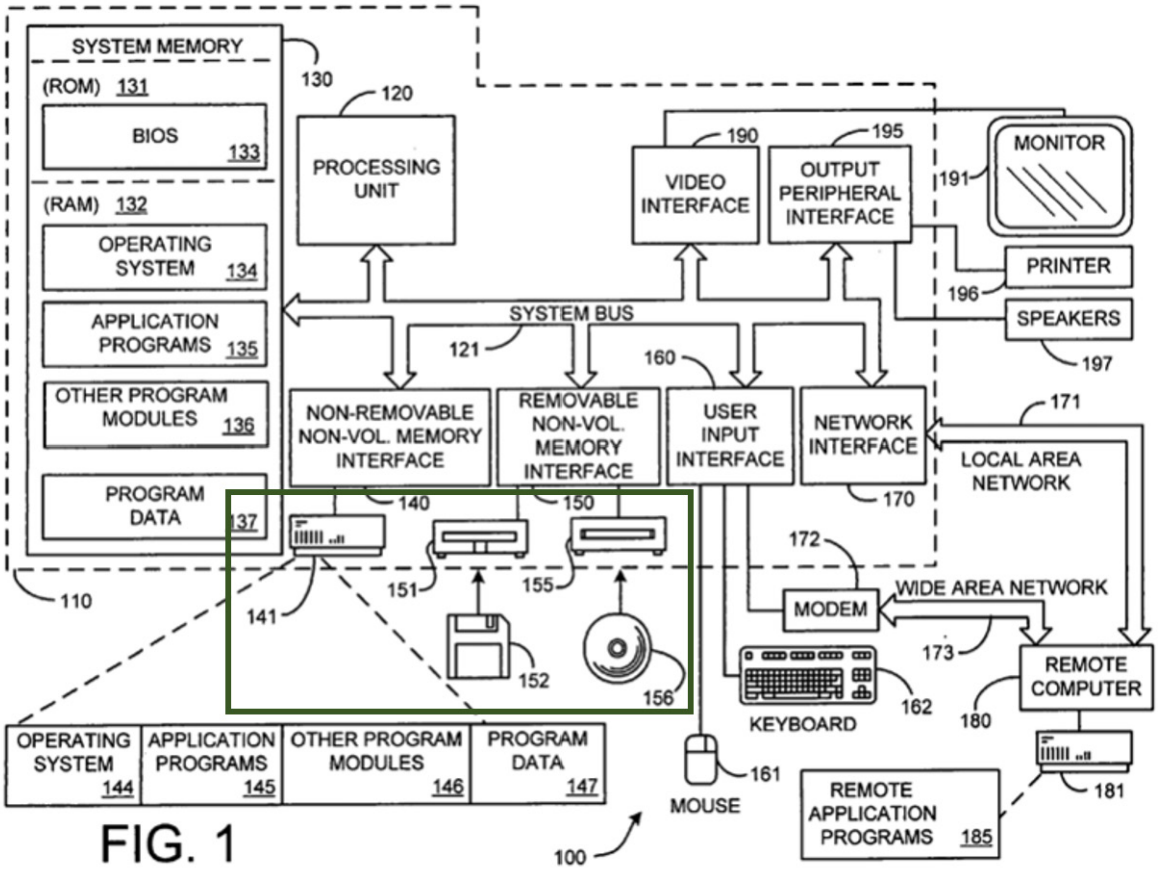


FIG. 1

Ex1004, Fig. 1 (annotated).

248. As I have explained above with respect to Claim 26, a user can provide commands and information to the combined Buck-Wilt system's computer through a user input interface, including commands specifying an output to be saved to disk to save the results of training or testing the artificial neural networks. Ex1004, [0049]–[0050]; Ex1005, [0071]. These user commands would be executed by Buck's CPU (i.e., as part of the user interaction stream executed by the CPU). Ex1004, [0049]–[0050]. In my opinion, it would have been obvious that, as the host processor, the CPU manages the flow of data within the system and transfers an output of the computations representing the artificial neural network being performed by its GPU to the hard drive disk. Ex1004, [0043]–[0049]; Ex1005, [0068]–[0070].

249. Thus, it is my opinion that the combination of Buck in view of Wilt teaches that the CPU can execute a data output stream that controls transfer of outputs of the computations representing the artificial neural network to disk.

**l) Claim 32: "The method of claim 21, further comprising: generating the inputs with a video camera during execution of the computations."**

250. It is my opinion that the combination of Buck in view of Wilt discloses or teaches this claim.

251. The combined Buck-Wilt system includes a number of input devices that are "not shown," such as "a television or broadcast video receiver" that are

capable of providing streaming video data. Ex1004, [0049]–[0050]; Ex1005, [0071]. These devices are "connected to the processing unit 120 through a user input interface 160 that is coupled to the system bus 121" or through "other interface and bus structures." Ex1004, [0049]–[0050]; Ex1005, [0071].

252. Furthermore, the system uses "predominantly DMA-based input," which is commonly used with "peripheral" devices, "such as a digital camera." Ex1005, [0085], [0093]. Wilt explains that this memory management method locks "the memory region where a digital camera, for example, is to write the image data from a picture it is about to take," specifying that memory location to the camera, and allowing the camera to write directly to that location. *Id.*, [0085]. Wilt also explains that its GPU supports "real-time" processing. *Id.*, [0107].

253. In my opinion, it would have been obvious to a POSITA that, in the Buck-Wilt combination, the CPU can receive a stream of input data from an input device such as a video camera. Like the "video receiver" in Buck-Wilt, a video camera provides video input. Thus, in my opinion, POSITA would have understood that Buck-Wilt's optimized system that accelerates the computation of artificial neural networks by shifting computations to the GPU can be used to process a stream of video data input captured by a video camera and input to the Buck-Wilt system.

254. In my opinion, it would have been obvious that Buck-Wilt's optimized, real-time system that accelerates the computation of artificial neural networks by

shifting computations to the GPU can be used to process a stream of video data input generated by a video camera.

255. Furthermore, the Buck-Wilt resource management system "facilitate[s] the host microprocessor's performance of useful work while the coprocessor hardware is running." Ex1005, [0114]. Therefore, the Buck-Wilt CPU can continue to receive and store input data gathered by the video camera in real-time and transmitted to the CPU while the GPU is executing computations. This video camera data is used as input to the Buck-Wilt artificial neural network. *Id.* Thus, in the Buck-Wilt combination, the CPU receives and processes the stream of video data input and transfers the resulting texture data to the GPU during the computations representing the artificial neural networks. Ex1007, 5.

256. Thus, in my opinion, the combination of Buck in view of Wilt teaches generating the inputs with a video camera during execution of the computations representing an artificial neural network.

**m) Claim 38: "The method of claim 21, wherein the user command causes interruption of the computations representing the artificial neural network."**

257. It is my opinion that the combination of Buck-Wilt discloses or renders obvious Claim 38.

258. As I have discussed above with respect to Element 21[d], the Buck-Wilt system for implementing artificial neural networks supports preemption.

Ex1004, [0055], [0211]–[0250]; Ex1005, [0127]. Certain commands, such as halting the simulation or changing the input data for the simulation, can be assigned a higher priority value than the currently executing computation, and therefore preempt—or interrupt—the currently executing artificial neural network computation. Ex1005, [0108]–[0110], [0127], [0152]–[0153].

259. As I have discussed above with respect to Claims 23 and 24, the Buck-Wilt system discloses a user interface through which a user can enter commands and information into the computer to define, set, and edit the computations representing the artificial neural network through a user input interface. Ex1004, [0049]–[0050]; Ex1005, [0071]. Those inputs would include commands that interrupt the artificial neural network computation.

260. Thus, it is my opinion that the combination of Buck in view of Wilt teaches executing user commands that cause interruption of the computations representing the artificial neural network and discloses or renders obvious this limitation.

**n) Claim 39: "The method of claim 21, wherein the user command causes a change in the inputs to the artificial neural network."**

261. In my opinion, the combination of Buck in view of Wilt discloses or renders obvious Claim 39.

262. As I have explained above with respect to Claims 23 through 25, a user can provide user interactions, or user commands, through the user input interface the Buck-Wilt system to define, set, and edit inputs to the artificial neural network, including weights, biases, shaders, and textures. Ex1004, [0011]–[0012], [0049]–[0050], [0235]–[0249]; Ex1005, [0071]. Furthermore, in my opinion, it would have been obvious to a POSITA that a user can cause a change in the inputs to an artificial neural network, because artificial neural networks operate on changing inputs to yield different outputs.

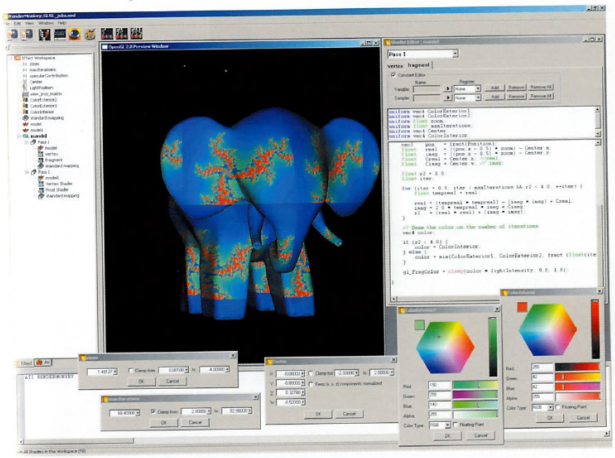
263. Thus, it is my opinion that the combination of Buck in view of Wilt teaches user commands that cause a change in the inputs to the artificial neural network and discloses or renders obvious this limitation.

**o) Claim 40: "The method of claim 21, wherein the user command causes a change in display properties of an output of the computations representing the artificial neural network."**

264. In my opinion, the combination of Buck in view of Wilt discloses or renders obvious Claim 40.

265. As I have explained above with respect to Claims 23, 24, and 26, the combined Buck-Wilt system allows a user to provide commands and information to its computer to specify an output of the artificial neural network to be displayed through a user input interface. Ex1004, [0049]–[0050], [0060], Fig. 1; Ex1005, [0071]. These commands can include changing display properties of the output to

be displayed. For example, Buck discloses that the output of a computation performed by its GPU-accelerated system "is displayed on the display device 191." Ex1004, [0060]. In my opinion, a POSITA would have found it obvious that the user command could cause a change in display properties of an output of the computations. The ability to change display properties, such as color, size, etc. of an output of a simulation was standard for user interfaces at the time of the '461 patent. Section IV.B. For example, GUIs that allowed a user to interactively adjust parameters used by the GPU for shading an object were widely used with graphics APIs such as OpenGL—an API that the '461 patent acknowledges is prior art.



**olor Plate 1** Screen shot of the RenderMonkey IDE user interface. The shader that is procedurally nerating a Julia set on the surface of the elephant is shown in the source code window. Color selection ols and user interface elements for manipulating user-defined uniform variables are also shown.

Ex1016, 27.

266. Thus, it is my opinion that the combination of Buck in view of Wilt teaches that executing a user command that causes a change in display properties of an output of the computations representing the artificial neural network.

**p) Claim 41: "The method of claim 21, wherein, during execution of the computations representing the artificial neural network, the computational stream controls the data exchange between the user interaction stream and the computational stream by requesting the inputs to the artificial neural network."**

267. In my opinion, the combination of Buck in view of Wilt discloses or renders obvious Claim 41.

268. As I have explained above with respect to Claim 21, by preventing data exchange when a computation is being executed, the command buffers (part of the computational stream) control the data exchange between the user interaction stream executed by the CPU and the computational stream executed by the GPU.

Just as in the '461 patent, in the Buck-Wilt system, when a computation (i.e., from a command buffer) is executing, new shaders and textures are made available to the GPU through the command buffer "not when ready but upon the request of the controller." Ex1001, 9:10–52. Wilt's GPU can notify the CPU when certain events occur, such as "the completion, or partial completion, of a command buffer, completion of a DMA operation to write data into memory visible to the host processor . . . or completion of a specific command." Ex1005, [0088], [0094]. Once Wilt's GPU has completed execution of a command buffer or the command buffer has been preempted, the GPU can notify the CPU that it is no longer executing that command buffer, signaling that it can now execute another command buffer and change the instructions it executes or the data on which it operates. *Id.*

269. In my opinion, it would have been obvious to a POSITA that, if the GPU has completed execution of the command buffer and new input data is required, the GPU's notification could result in a user being prompted for input data. Because the completion or interruption of a command buffer (a portion of the computational stream) causes the GPU to provide a notification to the CPU, which can, in turn, cause a user to be prompted for input data, the command buffer controls the data exchange between the user interaction stream and the computational stream by requesting the inputs. If the required resources included inputs to the artificial neural network, the command buffer would cause requesting the inputs from the user and transferring them from the CPU to the GPU memory.

270. Thus, in my opinion, the combination of Buck in view of Wilt teaches that the computational stream controls the data exchange between the user interaction stream and the computational stream by requesting the inputs to the artificial neural network.

**q) Claim 33:**

**i. 33[pre]: "A system for executing computations representing an artificial neural network, the system comprising:"**

271. In my opinion, the combination of Buck in view of Wilt discloses or teaches this limitation.

272. This limitation was addressed in the discussion of Element 21[pre], and it is disclosed or taught by Buck in view of Wilt for substantially the same reasons. Section VII.A.2.a.i. Buck discloses "systems and methods" for "processing and execution of general matrix operations" such as "neural network[s]." Ex1004, Abstract, [0029], [0077]–[0083], [0131], [0157], [0210], [0211]–[0228], [0229]–[0250].

**ii. 33[a]: "a camera to acquire input data for the artificial neural network;"**

273. In my opinion, the combination of Buck in view of Wilt discloses or teaches this limitation.

274. This limitation is substantially similar to Element 21[pre] and Claim 32 and is disclosed or taught by Buck in view of Wilt for substantially the same reasons. Section VII.A.2.a.i, VII.A.2.l. Element 33[a] differs from Claim 32 only in that it recites a "camera" and not a "video camera," and that the camera is part of the system for executing the computations representing an artificial neural network. The combined Buck-Wilt system includes a number of input devices "connected to the processing unit 120 through a user input interface 160 that is coupled to the system bus 121" or through "other interface and bus structures." Ex1004, [0049]–[0050]; Ex1005, [0071]. As I have explained above with respect to Claim 32, it is my opinion that a POSITA would have understood that the combined Buck and Wilt system could receive and process an input stream of video data generated by a video camera.

For those same reasons, it is my opinion that the Buck-Wilt combination teaches a video camera that acquires inputs to the artificial neural network.

275. Thus, it is my opinion that the Buck and Wilt combination discloses or renders obvious a camera to acquire input data for the artificial neural network that is part of the system for executing the computations representing an artificial neural network. Ex1004, [0049]–[0050]; Ex1005, [0071], [0085], [0093].

