**AvantZero**

Random data-generating algorithm for (experimental)

Quake III Machinima (post)-production

**TECHNICAL PERSPECTIVES**

Veenstra, J.P.

Machinima Artist, Software Developer and Lecturer

August 2025

**Table of Contents**

Chapter 1: Background p. 7

Chapter 2: Technology Stack p. 14

Chapter 3: Goals and Objectives p. 19

**Chapter 4: Data Structures & System Workflow** p. 23

Chapter 5: Error Handling p. 00

Chapter 6: Unit Testing p. 00

Chapter 7: Deployment & Future Development p. 00

Chapter 8: Limitations p. 00

**References p. 00**

Appendix A: Development Sprints AvantZero p. 00

Appendix B: Development Roadmap Avant p. 00

**List of Terms and Abbreviations**

**Python**

A versatile, high-level, general-purpose programming language; well known for its readability and expansive toolset for machine learning, scripting, algorithms, artificial intelligence, web development, stand-alone web applications and data analysis.

**CLI**

Abbrevation standing for *‘Command Line Interface’*; a piece of software that requires text commands for command/software initialization and execution instead of a graphical interface (containing menu’s, buttons and other graphical representations)

**Algorithm**

Software containing a specific and custom-made set of rules to get data such as media, text and the like from point A (input) towards a desired point B (output).

**Machinima**

Combination of the words ‘machine’ and ‘cinema’; a popular artform for participatory media creation since the late 1990’s/early 2000’s. Literal meaning: the creation of digital films inside videogame engines (often by utilizing videogame assets, code adjustments, code execution and game modification programs)[[1]](#footnote-1)

**Avant Garde**

Popular artform that develops new experimental concepts or techniques; often associated with the arts[[2]](#footnote-2)

**Dominion Algorithm**

A random data generator for experimental Quake III machinima with limited functionality. Predecessor of the Avant algorithm. Initially created in late 2022 – early 2023 with the JavaScript language. Now considered legacy and defunct.[[3]](#footnote-3)

**Proof of Concept**

A version of specific software with limited functionality and scope to prove the feasibility of a concept or idea. AvantZero is considered a ‘proof of concept’ and only provides limited functionality compared to Avant – which is the full version of AvantZero - and is currently, as of July 2025, still in active development.

**Quake III Arena**

Videogame released in December 1999 by game-developer *ID Software[[4]](#footnote-4).* Quake III Arena is a multiplayer-based first-person shooter, in which the player is able to compete (often during fast-paced gameplay) against computer-controlled enemies (bots) or other online players in arena-based levels (better known as ‘maps’) throughout a multitude of different game modes, such as, but not limited to: team deathmatch, 1 on 1 duels and Capture the Flag.

**NLE**

Abbrevation of: Non-Linear Editor; Software suite used for video-editing.

**GUI**

Abbrevation of *‘Graphical User Interface’*; a piece of software that contains a graphics-based interface with buttons, (colorful) text, navigable menus and other forms of functionality for easy reference and execution of tasks.

**Q3MME**

Abbrevation of *‘Quake III Movie Maker’s Edition’*. A modification made for the game Quake III Arena by the users HMage, ent, auri and CaNaBiS. Q3MME allows the users to capture footage within the ID Tech 3 engine from recorded gameplay and turn them into video files or image sequences for use in machinima or game art projects. The mod contains several advanced functions such as Multi-View support (allowing to see multiple point of views at once), Depth of Field generation, custom export parameters and the implementation of motion blur.[[5]](#footnote-5)

**CRUD**

An acronym (also known as C.R.U.D.) standing for ‘Create, Read, Update, Delete’: four basic operations in computing and data management.[[6]](#footnote-6)

**API**

Abbrevation standing for ‘Application Programming Interface’: a piece of standalone software or set of rules/protocols that allow applications to communicate or exchange functionality and/or data[[7]](#footnote-7).

**OOP**

Abbevration standing for ‘Object Oriented Programming’: a programming paradigm focused on the use of classes and objects (instances of classes) instead of function-based code.

**Chapter 1 – Background**

AvantZero is a Python-based tool designed to simplify and streamline the post-production process of randomized experimental machinima films with an emphasis placed on Quake III machinima. It was initially conceived as an experimental edit-decision randomizing algorithm with very basic functionality in late 2022 – early 2023 as the ‘Dominion’ algorithm. Written in Node.js and JavaScript over the course of a few weeks and released on NPM afterwards, Dominion was quickly considered defunct and legacy and more of a prototype for newer, upcoming algorithms due to its severe lack of automation capabilities and generation of vast amounts of data, which still had to be interpreted, sorted and used by hand. It merely outputted randomized edit decision data and production-data, to which the editor should adhere to during both the capturing and editing process. At that given point in time, Dominion failed to automate anything for the user.

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 1: The bitbucket repository containing the Dominion algorithm[[8]](#footnote-8)*

The Dominion algorithm was originally developed as a both a production and post-production tool for the machinima label ‘A Pixelated Point of View’. Its main functionality revolved around randomly deciding how several experimental and randomized compositions should be captured inside of ID Software’s ID Tech 3 engine (with the Quake III MovieMaker’s Edition mod – hereafter referred to as Q3MME) and how all pieces of footage should be assembled on the NLE timeline. In a sense, it presented a missing link inside a artistic, poetic and collaborative process between humans and machines. The main line of thought behind the Dominion algorithm was that it would make final decisions regarding the visual aspect of the film, while the user remained full control and sole responsibility for the auditory aspect of the film.

At its earliest stage of usability, Dominion painfully pointed out that its level of quality was too low to be used for any useful production work. Much of the work still had to be conducted by hand, the data was not properly sorted and interpreted by the algorithm and in fact: usage of the Dominion algorithm only added more work for the user. It merely helped the user with creating a tangible copy of production and post-production data, yet the data was often spread over numerous pages and proved to be often too extensive for any form of quick reference. Each of the steps that had to be followed afterwards still had to be conducted manually; thus, sparking the idea for the creation of a better, smarter and much more polished version of the Dominion algorithm by placing emphasis on functionality, broader usability and automation, while still embracing the core idea of ‘*randomized video (post)-production’.*

It was at that moment in early 2024 that *Avant* was born: the successor to the Dominion algorithm. Avant’s main functionality revolves around the generation of random, experimental Quake III Arena compositions while automating much of the production and post-production process. It can create all the required project files for the Quake III Movie Maker’s Edition mod, a simple executable bat file to start the entire capturing process and finally generate XML/EDL files required for the NLE to properly read and export the randomized compositions to movie files.

A screenshot of a computer

AI-generated content may be incorrect.*Figure 2: A code-snippet of the Dominion algorithm from late 2022,*

*highlighting its emphasis on rapid development and its prototypical nature[[9]](#footnote-9)*

Technically speaking, Avant allows for the generation of practically unlimited variations of potential experimental Quake III machinima films with a few mouse clicks and inputs. While the algorithm’s core functionality revolves around Quake III machinima, the algorithm contains additional functionality to simply randomize existing footage captured from an unlimited number of games and/or video sources. Even footage from video cameras and phones may be used with the Avant algorithm; no matter what the user wishes to randomize at that point. If it contains a codec that is compatible with the algorithm, Avant will be able to randomize it and output the required files for use in NLE software or even visualize the output.

Avant was named after the highly popular art movement *'avant-garde'*, as it outputs raw, randomized and basically unchecked film data for the primary purpose of creating innovative, experimental and unconventional Quake III machinima films. It is not until the actual post-production process that the user is able to see how the generated data is visualized and captured. The films generated with avant focus on design, graphical representation, artistic expression and the idea of challenging traditional norms and practices found within Quake III movie production – so called *‘avant-machinina’* or *‘algorithmic machinima’*, and not necessarily on their contents nor story; thus staying in true 'avant-garde fashion.

An extensive list of additional forms of functionality for the Avant algorithm has been planned in the future, such as the addition of an API containing extensive information regarding Quake III maps, such as thumbnails, spawn points and locations, map boundaries and an auto downloader function to automatically download and extract the required levels; or the implementation of machine learning: allowing the algorithm t0 embrace different editing styles (through my own machinima work serving as training data), learning from its choices, properly utilizing them and allowing the algorithm to step away from the pseudo-randomized nature of its generated compositions if desired.

Considering the Avant algorithm will most likely take a few more years to complete because of its complex nature and broad level of functionality, it was decided to develop AvantZero: a proof of concept for the Avant algorithm. Avant remains in active development to this day and finds itself in early stages of development (as of July 2025: version 0.2.9), however, all resources will currently be allocated towards the completion of AvantZero before development on Avant will be resumed. The full Avant algorithm, in its current state, contains a few forms of functionality such as a fully tested and working main menu, a work in progress documentation website, a fully working project manager with CRUD functionality, several functions that ensure proper initialization of the algorithm, a number of general utilities, and a few helper functions for the extraction of map-related data and .bsp files from a .pk3 archive.

The AvantZero algorithm serves as a *proof of concept*: proof that the functionality that is envisioned for Avant really is able to work as intended and can be further expanded upon. While AvantZero receives less functionality, it contains most of the core pieces of functionality that Avant should contain such as randomized compositions and depth of field map generation. Furthermore, it can output and visualize all generated data in PDF and/or CSV format for data analysists or enthusiasts.

A screen shot of a computer

AI-generated content may be incorrect.