**iii. 33[b]: "a first memory partition;"**

276. In my opinion, the combination of Buck in view of Wilt discloses or teaches this limitation.

277. This limitation recites nothing more than is included in Element 21[pre] and is disclosed or taught by Buck in view of Wilt for substantially the same reasons. Section VII.A.2.a.i. Buck's segmented GPU memory includes at least a first memory partition. Ex1004, [0158], [0235]–[0236], Figs. 15, 16.

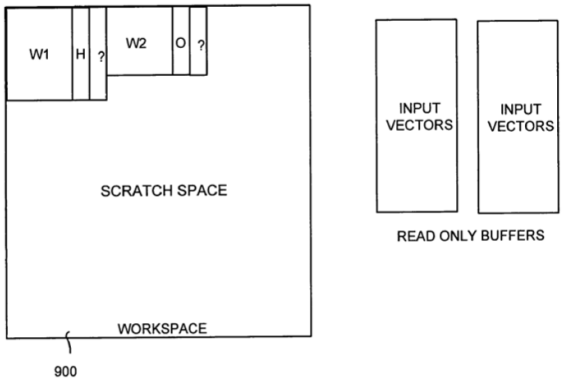



FIG. 9

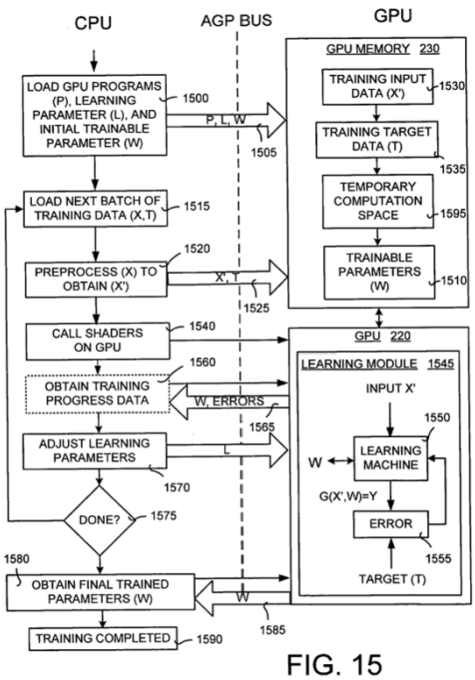



FIG. 15

Ex1004, Figs. 9, 15.

### iv. 33[c]: "a second memory partition;"

278. In my opinion, the combination of Buck in view of Wilt discloses or teaches this limitation.

279. This limitation recites nothing more than is included in Element 21[pre] and is disclosed or taught by Buck in view of Wilt for substantially the same reasons. Section VII.A.2.a.i. Buck's segmented GPU memory includes at least a second memory partition. Ex1004, [0158], [0235]–[0236], Figs. 15, 16.

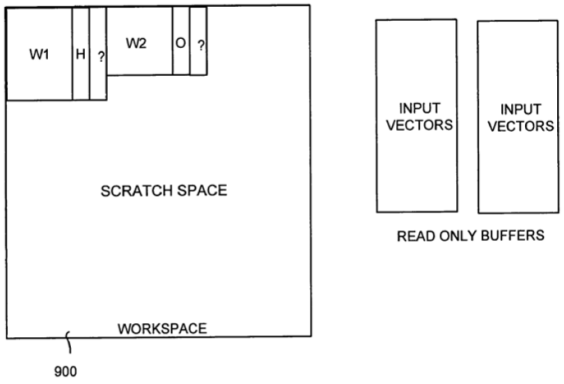



FIG. 9

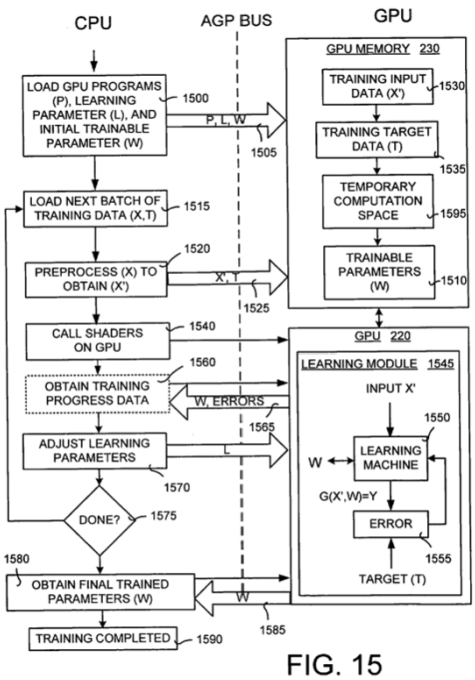



FIG. 15

Ex1004, Figs. 9, 15.

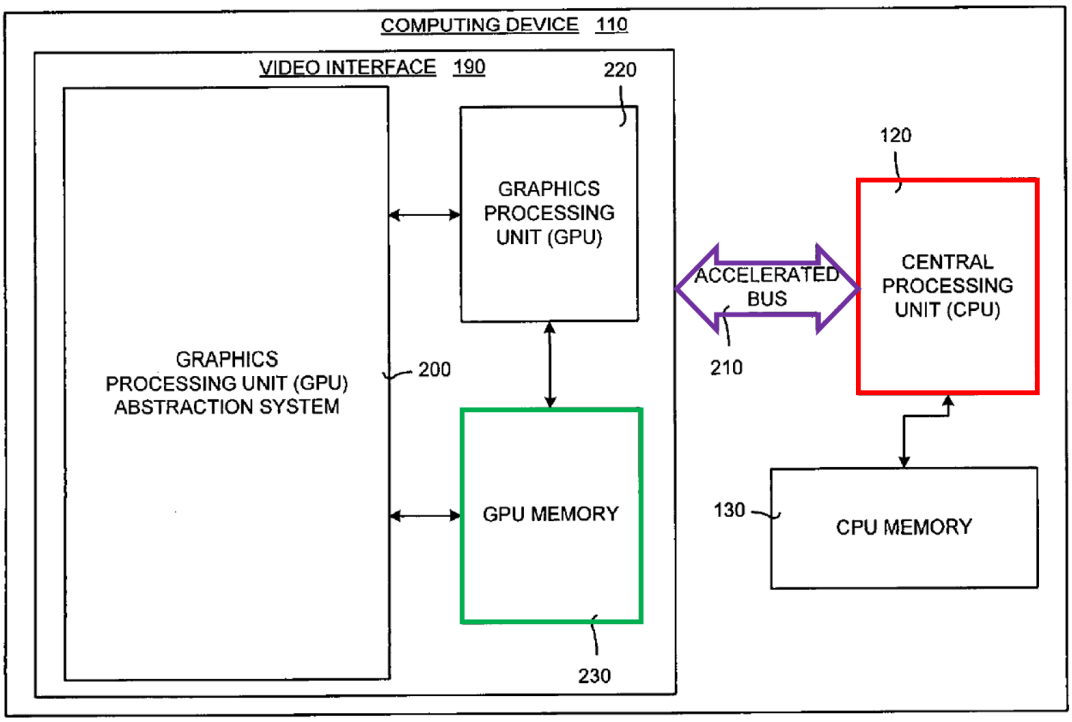
**v. 33[d]: "at least one central processing unit (CPU), operably coupled to the camera, the first memory partition, and the second memory partition, to execute a user interaction stream, the user interaction stream controlling transfer of the input data acquired by the camera to the first memory partition and the second memory partition during execution of the computations representing the artificial neural network;"**

280. In my opinion, the combination of Buck in view of Wilt discloses or teaches this limitation.

281. This limitation is substantially similar to Elements 21[pre] and 21[a] (a CPU that executes a user interaction stream controlling transfer of inputs to the artificial neural network to the first memory partition and the second memory

partition) and is disclosed or taught by Buck in view of Wilt for substantially the same reasons. Sections VII.A.2.a.i, VII.A.2.a.ii.

282. As I have illustrated below, Buck's central processing unit (CPU 120, annotated in red) is "in communication with" video interface 190, of which the first and second memory partitions (within GPU memory 230, annotated in green) are part, via accelerated bus 210 (annotated in purple). Ex1004, [0049]–[0050], [0055]–[0057]; *see also* Ex1005, [0067]–[0071].




Ex1004, Fig. 2 (annotated).

283. As I have explained above with respect to Claim 32, the Buck-Wilt system's CPU can receive real-time input data from an input device such as a video camera. A video camera providing input to the combined Buck-Wilt system would

be connected to the CPU through the system bus and a user input interface (or other interface and bus structures, such as "a parallel port, game port or a universal serial bus (USB)"). Ex1004, [0050]; *see also* Ex1005, [0071].

284. As I have explained above with respect to Element 21[e], the combined Buck-Wilt system's resource management system "facilitate[s] the host microprocessor's performance of useful work while the coprocessor hardware is running." Ex1005, [0114]. Therefore, the Buck-Wilt CPU can continue to receive and store input data generated by the video camera while the GPU is executing computations. Because this real-time video camera data is used as input to the combined Buck-Wilt system's artificial neural network, the CPU transfers the data to the GPU while the GPU continues to perform computations representing the artificial neural network. Ex1007, 5.

285. And, as I have explained above with respect to Claim 21, the combined Buck-Wilt system discloses processing commands and information, including real-time video data, received through the user input interface and transferring the resulting textures and shader programs to various segments of GPU memory. Ex1004, [0069]–[0075]; *see also id.* at [0012], [0055], [0060]–[0061], [0088]–[0091], [0235]–[0250], Figs. 9, 15, 16. Thus, the combination of Buck in view of Wilt discloses executing a user interaction stream controlling transfer of the input

data acquired by the camera to the first memory partition and the second memory partition.

286. Therefore, in my opinion, the combination of Buck in view of Wilt teaches that the CPU is operably coupled to the camera and the first and second memory partitions and that the CPU executes a user interaction stream controlling transfer of the input data acquired by the camera to the first memory partition and the second memory partition during execution of the computations representing the artificial neural network.

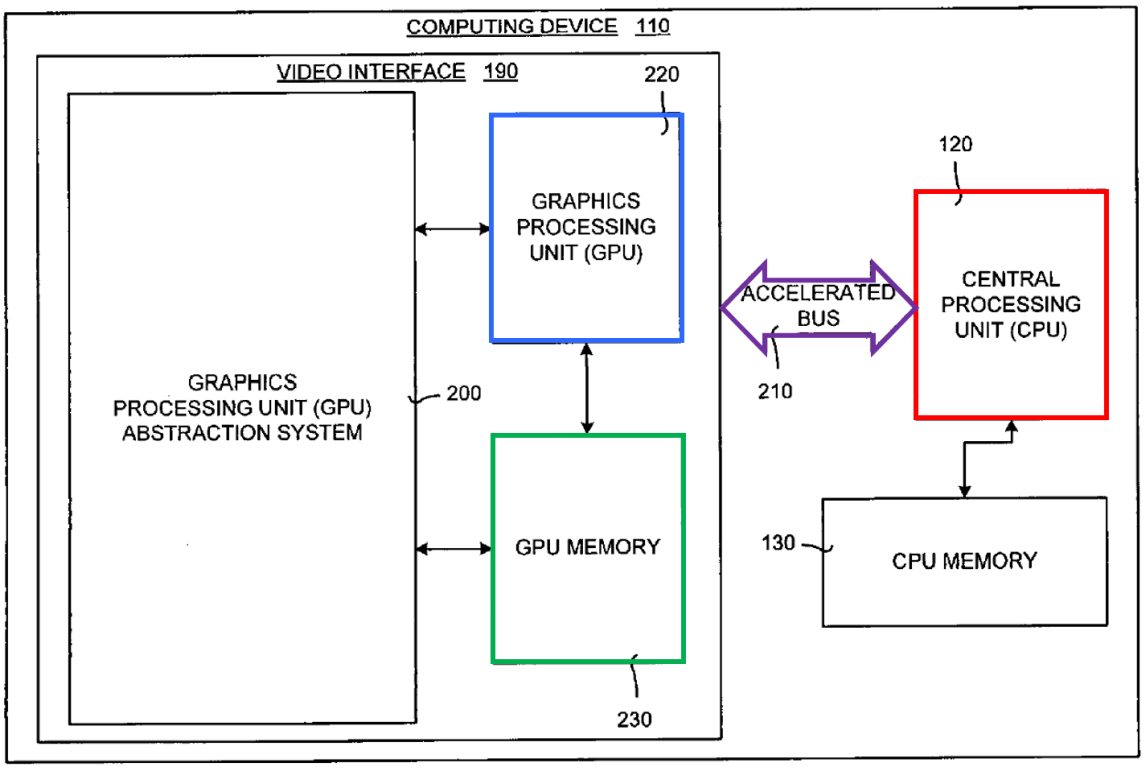
**vi. 33[e]: "a processing unit, operably coupled to the first memory partition, the second memory partition, and the at least one CPU, to execute a computational stream, the computational stream controlling transfer of the input data from the first memory partition and the second memory partition during execution of the computations representing the artificial neural network, the execution of the computations representing the artificial neural network occurring while the camera is acquiring the input data; and"**

287. In my opinion, the combination of Buck in view of Wilt discloses or teaches this limitation.

288. This limitation is substantially the same as Elements 21[pre], 21[b], and 21[c] and is disclosed or taught by Buck in view of Wilt for substantially the same reasons. Sections VII.A.2.a.i, VII.A.2.a.iii, VII.A.2.a.iv.

289. As I have illustrated below, Buck's processing unit (GPU 220, annotated in blue) is "in communication with" first and second memory partitions

(within GPU memory 230, annotated in green) and with CPU 120 (annotated in red) via accelerated bus 210 (annotated in purple). Ex1004, [0049]–[0050], [0055]–[0057]; *see also* Ex1005, [0067]–[0071].




Ex1004, Fig. 2 (annotated).

290. As I have explained above with respect to Claim 21, the combined Buck-Wilt system discloses that the computational stream controls the data exchange between the user interaction stream and the computational stream. Section VII.A.2.a.

291. Furthermore, as I have explained with respect to Claim 41, Wilt's command buffer controls the data exchange between the user interaction stream and the computational stream by requesting the inputs. Specifically, the completion or

interruption of a command from the command buffer causes the GPU to provide a notification to the CPU, which can then cause a user to be prompted for input data. Wilt's scheduler makes resources and data available according to the requirements of the command buffers (part of the computational stream). Ex1005, [0024]. The scheduler can then facilitate the transfer of this input data from the CPU (first memory partition) to a segment of the GPU memory (second memory partition). Thus, the combination of Buck and Wilt discloses that the computational stream controls transfer of the input data from the first memory partition and the second memory partition.

292. And, as I have explained above with respect to Element 33[d], the Buck-Wilt system's CPU can continue to receive and store input data acquired by the camera while the GPU is executing computations. Ex1005, [0114]; Ex1007, 5; Section VII.A.2.q.v. Thus, the combination of Buck in view of Wilt discloses that the computations representing the artificial neural network occur while the camera is acquiring the input data.

293. Therefore, in my opinion, a POSITA would have understood the combination of Buck in view of Wilt teaches that the processing unit is operably coupled to the first and second memory partitions and the CPU and that the processing unit executes a computational stream that controls transfer of the input data from the first memory partition and the second memory partition during

execution of the computations representing the artificial neural network that occur while the camera is acquiring the input data.

**vii. 33[f]: "a controller, operably coupled to the at least one CPU and the processing unit, to queue user interactions received by the user interaction stream during the execution of the computations representing the artificial neural network for performance at times selected to avoid data corruption."**

294. In my opinion, the combination of Buck in view of Wilt discloses or teaches this limitation.

295. This limitation is substantially similar to Elements 21[e] and 21[f] (queuing a user command received by the user interaction stream during execution of the computations representing the artificial neural network) and is disclosed or taught by Buck in view of Wilt for substantially the same reasons. Sections VII.A.2.a.vi, VII.A.2.a.vii. Element 33[f] differs from Elements 21[e] and 21[f] in that (a) it requires a controller, operably coupled to the CPU and the processing unit, to perform the queuing of the user interactions, and (b) the user interactions are executed at times selected to avoid data corruption.

296. The Buck-Wilt system uses a scheduler, a component of "the operating system of the host computing system," schedules and queues command buffers for the GPU from the CPU. Ex1005, [0024]. The Buck-Wilt scheduler is a controller because it schedules and controls the execution of tasks within the system. The '461

patent explicitly contemplates that the "controller can be partially or even fully implemented in software." Ex1001, 5:22-24.

297. Additionally, and in the alternative, it is my opinion that a POSITA would have understood that the Buck-Wilt system's scheduler could be implemented through a graphics interface, such as Wilt's graphics interface 182, as either software or hardware. As illustrated below, Wilt's graphics interface 182 (annotated in yellow) communicates with the processing unit (GPU 184, annotated in blue) and CPU (processing unit 120, annotated in red). Ex1005, [0071].

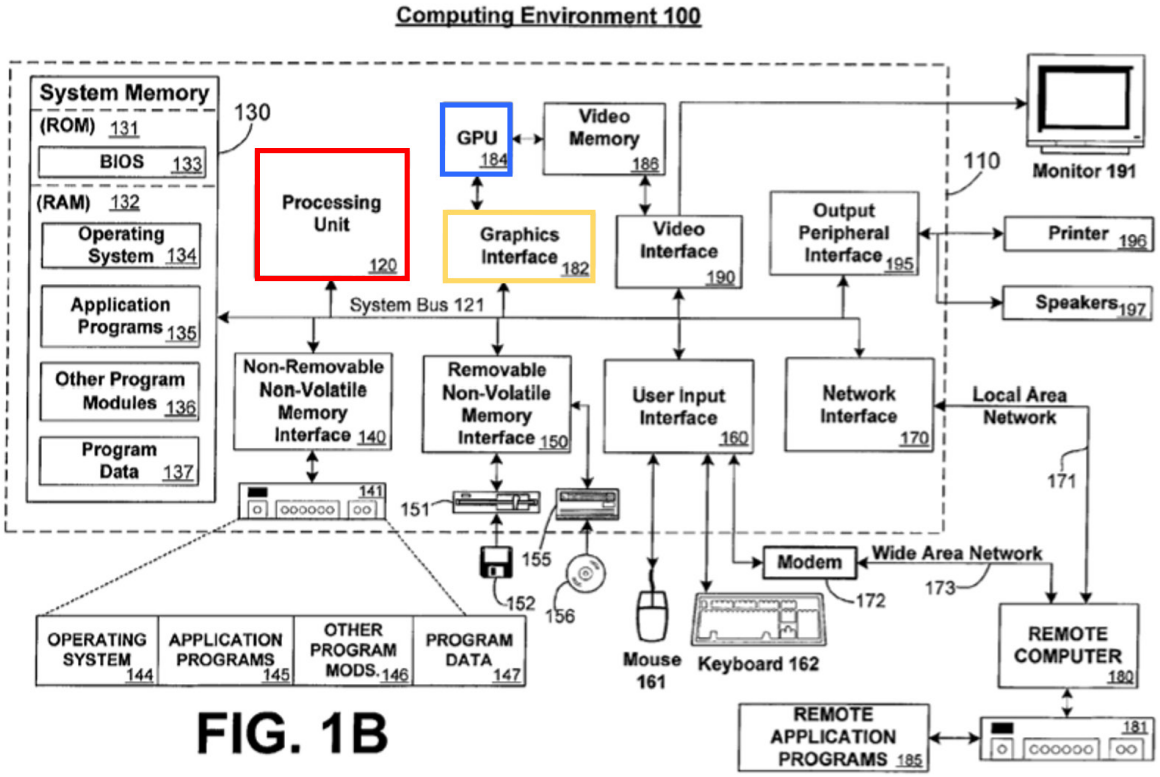



FIG. 1B

Ex1005, Fig. 1B (annotated).

298. The Buck-Wilt system's graphics interface "communicates with the CPU" over system bus 121 and "assumes responsibility" for Accelerated Graphics Port (AGP) communications that "transfer data from system memory to a graphics processor." Ex1005, [0087]. Therefore, the combined Buck-Wilt method of scheduling and queuing command buffers could be performed and communicated to the GPU through a controller such as graphics interface 182. For example, schedulers specifically for "scheduling the processing of a coprocessor" such as a GPU were known in the art prior to the '461 patent. *See* Ex1024.

299. Additionally, in my opinion, a POSITA would have understood that the scheduler queues tasks to be executed at times that would not cause data corruption. Although the '461 patent does not provide any examples of how it avoids data corruption, data corruption was well-known as an undesirable outcome that could cause the system to malfunction or behave erratically. Indeed, a POSITA designing a sophisticated system for managing coprocessor resources and implementing artificial neural networks would have designed the scheduler to queue tasks to be executed at times that "avoid data corruption." For example, for a real-time input, it was known that data corruption could occur if a user interaction is processed before computations on a frame are finished and written to memory. In my opinion, a POSITA would certainly have understood the basic scheduling principle of allowing necessary computations and commands to occur before allowing interruption.

300. Thus, in my opinion, the combination of Buck in view of Wilt discloses or renders obvious a controller operably coupled to the CPU and the processing unit that queues user interactions received by the user interaction stream during the execution of the computations representing the artificial neural network for performance at times selected to avoid data corruption. Therefore, Buck in view of Wilt renders obvious the entirety of Claim 33.

**r) Claim 34: "The system of claim 33, wherein the user interactions cause interruption of the computations representing the artificial neural network."**

301. In my opinion, the combination of Buck in view of Wilt discloses or teaches this claim.

302. This claim is substantially the same as Claim 38 and is disclosed or taught by Buck in view of Wilt for substantially the same reasons. Section VII.A.2.m. Claims 34 and 38 differ only in that Claim 34 recites "user interactions" and Claim 38 recites a "user command." The user commands that the combined Buck-Wilt system receives through the user input interface are user interactions. Thus, in my opinion, Buck in view of Wilt discloses or teaches the user interactions cause interruption of the computations representing the artificial neural network.

**s) Claim 35: "The system of claim 33, wherein the user interactions cause a change in inputs to the artificial neural network."**

303. In my opinion, the combination of Buck in view of Wilt discloses or teaches this claim.

304. This claim is substantially the same as Claim 39 and is disclosed or taught by Buck in view of Wilt for substantially the same reasons. Section VII.A.2.n. Claims 35 and 39 differ only in that Claim 35 recites "user interactions" and Claim 39 recites a "user command."

305. The user commands that the Buck-Wilt system receives through the user input interface are user interactions. Thus, in my opinion, Buck in view of Wilt discloses or teaches the user interactions cause a change in inputs to the artificial neural network.

**t) Claim 36: "The system of claim 33, wherein the user interactions cause a change in display properties of an output of the computations representing the artificial neural network."**

306. In my opinion, the combination of Buck in view of Wilt discloses or teaches this claim.

307. This claim is substantially the same as Claim 40 and is disclosed or taught by Buck in view of Wilt for substantially the same reasons. Section VII.A.2.o. Claims 36 and 40 differ only in that Claim 36 recites "user interactions" and Claim 40 recites a "user command." The user commands that the Buck-Wilt system

receives through the user input interface are user interactions. Thus, in my opinion, Buck in view of Wilt discloses or teaches the user interactions cause a change in display properties of an output of the computations representing the artificial neural network.

**u) Claim 37: "The system of claim 33, wherein the controller is configured to request the input data during the execution of the computations representing the artificial neural network."**

308. In my opinion, the combination of Buck in view of Wilt discloses or teaches this claim.

309. This claim is substantially the same as Claim 41 and is disclosed or taught by Buck in view of Wilt for substantially the same reasons. Section VII.A.2.p. Whereas Claim 41 recites that the computational stream requests inputs to the artificial neural network, Claim 37 recites that the controller is configured to request the input data for the artificial neural network.

310. The Buck-Wilt system's scheduler (controller) makes resources available to the GPU, and therefore Buck-Wilt teaches that the controller requests the input data to be transferred to GPU memory during execution of the artificial neural network computations. Thus, in my opinion, Buck in view of Wilt discloses or teaches the controller is configured to request the input data during the execution of the computations representing the artificial neural network.

### B. Ground 2: Buck in view of Wilt and nnet Render Obvious Claims 23–26, 34–36, and 38–40

311. The combination of Buck, Wilt, and nnet teaches all elements of claims 23–26, 34–36, and 38–40 of the ’461 patent, rendering obvious those claims. All of the analysis above for Buck in view of Wilt is incorporated herein and nnet additionally teaches the elements of claims 23–26, 34–36, and 38–40 including disclosing a graphical user interface through which a user can provide specific commands to an application performing computations representing an artificial neural network.

#### 1. Motivation to Combine nnet with Buck and Wilt

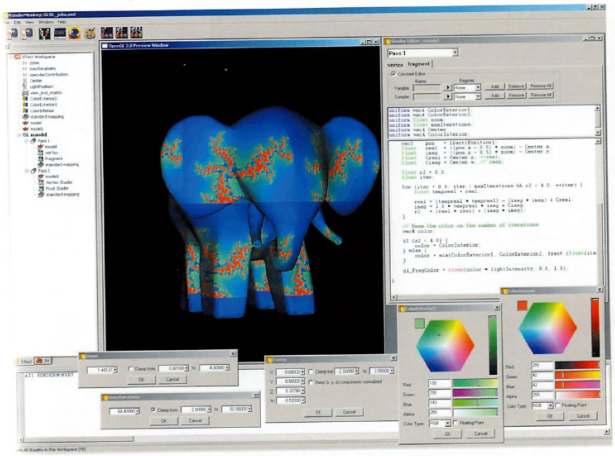
312. As I have explained above, user interfaces, such as GUIs, were widely used before the ’461 patent, including for designing the simulation of artificial neural networks. Section IV.B.3. In my opinion, a POSITA would have been motivated to combine Buck with Wilt and nnet. Indeed, it is my opinion that a POSITA would have been motivated to implement the interactive GUI for simulating artificial neural networks disclosed by nnet in an efficient system for doing so, as disclosed by the combined Buck-Wilt system.

313. In particular, nnet describes a user interface through which users can create, edit, train, and simulate neural networks using MathWorks’ Neural Network Toolbox. Ex1006, 31. nnet describes its GUI as “simple and user friendly,” and, in my opinion, a POSITA would have been motivated to include a GUI as disclosed by

nnet in the Buck-Wilt GPU-accelerated system for implementing artificial neural networks to achieve the benefits of allowing users to interact with an artificial neural network in an accessible way, provide visual feedback in response to user-indicated changes, and display the results of the computations. *Id.*, 91.

314. Command line interfaces and GUIs had been ubiquitous in the field for decades by the time of the '461 patent, and nnet discloses just two examples of the many types of simple user interfaces providing "standard interface functions" available. Indeed, nnet's user interfaces are similar to those used with graphic APIs, such as OpenGL and DirectX, that the '461 patent explicitly characterizes as prior art. Ex1001, 1:62-64, 8:3-5.

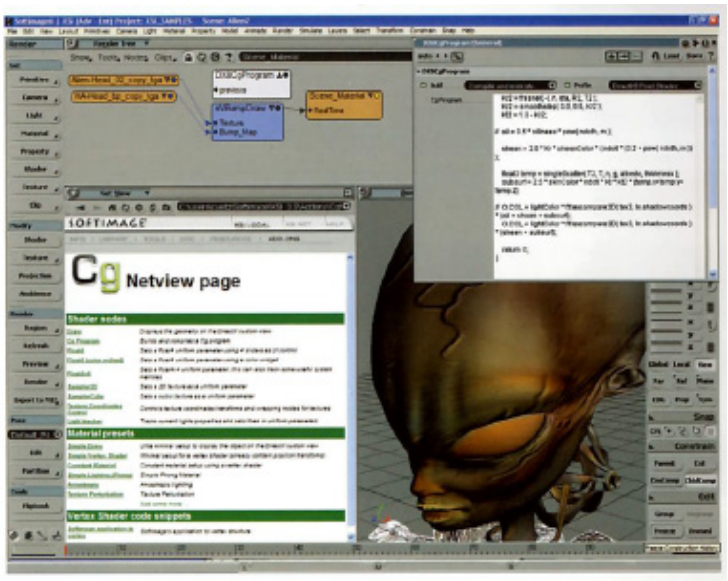
315. As I discussed above, visual tools for interacting with GPUs had been in use for years prior to the '461 patent. For example, the book *OpenGL Shading Language* shows a user interface that allowed a user to interactively adjust parameters used by the GPU for shading an object.



**olor Plate 1** Screen shot of the RenderMonkey IDE user interface. The shader that is procedurally nerating a Julia set on the surface of the elephant is shown in the source code window. Color selection ols and user interface elements for manipulating user-defined uniform variables are also shown.

Ex1016, 27.

316. Additionally, the book *Cg: The Cg Tutorial* describes and shows various Cg viewers that can be used to facilitate user interaction.




Ex1017, 21.

317. The combined Buck-Wilt system explicitly discloses a user input interface through which a "user may enter commands and information into the computer." Ex1004, [0049]–[0050]; Ex1005, [0071]. A typical user interface as of 2005 was a graphical user interface and thus, in my opinion, a POSITA would have been motivated to look to other references and resources, such as nnet, for implementation details regarding this basic, well-known feature of a computer programming system for use with an artificial neural network. In my opinion, a POSITA would have been motivated to include a "simple and user friendly" GUI that provides the benefit of allowing a user to interact with a computing system in

an accessible way while also providing visual feedback and display of the results of the computations. Ex1006, 91. nnet's GUI would also allow users to interact with Buck-Wilt's artificial neural network in an accessible way, inputting information and commands that are processed by the CPU, managed and scheduled by the scheduler, and executed by the GPU.

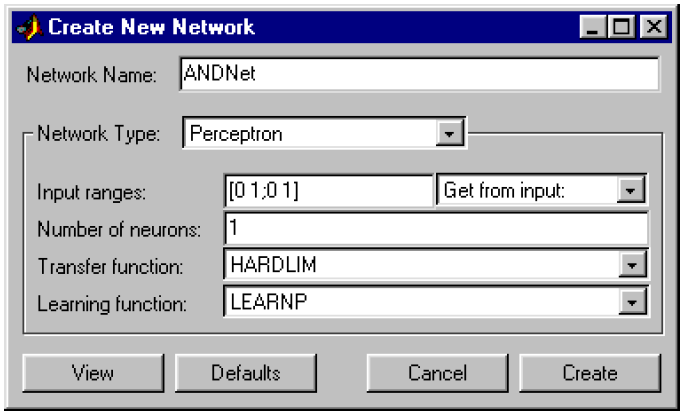
318. Given the common nature of such user interfaces and functions, a POSITA would have had a reasonable expectation of success of implementing a known feature (nnet's GUI) into a known system (Buck-Wilt's GPU-accelerated system) to yield predictable results (e.g., allowing a user to easily provide input).

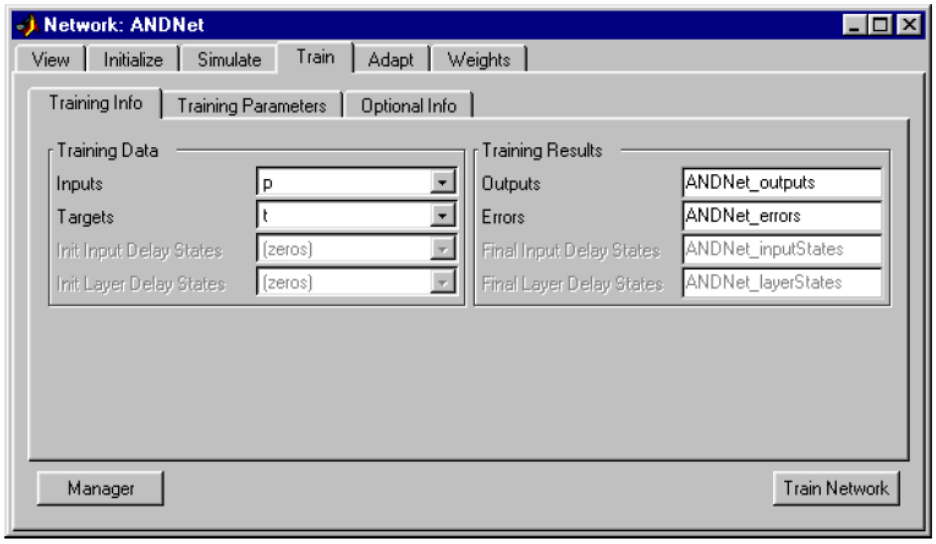
**2. Element-by-Element Analysis**

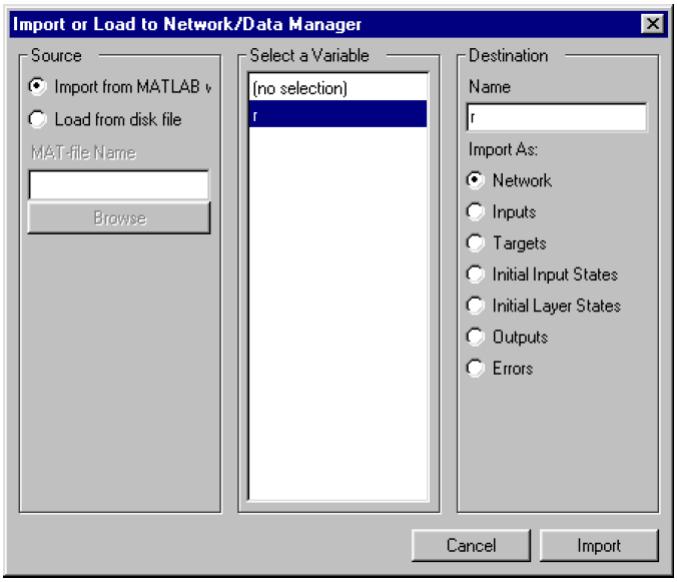
**a) Element 21[a]: "executing, by the at least one CPU, a user interaction stream, the user interaction stream controlling transfer of inputs to the artificial neural network to the first memory partition and the second memory partition"**

319. In my opinion, the combination of Buck in view of Wilt and nnet discloses or renders obvious Element 21[a].

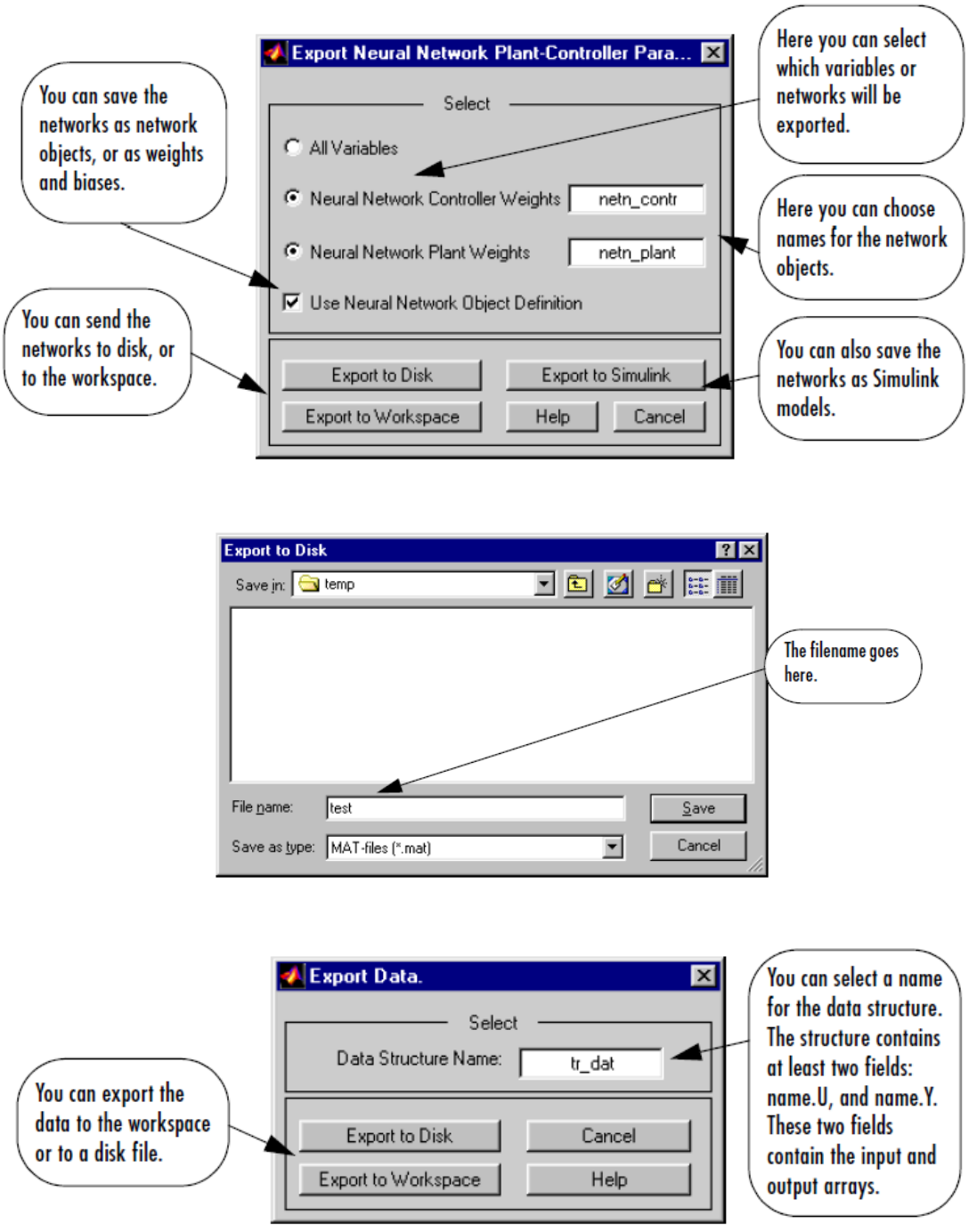
320. nnet discloses an interactive GUI through which a user can define, set, or edit functions, parameters, variables, and sources of inputs to an artificial neural network, such as commands and data. Ex1006, 92–100, 392–413. The GUI also allows users to export, save, display, or otherwise manipulate outputs of the artificial neural network as well. *Id.*, 97–98, 237–243.


Create New Network
Network Name:
ANDNet
Network Type:
Perceptron
Input ranges:
[0 1;0 1]
Get from input:
Number of neurons:
1
Transfer function:
HARDLIM
Learning function:
LEARNP
View
Defaults
Cancel
Create


Network: ANDNet
View
Initialize
Simulate
Train
Adapt
Weights
Training Info
Training Parameters
Optional Info
Training Data
Inputs
p
Targets
t
Init Input Delay States
(zeros)
Init Layer Delay States
(zeros)
Training Results
Outputs
ANDNet_outputs
Errors
ANDNet_errors
Final Input Delay States
ANDNet_inputStates
Final Layer Delay States
ANDNet_layerStates
Manager
Train Network


Import or Load to Network/Data Manager
Source
Import from MATLAB
Load from disk file
MAT-file Name
Browse
Select a Variable
(no selection)
r
Destination
Name
r
Import As:
Network
Inputs
Targets
Initial Input States
Initial Layer States
Outputs
Errors
Cancel
Import

Ex1006, 94, 95, 99.

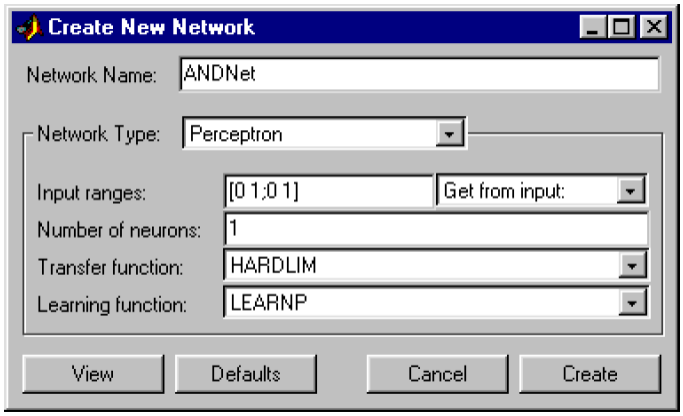



*Id.*, 237, 238, 241.

**b) Claim 23: "The method of claim 21, wherein executing the user interaction stream comprises: controlling setting and editing of computational elements of the computations representing the artificial neural network."**

321. In my opinion, the combination of Buck in view of Wilt and nnet discloses or renders obvious Claim 23.

322. nnet teaches that an application can receive user commands controlling the setting and editing of computational elements of the computations representing the artificial neural network through two kinds of user interfaces. When creating neural networks, nnet allows users to select from a number of predefined simulation, initialization, learning, and self-organizing map functions, or to create their own. *Id.*, 392–413. Users can define their own functions using nnet's command line interface (i.e., the "workspace") and, as illustrated below, through its GUI.




Ex1006, 94.

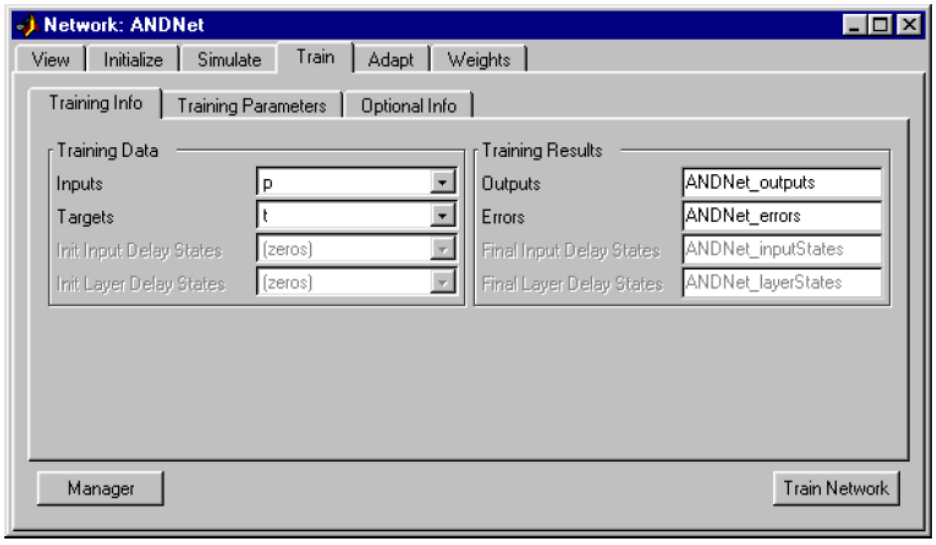
323. Thus, it is my opinion that the combination of Buck in view of Wilt and nnet discloses or renders obvious that executing the user interaction stream

comprises controlling setting and editing of computational elements of the computations representing the artificial neural network.

**c) Claim 24: "The method of claim 21, wherein executing the user interaction stream comprises: controlling setting and editing of parameters of the computations representing the artificial neural network."**

324. In my opinion, the combination of Buck in view of Wilt and nnet discloses or renders obvious Claim 24.

325. nnet teaches that an application can receive user commands controlling setting and editing of parameters of the computations representing the artificial neural network. nnet allows users to change various parameters (e.g., weights and biases) of a neural network when training a network through nnet's GUI. *See, e.g.*, Ex1006, 95.



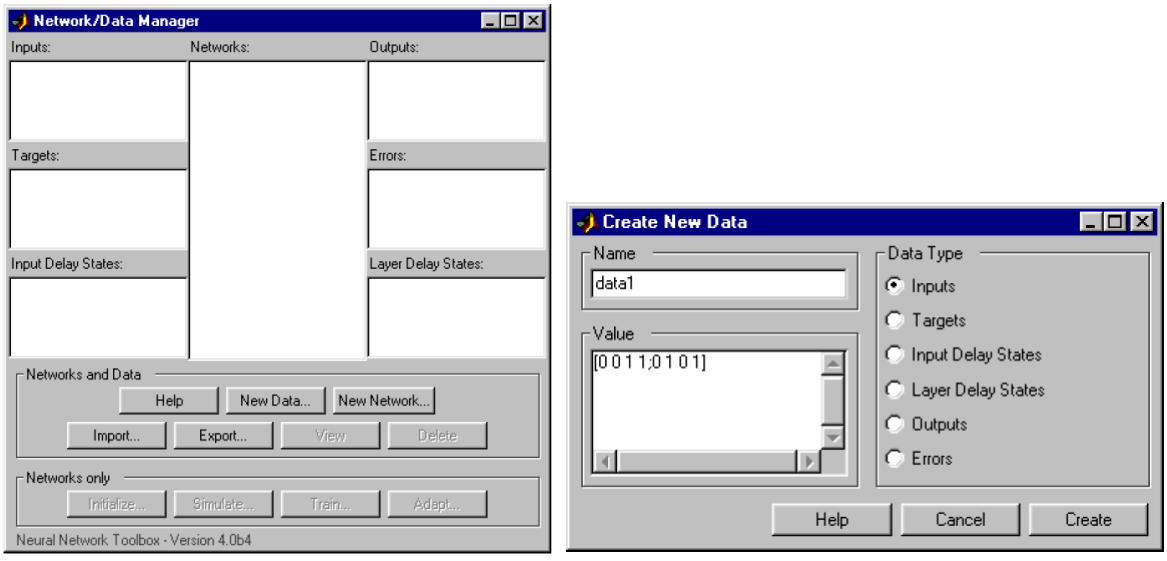

*Id.*

326. Thus, it is my opinion that the combination of Buck in view of Wilt and nnet discloses or renders obvious that executing the user interaction stream comprises controlling setting and editing of parameters of the computations representing the artificial neural network.

**d) Claim 25: "The method of claim 21, wherein executing the user interaction stream comprises: controlling setting and editing of parameters of the inputs to the artificial neural network."**

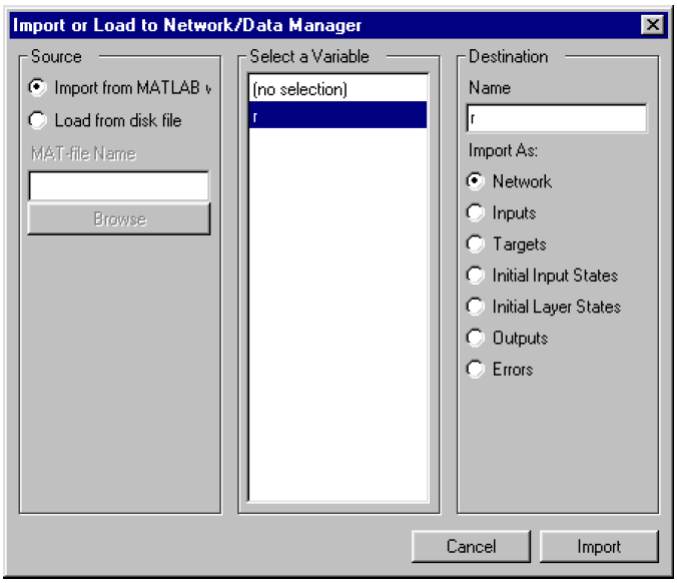
327. In my opinion, the combination of Buck in view of Wilt and nnet discloses or renders obvious Claim 25.

328. nnet teaches that an application can receive user commands controlling setting and editing of parameters of the inputs to the artificial neural network. nnet users can create input files and variables using nnet's command line interface and, as illustrated below, through its GUI. Ex1006, 92–100.



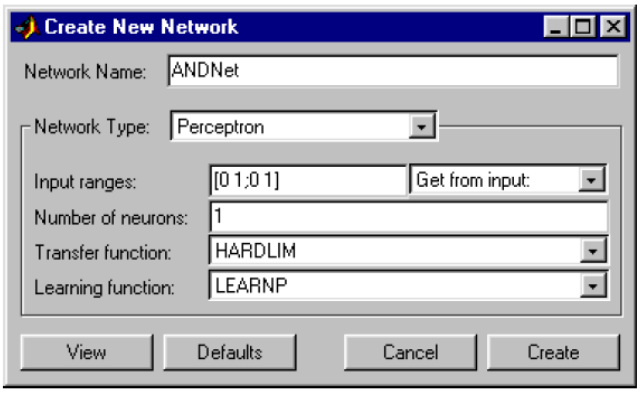

*Id.*, 92, 93.

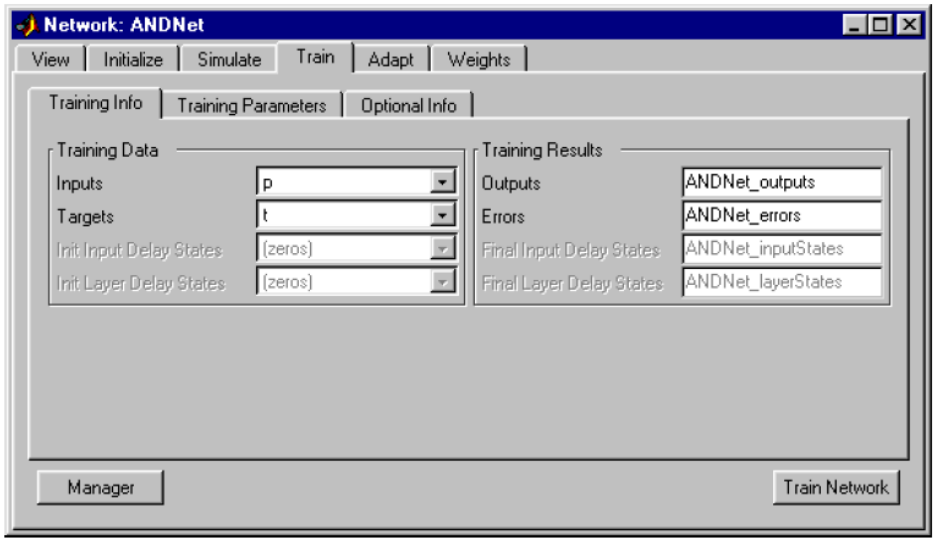
329. Users can also import data from the workspace or load data from a file through the command line interface or the GUI. Ex1006, 92–100.




*Id.*, 99.

330. Regardless of how the input variables or files are obtained, nnet allows users to select input sources and ranges when creating, training, and simulating neural networks.






*Id.*, 94, 95.

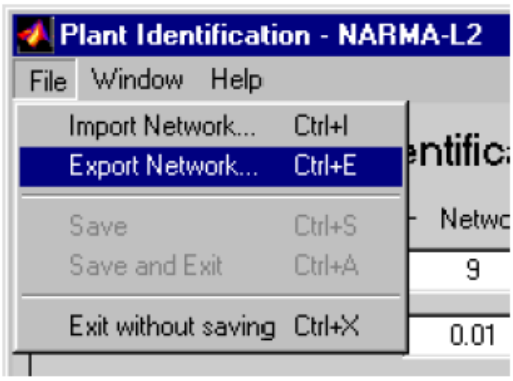
331. The range of data from within an input file or anything that will be used for the data (e.g., the "Training Parameters" shown in nnet) is a "parameter" of the input.

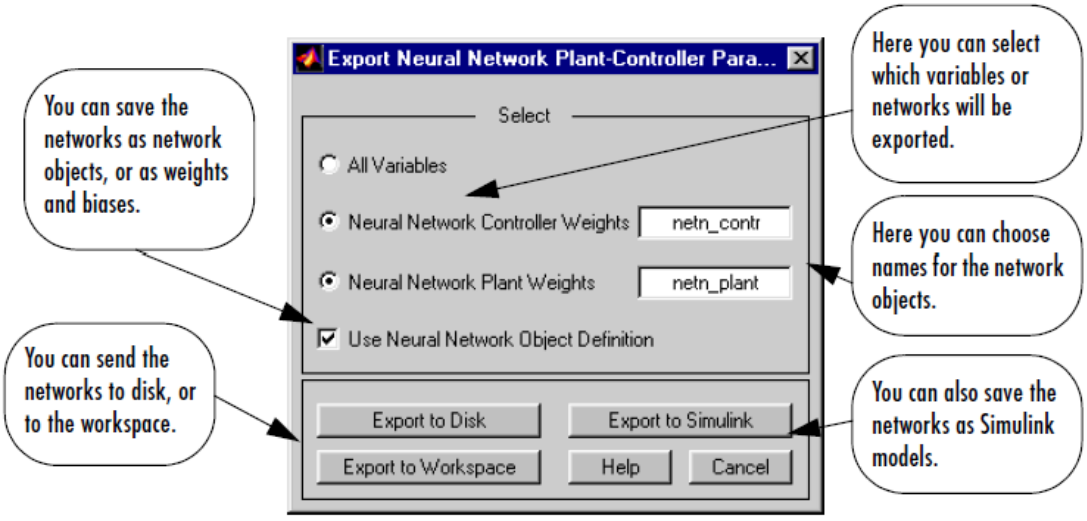
332. Thus, it is my opinion that the combination of Buck in view of Wilt and nnet discloses or renders obvious that executing the user interaction stream comprises controlling setting and editing of parameters of the inputs to the artificial neural network.

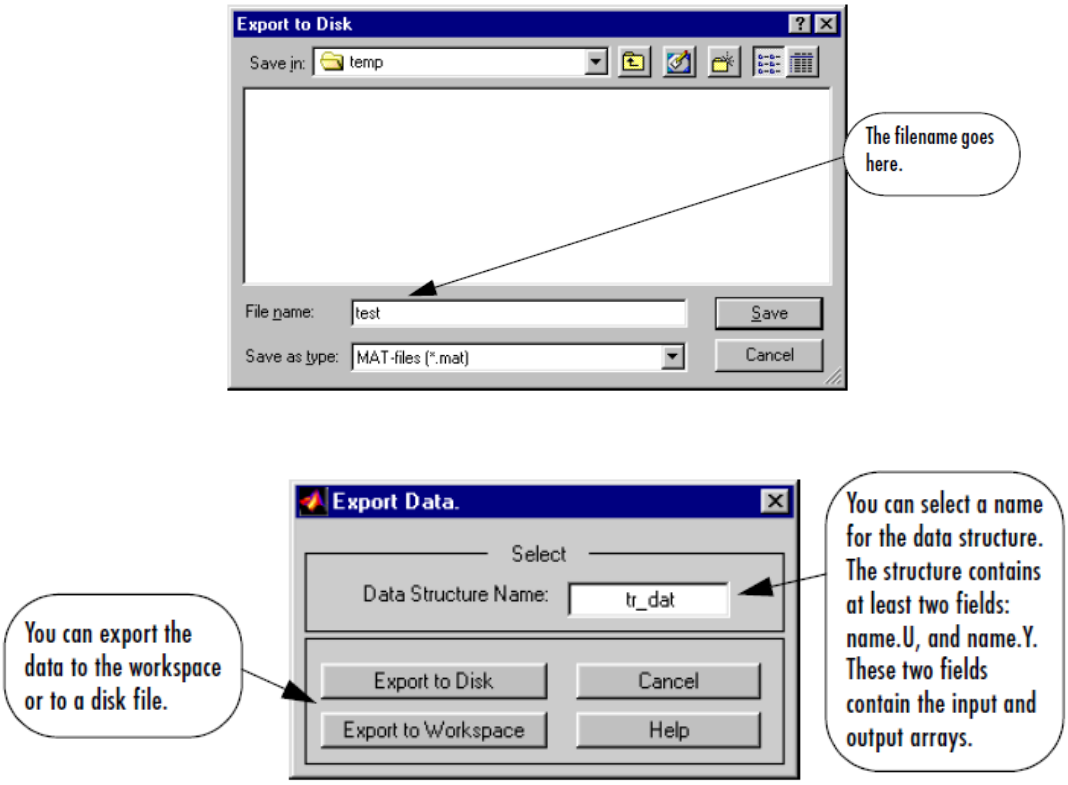
**e) Claim 26: "The method of claim 21, wherein executing the user interaction stream comprises: specifying an output to be saved to disk and/or displayed on a screen."**

333. In my opinion, the combination of Buck in view of Wilt and nnet discloses or renders obvious Claim 26.

334. nnet teaches that an application can receive user commands specifying an output to be saved to disk and/or displayed on a screen. nnet allows users to save neural network outputs, networks themselves, and error reports to disk or to the workspace using nnet's command line interface and, as illustrated below, through its GUI. Ex1006, 97–98, 237–243.

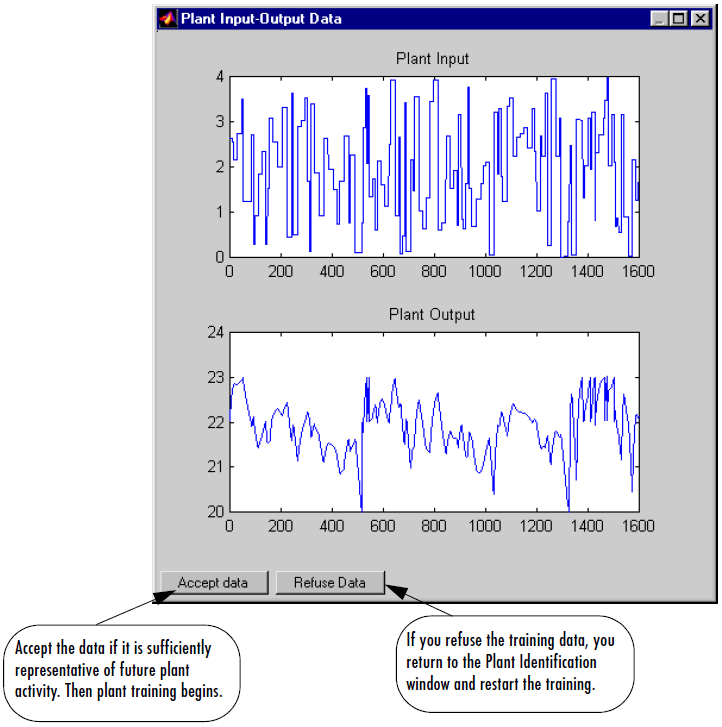


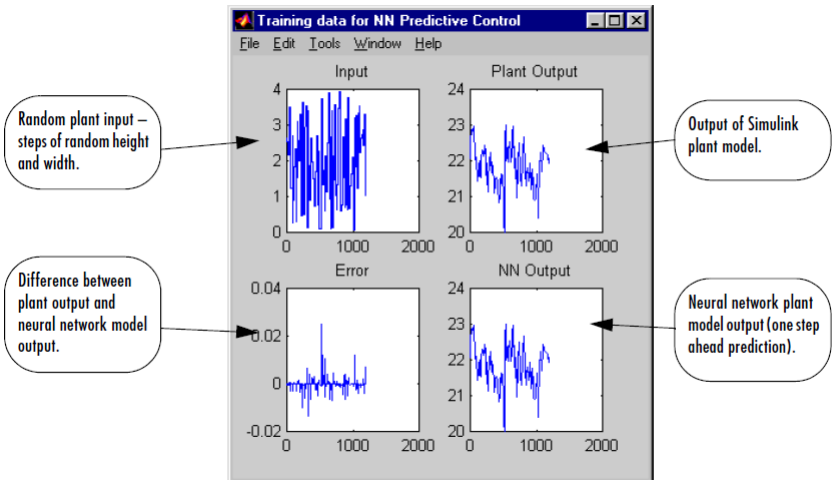



*Id.*, 237, 238, 241.

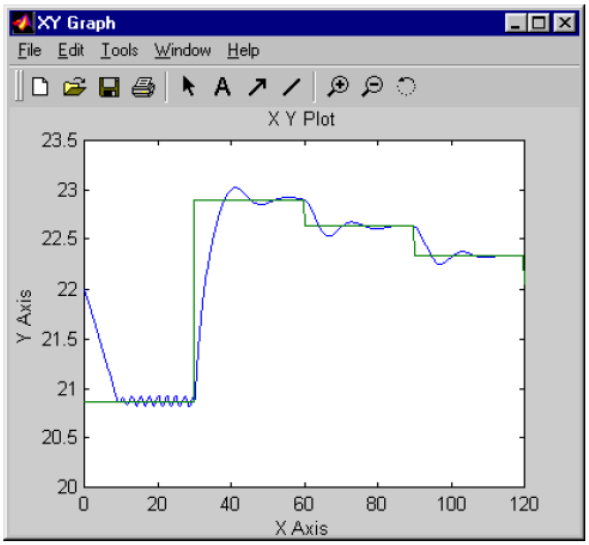
335. In addition to displaying training data and neural network output data, nnet functions "disp" and "display" allow a user to display a neural network's properties and its variables' names and properties. Ex1006, 454 ("Network Use Functions" including "disp" to "Display a neural network's properties" and "display" to "Display a neural network variable's name and properties"), 503 ("disp"), 504 ("display").




Ex1006, 217 (output of "Generate Training Data").




Ex1006, 218 (output of "Train Network").




Ex1006, 219 (output of "Simulate").

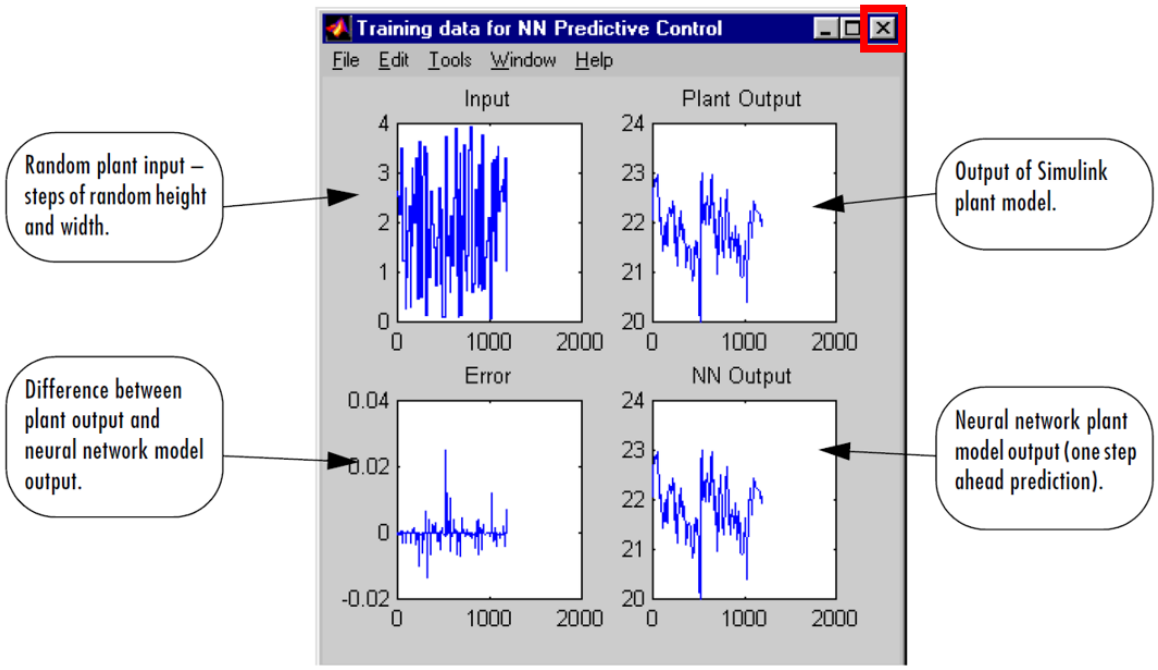
336. Thus, it is my opinion that the combination of Buck in view of Wilt and nnet discloses or renders obvious that executing the user interaction stream comprises specifying an output to be saved to disk and/or displayed on a screen.

**f) Claim 38: "The method of claim 21, wherein the user command causes interruption of the computations representing the artificial neural network."**

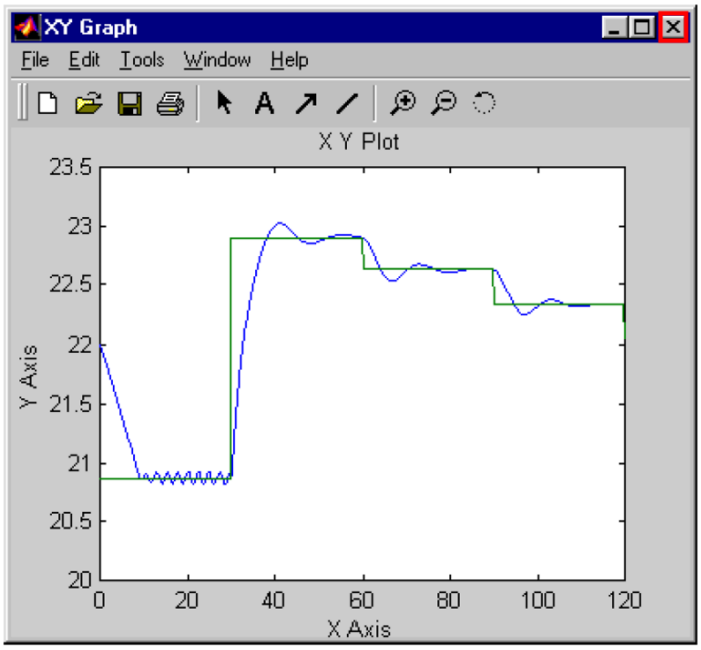
337. In my opinion, the combination of Buck in view of Wilt and nnet discloses or renders obvious Claim 38.

338. nnet teaches that an application can receive user commands that cause interruption of the computations representing the artificial neural network. For example, nnet's GUI provides a user with interface elements that allow a user to close the window or cancel a process. Ex1006, 218–219. These, and other commands, can be high priority commands that preempt the execution of a currently

running command buffer on the Buck-Wilt-nnet system. Furthermore, the ideas of closing a window to cancel a process and a user interface element to cancel a process were well-known long before the ’461 patent.




Ex1006, 218.




Ex1006, 219.

339. Thus, it is my opinion that the combination of Buck in view of Wilt and nnet discloses or renders obvious that the user command causes interruption of the computations representing the artificial neural network.

**g) Claim 39: "The method of claim 21, wherein the user command causes a change in the inputs to the artificial neural network."**

340. It is my opinion that the combination of Buck in view of Wilt and nnet discloses or renders obvious Claim 39.

341. This analysis incorporates by reference my analysis of Claim 25 Sections VII.A.2.q, VII.A.2.o, VII.A.2.t, VII.C.2.b. As I have explained above with respect to Claim 25, the combined Buck-Wilt-nnet system allows a user to provide input controlling setting and editing of parameters of the inputs to the artificial neural

network through a user input interface, including selecting input sources and ranges when creating, training, and simulating neural networks. Ex1006, 92–100.

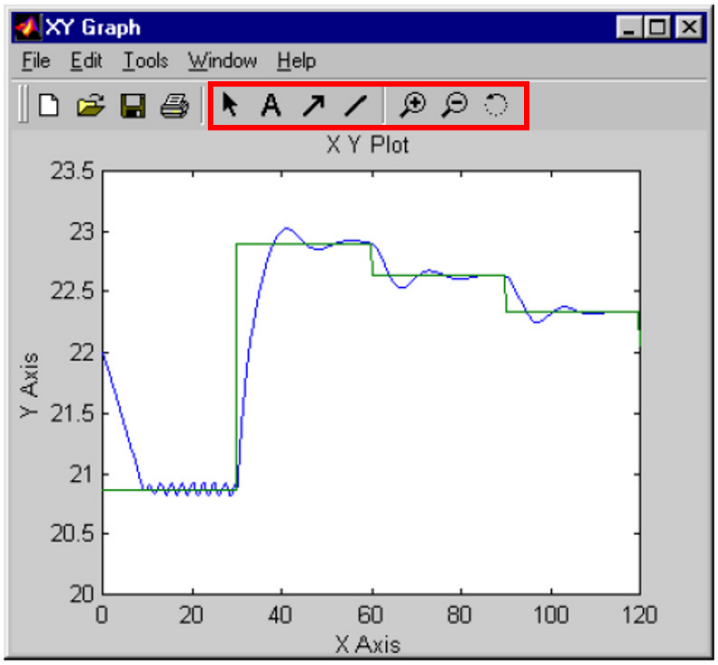
342. Thus, it is my opinion that the combination of Buck in view of Wilt and nnet discloses or renders obvious that the user command causes a change in the inputs to the artificial neural network.

**h) Claim 40: "The method of claim 21, wherein the user command causes a change in display properties of an output of the computations representing the artificial neural network."**

343. In my opinion, the combination of Buck in view of Wilt and nnet discloses or renders obvious claim 40.

344. nnet teaches that an application can receive user commands that cause a change in display properties of an output of the computations representing the artificial neural network.

345. As illustrated below, a user can change display properties of an output of the computations by using standard interface functions like zooming in and out, rotating a plot, and adding labels. Ex1006, 219.




*Id.*, 219.

346. Thus, it is my opinion that the combination of Buck in view of Wilt and nnet discloses or renders obvious that the user command causes a change in display properties of an output of the computations representing the artificial neural network.

**i) Claim 34: "The system of claim 33, wherein the user interactions cause interruption of the computations representing the artificial neural network."**

347. This limitation is substantially the same as Claim 38 and is disclosed or taught by the combination of Buck in view of Wilt and nnet for substantially the same reasons. Claims 34 and 38 differ only in that Claim 34 recites "user interactions" and Claim 38 recites a "user command." The user commands that the combined Buck-Wilt system receives through the user input interface are user interactions. Sections VII.A.m, VII.A.r, VII.B.e.

348. Thus, the Buck, Wilt, and nnet combination teaches that the user interactions cause interruption of the computations representing the artificial neural network.

**j) Claim 35: "The system of claim 33, wherein the user interactions cause a change in inputs to the artificial neural network."**

349. This limitation is substantially the same as Claim 39 and is disclosed or taught by the combination of Buck in view of Wilt and nnet for substantially the same reasons. Sections VII.A.n, VII.A.s, VII.B.f. Claims 35 and 39 differ only in that Claim 35 recites "user interactions" and Claim 39 recites a "user command." The user commands that the combined Buck-Wilt system receives through the user input interface are user interactions.

350. Thus, the Buck, Wilt, and nnet combination teaches that the user interactions cause a change in inputs to the artificial neural network.

**k) Claim 36: "The system of claim 33, wherein the user interactions cause a change in display properties of an output of the computations representing the artificial neural network."**

351. This limitation is substantially the same as Claim 40 and is disclosed or taught by the combination of Buck in view of Wilt and nnet for substantially the same reasons. Sections VII.A.o, VII.A.t, VII.B.g. Claims 36 and 40 differ only in that Claim 36 recites "user interactions" and Claim 40 recites a "user command."

The user commands that the combined Buck-Wilt system receives through the user input interface are user interactions.

352. Thus, the Buck, Wilt, and nnet combination teaches that the user interactions cause a change in display properties of an output of the computations representing the artificial neural network.

### C. Grounds 3 and 4: Buck in view of Wilt, and ANN or GPU Gems Render Obvious Claims 32 and 33–37

353. In my opinion, claims 32 and 33–37 of the ’461 patent would have been obvious over the combination of Buck in view of Wilt, and ANN or GPU Gems. All of the analysis above for Buck in view of Wilt is incorporated herein and ANN and GPU Gems each additionally teaches the elements of claims 32 and 33–37 including disclosing acquiring real-time data from a video camera to use as input data for its neural network simulation.

#### 1. Motivation to Combine ANN or GPU Gems with Buck and Wilt

354. Computationally intensive and repetitive, computer vision algorithms were a common early application for GPGPU computing. Ex1008, 649–66. Several “fully interactive,” “real-time computer vision programs” for execution by a GPU were publicly available prior to the ’461 patent. Section IV.B.5. ANN and GPU Gems provide examples of such interactive and real-time computer vision programs, which are described in those references as being implemented on GPUs. In my

opinion, a POSITA would have been motivated to combine Buck, Wilt, and ANN or GPU Gems. Indeed, a POSITA would have understood that the combination of Buck, Wilt, and ANN would involve incorporating a known feature (receiving input from a video camera) into a known system (Buck-Wilt's GPU-accelerated system) ready for improvement to yield predictable results (e.g., receiving real-time video data to be processed by a neural network implemented on a GPU). Section IV.B.5.

**a) ANN**

355. A POSITA seeking to execute the multi-layer artificial neural network for real-time object tracking disclosed in ANN would have sought out advanced systems with a programmable GPU, such as that disclosed in the combination of Buck and Wilt. As I have explained above, ANN discloses using a "[t]hree layer MLP neural network" to "recognize and trace [a] ball in real time" for robots playing soccer. Ex1007, 3. Although ANN used NVIDIA's GF6000 GPU for its testing, in my opinion, a POSITA would have understood that ANN is ultimately agnostic as to the specific GPU architecture for implementing the artificial neural network. Thus, to achieve even further efficiency in computation, it is my opinion that a POSITA would have been motivated to implement ANN's artificial neural network on state-of-the-art GPU-accelerated parallel processing systems, such as that disclosed in Buck-Wilt. Indeed, in my opinion, a POSITA would have understood that the GPU-accelerated system for implementing neural networks disclosed in

Buck-Wilt would be an ideal system for simulating the artificial neural network of ANN. The Buck-Wilt system is specifically designed to accelerate the parallel computations required for simulating artificial neural networks. Buck emphasizes that its system "exploit[s] the parallelism and high-memory bandwidth of GPUs" as well as "flexibility introduced by programmable pixel shaders" to implement machine learning algorithms, such as training and testing artificial neural networks. Ex1004, [0010]–[0011], [0055]. Thus, in my opinion, a POSITA would have understood the Buck-Wilt system to be well-suited for implementing the artificial neural networks underlying ANN's computer vision algorithm.

**b) GPU Gems**

356. Similarly, in my opinion, a POSITA reviewing GPU Gems would have been motivated to implement GPU Gems's GPU-implemented computer vision algorithm for real-time feature tracking on the GPU-accelerated parallel processing system in Buck-Wilt. GPU Gems describes receiving and processing "real-time" input from a video camera to perform feature tracking. Ex1008, 650–51. A POSITA reviewing GPU Gems's discussion of using GPUs "to accelerate computer vision computation and free up the CPU for other tasks," would have been motivated to implement GPU Gems's feature-tracking algorithm on state-of-the-art GPU-accelerated parallel processing systems, such as that disclosed in Buck-Wilt. *Id.*, 650. Indeed, in my opinion, a POSITA would have understood that Buck-Wilt's

optimized system for implementing "general matrix operations" involving "large amounts of data and processing" disclosed in would be an ideal system to accelerate GPU Gems's computer vision program. The Buck-Wilt system is specifically designed to accelerate the parallel computations required for simulating artificial neural networks. Buck emphasizes that its system "exploit[s] the parallelism and high-memory bandwidth of GPUs" as well as "flexibility introduced by programmable pixel shaders" to implement computationally intensive non-graphics applications. Ex1004, [0010]–[0011], [0055]. Thus, in my opinion, a POSITA would have understood Buck-Wilt's system to be well-suited for implementing GPU Gems' computer vision algorithm.

**c) Implementing the Computer Vision Method of ANN or GPU Gems in Buck-Wilt's System**

357. The computer vision techniques discussed by ANN or GPU Gems are just two examples of well-known GPGPU computing applications that a POSITA would have understood could be performed on Buck/Wilt's GPU-accelerated system. Although Buck in view of Wilt does not explicitly disclose receiving input from a video camera, its system includes a number of input devices that are "not shown." Ex1004, [0049]–[0050]; Ex1005, [0071]. These devices are "connected to the processing unit 120 through a user input interface 160 that is coupled to the system bus 121" or through "other interface and bus structures." Ex1004, [0049]–[0050]; Ex1005, [0071]. Furthermore, the system uses "predominantly DMA-based

input," which is commonly used with "peripheral" devices, "such as a digital camera," and allows those devices to "read from or write to memory without directly involving the host processor." Ex1005, [0085], [0093]. Wilt explains that this memory management method locks "the memory region where a digital camera, for example, is to write the image data from a picture it is about to take," specifying that memory location to the camera, and allowing the camera to write directly to that location. *Id.*, [0085].

358. At the time of the '461 patent, real-time computer vision techniques were a common application for GPU computing. Section IV.B.4–5. Computer vision tasks often "exceed[ed] the real-time capabilities of the CPU," but "map[ped] efficiently onto the modern GPU, whose programmability allows a wide range" of algorithms to be implemented. Ex1008, 649–66. Indeed, a number of "fully interactive," "real-time computer vision programs" were publicly available at the time, and, in my opinion, a POSITA would have understood how to implement such programs on Buck-Wilt's GPU accelerated system. Ex1008, 649-66 ("OpenVIDIA GPU computer vision library").

359. It is my opinion that a POSITA reading ANN or GPU Gems would have understood not only that ANN or GPU Gems's object recognition method could be performed on a GPU (such as the optimized GPU of Buck-Wilt), but also how to use the Buck-Wilt system's GPU-accelerated system to implement it. And, in my

opinion, a POSITA would have had a reasonable expectation of success in the combination because Buck-Wilt's system is configured to "exploit the parallelism and high-memory bandwidth of GPUs" and "take full advantage of the new flexibility introduced by programmable pixel shaders" and operate on "large floating point matrices and vectors" as contemplated in ANN. Ex1004, [0010]–[0012]; Ex1007, Abstract, 1, 3.

360. Additionally, in my opinion, a POSITA would have been motivated by ANN's discussion of receiving artificial neural network input data from a video camera and GPU Gems's discussion of receiving real-time data from a video camera to incorporate this source of input into the Buck-Wilt optimized GPU-accelerated system. The combined Buck-Wilt system explicitly discloses receiving input from various external input devices, including devices that generate audio and/or video input data in real-time, through its user input interface connected to the CPU. Ex1004, [0049]–[0050]; Ex1005, [0071], [0107]. Thus, in my opinion, a POSITA would have found it obvious to add video camera input to Buck-Wilt's system as disclosed by ANN or GPU Gems and would have had a reasonable expectation of success in doing so.

**2. Element-by-Element Analysis**

**a) Claim 32: "The method of claim 21, further comprising: generating the inputs with a video camera during execution of the computations."**

361. This analysis incorporates by reference my analysis of Claims 21 and 32 from Ground 1. Section VII.A.2.a, VII.A.2.l.

362. In my opinion, the combination of Buck in view of Wilt, and ANN or GPU Gems discloses or renders obvious Claim 32.

363. As I have discussed above for Claim 21, Buck's CPU receives and processes input data—including from input devices connected through the user input interface and other interfaces—translating it into texture data to be transmitted to the GPU. Ex1004, [0069]–[0075]; *see also id.* at [0008], [0012], [0055], [0060]–[0061], [0088]–[0091], [0235]–[0250], Figs. 9, 15, 16. In the Buck-Wilt-ANN or GPU Gems combination, the input data received by the CPU is video data generated by a video camera. Ex1007, 3-5; Ex1008, 649-66.

a) **ANN**

364. As I have explained above, ANN discloses receiving visual data from a soccer-playing robot in real-time to "recognize" and "trace" the ball. Ex1007, 2–3. ANN explains that systems having a GPU with a "video in" function can retrieve image information "directly" from the video camera and store that information in GPU memory, allowing the system to avoid "transfer[ring] data between CPU and

GPU" and "speed [up] the process." *Id.* GPUs that do not have a "video in" function, such as the one in the ANN system, require the CPU to receive, process, and transfer video data to GPU memory. Thus, real-time video data from ANN's video camera is input to the Buck-Wilt system CPU through the user input interface (or another interface) and is transferred from the CPU to GPU memory for processing. Ex1007, 5.

b) **GPU Gems**

365. Similarly, GPU Gems discloses receiving and processing video data from a video camera to "detect and track distinctive points (feature) in real time as the camera pans." Ex1008, 661. Video data generated by GPU Gems's video camera is input to the Buck-Wilt system CPU through the user input interface (or another interface) and is transferred from the CPU to GPU memory for processing. *See* Ex1008, 649–66.

c) **Implementing the Computer Vision Method of ANN or GPU Gems in Buck-Wilt's System**

366. Furthermore, the combined Buck-Wilt system's resource management system "facilitate[s] the host microprocessor's performance of useful work while the coprocessor hardware is running." Ex1005, [0114]. Therefore, the Buck-Wilt-ANN or GPU Gems CPU can continue to receive and store input data from the video camera in real-time while the GPU is executing computations. This real-time video

camera data is used as input to the Buck-Wilt artificial neural network. *Id.* Thus, in the Buck-Wilt-ANN or GPU Gems combination, the CPU receives and processes the real-time video data input and transfers the resulting texture data to the GPU during the computations representing the artificial neural networks. Ex1007, 5.

367. Thus, in my opinion, the combination of Buck in view of Wilt, and ANN or GPU Gems discloses or renders obvious generating the inputs with a video camera during execution of the computations.

**b) Claim 33**

**i. 33[pre]: "A system for executing computations representing an artificial neural network, the system comprising:"**

368. In my opinion, the combination of Buck in view of Wilt, and ANN or GPU Gems discloses or teaches this limitation.

369. This limitation recites nothing more than is included in Element 21[pre] and is disclosed or taught by Buck in view of Wilt for substantially the same reasons. Section VII.A.2.a.i, VII.A.2.q.i.

**ii. 33[a]: "a camera to acquire input data for the artificial neural network;"**

370. In my opinion, the combination of Buck in view of Wilt, and ANN or GPU Gems discloses or teaches this limitation.

371. This limitation substantially similar to Element 21[pre] and Claim 32 and is disclosed or taught by Buck in view of Wilt and ANN or GPU Gems for

substantially the same reasons. Section VII.A.2.a.i, VII.A.2.l. Element 33[a] differs from Claim 32 only in that it recites a "camera" and not a "video camera," and that the camera is part of the system for executing the computations representing an artificial neural network.

372. As I have explained above with respect to Claim 32, the Buck, Wilt, and ANN or Buck, Wilt, and GPU Gems combination discloses generating real-time input to the artificial neural network computations executed by the GPU with a video camera while the executions are performed. Section VII.C.2.a; Ex1007, 2–3, 5; Ex1008, 649–66. Thus, the combined Buck, Wilt, and ANN or GPU Gems system receives and processes real-time data from a video camera. For those same reasons, it is my opinion that the Buck, Wilt, and ANN or GPU Gems combination discloses a video camera that generates inputs to the artificial neural network.

**iii. 33[b]: "a first memory partition;"**

373. In my opinion, the combination of Buck in view of Wilt, and ANN or GPU Gems discloses or teaches this limitation.

374. This limitation recites nothing more than is included in Element 21[pre] and is disclosed or taught by Buck in view of Wilt for substantially the same reasons. Sections VII.A.2.a.i, VII.A.2.q.i.

**iv. 33[c]: "a second memory partition;"**

375. In my opinion, the combination of Buck in view of Wilt, and ANN or GPU Gems discloses or teaches this limitation.

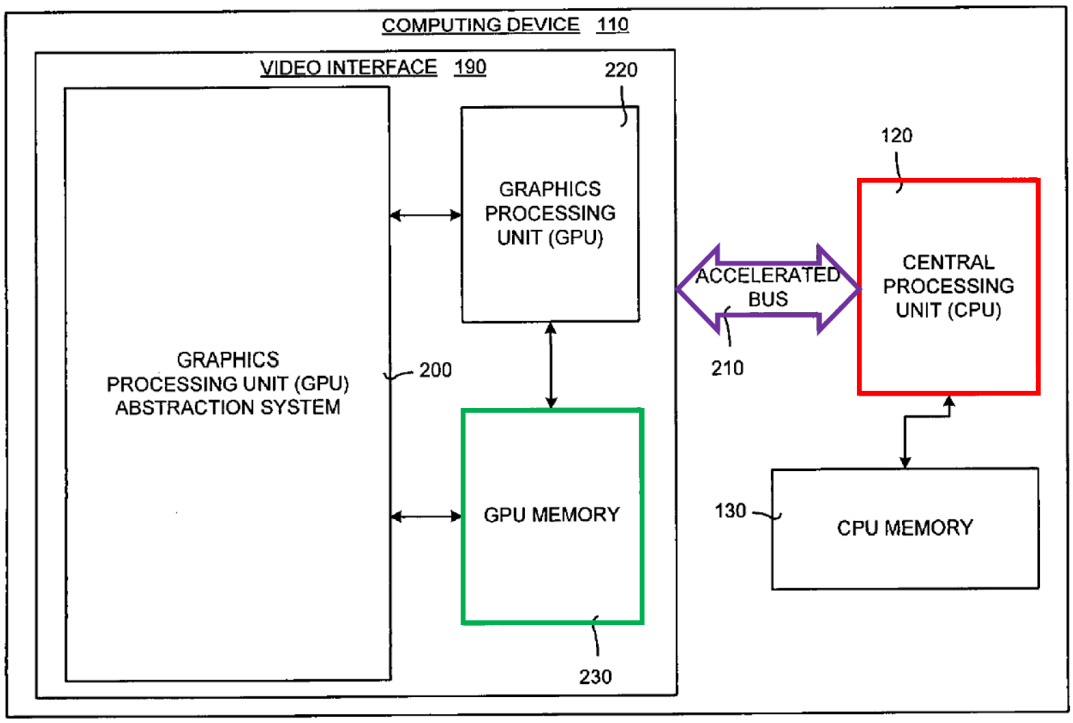
376. This limitation recites nothing more than is included in Element 21[pre] and is disclosed or taught by Buck in view of Wilt for substantially the same reasons. Sections VII.A.2.a.i, VII.A.2.q.i.

**v. 33[d]: "at least one central processing unit (CPU), operably coupled to the camera, the first memory partition, and the second memory partition, to execute a user interaction stream, the user interaction stream controlling transfer of the input data acquired by the camera to the first memory partition and the second memory partition during execution of the computations representing the artificial neural network;"**

377. In my opinion, the combination of Buck in view of Wilt, and ANN or GPU Gems discloses or teaches this limitation.

378. This limitation is substantially the same as Element 21[a] and is disclosed or taught by Buck in view of Wilt for substantially the same reasons. Sections VII.A.2.a.ii, VII.A.2.q.ii.

379. Furthermore, as I have illustrated below, Buck's central processing unit (CPU 120, annotated in red) is "in communication with" video interface 190, of which the first and second memory partitions (within GPU memory 230, annotated in green) are part, via accelerated bus 210 (annotated in purple). Ex1004, [0049]–[0050], [0055]–[0057]; *see also* Ex1005, [0067]–[0071].




Ex1004, Fig. 2 (annotated).

380. A video camera providing input to the Buck-Wilt-ANN or GPU Gems system would have been connected to the CPU through the system bus and a user input interface (or other interface and bus structures, such as "a parallel port, game port or a universal serial bus (USB)"). Ex1004, [0050]; *see also* Ex1005, [0071]. Thus, the camera is operably coupled to the CPU.

381. As I have explained above, the Buck-Wilt system's resource management system "facilitate[s] the host microprocessor's performance of useful work while the coprocessor hardware is running." Ex1005, [0114]. Therefore, the Buck-Wilt-ANN or GPU Gems CPU can continue to receive and store input data from the video camera in real-time while the GPU is executing computations.

Because this real-time video camera data is used as input to the Buck-Wilt-ANN or GPU Gems artificial neural network, the CPU transfers the data to the GPU on while the GPU continues to perform computations. Ex1007, 5; Ex1008, 649-66.

382. And, as I have explained above with respect to Claim 21, the combined Buck-Wilt system discloses transferring data or instructions received through the user input interface, including from input devices other than user input devices, to GPU memory (e.g., in a segment for data or a segment for instructions). Ex1004, [0069]–[0075]; *see also id.* at [0012], [0055], [0060]–[0061], [0088]–[0091], [0235]–[0250], Figs. 9, 15, 16. Thus, it is my opinion that the combination of Buck in view of Wilt, and ANN or GPU Gems discloses executing a user interaction stream controlling transfer of the input data acquired by the camera to the first memory partition and the second memory partition.

383. Therefore, it is my opinion that the combination of Buck in view of Wilt, and ANN or GPU Gems teaches that the CPU is operably coupled to the camera and the first and second memory partitions and that the CPU executes a user interaction stream controlling transfer of the input data acquired by the camera to the first memory partition and the second memory partition during execution of the computations representing the artificial neural network.

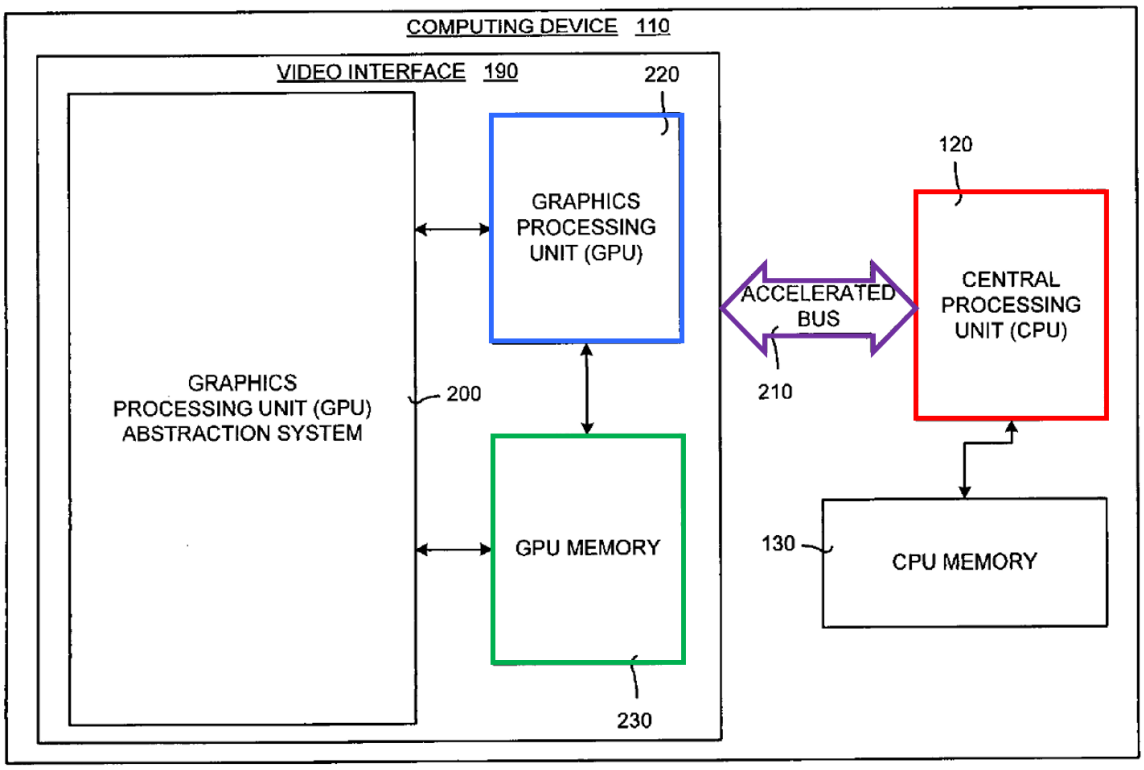
**vi. 33[e]: "a processing unit, operably coupled to the first memory partition, the second memory partition, and the at least one CPU, to execute a computational stream, the computational stream controlling transfer of the input**

**data from the first memory partition and the second memory partition during execution of the computations representing the artificial neural network, the execution of the computations representing the artificial neural network occurring while the camera is acquiring the input data; and"**

384. In my opinion, the combination of Buck in view of Wilt, and ANN or GPU Gems discloses or teaches this limitation.

385. This limitation is substantially the same as Element 21[b] and is disclosed or taught by Buck in view of Wilt for substantially the same reasons. Sections VII.A.2.a.ii, VII.A.2.q.ii.

386. As illustrated below, Buck's processing unit (GPU 220, annotated in blue) is "in communication with" first and second memory partitions (within GPU memory 230, annotated in green) and with CPU 120 (annotated in red) via accelerated bus 210 (annotated in purple). Ex1004, [0049]–[0050], [0055]–[0057]; *see also* Ex1005, [0067]–[0071].




Ex1004, Fig. 2 (annotated).

387. As I have explained above with respect to Claim 21, the Buck-Wilt system discloses that the computational stream controls the data exchange between the user interaction stream and the computational stream. Section VII.A.2.a.

388. Furthermore, Wilt discloses that when a command buffer (i.e., part of the computational stream) requires a specific resource, the scheduler makes the required resource available. Ex1005, [0024]. Therefore, if the required resources included the inputs to the artificial neural network, the command buffer (part of the computational stream) would request the inputs to the artificial neural network and the scheduler would transfer this input data from the CPU to a segment of the GPU memory for input data. Thus, Buck-Wilt-ANN or GPU Gems discloses that the

computational stream controls transfer of the input data from the first memory partition and the second memory partition.

389. And, as I have explained above with respect to Element 33[d], the Buck-Wilt-ANN or GPU Gems system's CPU can continue to receive and store input data from the video camera in real-time while the GPU is executing computations. Ex1005, [0114]; Ex1007, 5. Thus, Buck-Wilt-ANN or GPU Gems discloses that the computations representing the artificial neural network occur while the camera is acquiring the input data.

390. Therefore, in my opinion, a POSITA would have understood the combination of Buck in view of Wilt, and ANN or GPU Gems teaches that the processing unit is operably coupled to the first and second memory partitions and the CPU and that the processing unit executes a computational stream that controls transfer of the input data from the first memory partition and the second memory partition during execution of the computations representing the artificial neural network that occur while the camera is acquiring the input data.

**vii. 33[f]: "a controller, operably coupled to the at least one CPU and the processing unit, to queue user interactions received by the user interaction stream during the execution of the computations representing the artificial neural network for performance at times selected to avoid data corruption."**

391. In my opinion, the combination of Buck in view of Wilt, and ANN or GPU Gems discloses or teaches this limitation.

392. This limitation is substantially the same as Elements 21[e] and 21[f] and is disclosed or taught by Buck in view of Wilt for substantially the same reasons. Sections VII.A.2.a.vi, VII.A.2.a.vii, VII.A.2.q.vii.

**c) Claim 34: "The system of claim 33, wherein the user interactions cause interruption of the computations representing the artificial neural network."**

393. This analysis incorporates by reference my analysis of claims 33 and 38. Sections VII.A.2.q, VII.A.2.m, VII.A.2.r, VII.C.2.b.

**d) Claim 35: "The system of claim 33, wherein the user interactions cause a change in inputs to the artificial neural network."**

394. This analysis incorporates by reference my analysis of claims 33 and 39. Sections VII.A.2.q, VII.A.2.n, VII.A.2.s, VII.C.2.b.

**e) Claim 36: "The system of claim 33, wherein the user interactions cause a change in display properties of an output of the computations representing the artificial neural network."**

395. This analysis incorporates by reference my analysis of claims 33 and 40. Sections VII.A.2.q, VII.A.2.o, VII.A.2.t, VII.C.2.b.

**f) Claim 37: "The system of claim 33, wherein the controller is configured to request the input data during the execution of the computations representing the artificial neural network."**

396. This analysis incorporates by reference my analysis of claims 33 and 41. Sections VII.A.2.q, VII.A.2.p, VII.A.2.u, VII.C.2.b.

### D. Grounds 5 and 6: Buck in view of Wilt, ANN or GPU Gems, and nnet Render Obvious Claims 34–36

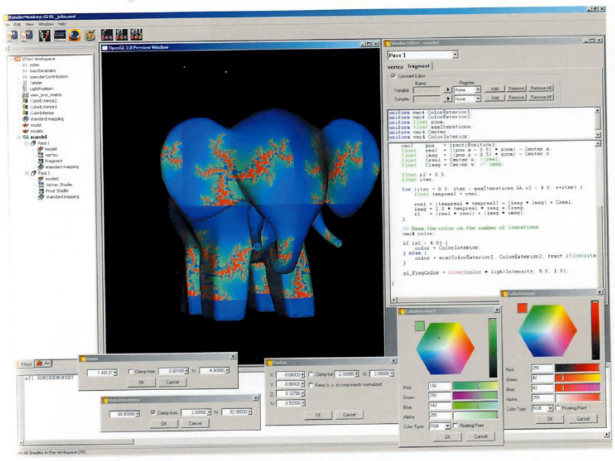
397. In my opinion, claims 34–36 of the '461 patent would have been obvious over Buck in view of Wilt, ANN or GPU Gems, and nnet. All of my analysis above for Buck in view of Wilt and for Buck in view of Wilt and further in view of ANN or GPU Gems is incorporated herein and nnet additionally teaches elements of claims 34–36 and 38–40 including disclosing a graphical user interface through which a user can provide specific commands to an application performing computations representing an artificial neural network.

398. For the same reasons a POSITA would have been motivated to apply nnet's GUI in the Buck/Wilt system, *see* Section VII.B.1, the POSITA would have been motivated to use the GUI when ANN's or GPU Gems' computer vision algorithms are implemented on the Buck/Wilt system. I therefore incorporate by reference my analysis of the motivations to combine Buck, Wilt, and ANN or GPU Gems, and Buck, Wilt, and nnet. Sections VII.A.1, VII.B.1, VII.C.1. GPU-accelerated systems for applying artificial neural networks to perform real-time detecting and tracking of an object in a video at the time were "fully interactive." Section IV.B.3; Ex1008, 651.

399. GUIs were widely used with GPU-accelerated systems to allow users to interact with and control the systems well before the '461 patent. Section IV.B.2. In particular, GPU-accelerated systems for applying artificial neural networks to

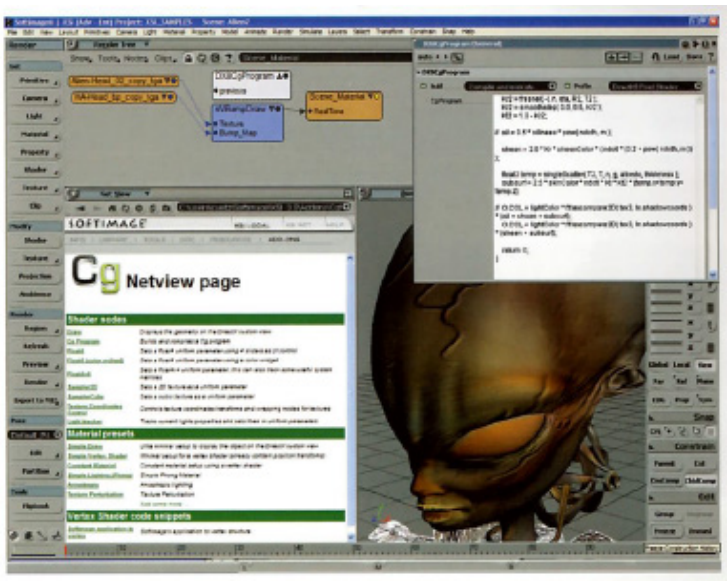
perform real-time detecting and tracking of an object in a video at the time were "fully interactive." Section IV.B.4; Ex1008, 651.

400. As I have explained above, nnet describes its GUI as "simple and user friendly," and, in my opinion, a POSITA would have found it obvious that the combined Buck-Wilt-ANN or GPU Gems system includes a GUI as part of its user input interface 160 through which a user can "enter commands and data." Ex1004, [0049]–[0050]; Ex1005, [0071]. And, as I discussed above, visual tools for interacting with GPUs had been in use for years prior to the '461 patent.



**olor Plate 1** Screen shot of the RenderMonkey IDE user interface. The shader that is procedurally nerating a Julia set on the surface of the elephant is shown in the source code window. Color selection ols and user interface elements for manipulating user-defined uniform variables are also shown.

Ex1016, 27.




Ex1017, 41; *see also id.*, 39–40, 44.

401. At a minimum, in my opinion, a POSITA would have been motivated to implement a GUI that provides the benefit of allowing a user to interact with a computing system in an accessible way while also providing visual feedback and display of the results of the computations. Ex1006, 91.

402. nnet's user interfaces are similar to those available for use with admitted prior art graphic APIs, such as OpenGL and DirectX. Ex1001, 1:62-64, 8:3-5. Indeed, in my opinion, POSITA would have had a reasonable expectation of success of implementing a known feature (nnet's GUI) into a known system (the combined Buck-Wilt-ANN or GPU Gems's GPU-accelerated system) to yield predictable results (e.g., allowing a user to easily provide input). It is therefore my opinion that a POSITA would have been motivated, and would have found it obvious

to look to nnet's teachings of a graphical user interface and its assorted functions to implement in the combined Buck-Wilt-ANN or Buck-Wilt-GPU Gems system.

## VIII. SECONDARY CONSIDERATIONS

403. I am not aware of any evidence of secondary considerations of nonobviousness relied upon by Patent Owner. To the extent Patent Owner is permitted to provide any evidence of secondary considerations to rebut my opinions expressed herein, I reserve the right to respond by supplementing and/or amending this declaration or submitting a reply declaration to consider the alleged secondary considerations of nonobviousness.

## IX. RIGHT TO SUPPLEMENT

404. I reserve the right to supplement this declaration, if appropriate, should additional information come to light that is relevant to my opinions in this declaration. I also reserve the right to consider and comment on additional evidence, e.g., objective evidence of nonobviousness, that may be presented by Patent Owner or its experts.

405. I will continue my study and evaluation of relevant prior art as these proceedings continue, at least up until the end of the period prescribed by the relevant rules and regulations. In the event further issues arise that concern the subject matter in my declaration, I expressly reserve the right to supplement, modify, or amend my opinions as stated in this declaration as well as the bases supporting my opinions. I

specifically reserve the right to supplement and/or amend this declaration at such time that the Board defines disputed terms in the Challenged Claims, particularly if the Board adopts alternate instructions not advanced by Petitioners. I further reserve the right to supplement and/or amend this declaration to submit a reply declaration based on any information or contentions expressed by Patent Owner or its experts attempting to rebut my opinions in this declaration.

## X. CONCLUSION

406. For the reasons set forth above, it is my opinion that the Challenged Claims of the '461 patent are invalid.

407. I hereby declare that all statements made herein of my own knowledge are true, and that all statements made on information and belief are believed to be true, and that these statements were made with knowledge that willful false statements and the like are so made punishable by fine or imprisonment, or both, under section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the Petition for *inter partes* review.

Executed on March 5, 2025

San Diego, California



By: _____

Dr. Tajana S. Rosing