*Figure 3: Version 0.2.9 of the Avant algorithm running the first-time setup*

*inside of the command line[[10]](#footnote-10)*

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 4: Version 0.2.9 of the Avant algorithm has successfully*

*completed its first-time setup[[11]](#footnote-11)*

Once development for AvantZero has been completed, its proof of concept serves a few purposes. First, AvantZero serves as a final submission for CS50X: Harvard’s Introduction to Computer Science and Programming course; thus marking the completion of said course. Secondly, AvantZero will be used for the generation of a few works of ‘avant-machinima’ or ‘algorithmic-machinima’, which serve for both artistic exploration and submission to festivals, galleries, museums and other exhibitions.

Considering a personal background as a machinima artist and software developer, AvantZero allows for a highly interesting connection between cinema, code, art and machinima. Lastly, all functionality that has been successfully implemented within AvantZero will be added in the full version of Avant and further expanded upon.

This deep-layered and ever-evolving development approach (Dominion – AvantZero – Avant) illustrates a trajectory of design, technological refinement and deepening of concepts. Each stage serves a purpose in the vision to merge algorithmic processes with create co-authorship in the context of machinima. While limited, Dominion exposed the core functionality but also the core bottlenecks of early automation attempts. It laid the foundation for a more extensible system. Avant, while still in early development, reflects an ambitious take at creating a modular, intelligent, ever-learning and fully integrated piece of software that aligns with efficiency and artistic unpredictability. AvantZero, in this larger context, operates as a technical manifest. It allows for entry into assisted forms of algorithmic machinima and ‘avant-machinima’, opening possibilities for students, researchers and artists to utilize and experiment with both algorithms.

The project, while rooted inside of Quake III Arena, introduces an interesting topic of conversation revolving around how experimental filmmaking can evolve through co-creation, partial automation, machine-based assistance and randomized activity. In this model, the algorithm is not merely a tool but a collaborator as the algorithm outputs decisions concerning the visual aspect of the film, while the filmmaker will remain in control of the auditory scope. Even so, the filmmaker is and will always be able to make any kind of adjustments to the generated sequences: thus, staying in full control of their creative process while always having the possibility to opt in new, exciting forms of hybrid filmmaking thanks to the highly modular philosophy of both the AvantZero and Avant algorithms.

A screenshot of a computer program

AI-generated content may be incorrect.

*Figure 5: Main menu of the Avant algorithm (Version 0.2.9)[[12]](#footnote-12)*

**Chapter 2 – Technology Stack**

Both the AvantZero and Avant algorithms are entirely developed with Python and utilize several internal and external (PyPi-based) libraries. The choice for Python was motivated through a combination of practical, technical and educational reasons. First, Python’s combination of clear syntax and object-oriented programming facilitates speedy development while maintaining clarity and modularity across the codebase. This is crucial given the layered and class-based structure of AvantZero, which spans across multiple files and dynamically linked components.

Secondly, Python offers an extensive ecosystem of ready-to-use libraries for data analysis, visualization, depth estimation and further means of automation; resources that significantly contribute to the development and refinement of AvantZero’s algorithmic components and workflows. A few great examples of ready-to-use libraries are: *matplotlib* (for visualizing and plotting data), *reportlab* (for generating PDF files containing text and/or images) and *DepthAnything* (Scientific Python-based Depth of Field estimation model that works with both image and video).

Equally essential to the project was the implementation of a Graphical User Interface. Although frameworks such as Microsoft .NET MAUI (using the C# programming language) or Electron (JavaScript) are popular choices for cross-platform GUI-based development, Python’s native tkinter library proved to be a pragmatic and effective solution. Its simplicity and direct integration with the rest of the codebase and ease of customization made it ideal for any of the algorithmic requirements. Importantly, tkinter allowed for Python to be maintained as the single language across all layers of the project; from logic to interface, reducing overhead and improving maintainability. Packaging tools such as PyInstaller were used to convert the project into standalone executables for Windows and MacOS operating systems, making deployment seamless and providing users with direct access to the algorithm with little to none setup required.

Lastly, the choice of Python as the sole programming language for the AvantZero algorithm was also personally motivated due to broad familiarity with its ecosystem, syntax, packages and was also pedagogically motivated because of a personal educational background as a Web Development and Python lecturer. By creating the AvantZero algorithm solely within Python, it could serve an additional purpose as a dual-point of inspiration and (case-study) reference for students over the course of the two marking periods during which they would be following Python lectures. Many of the fundamental aspects found within AvantZero, such as the use of functions, loops, objects, classes, internal libraries (such as *os[[13]](#footnote-13), sys*[[14]](#footnote-14)and *random*[[15]](#footnote-15)) and external libraries (such as *matplotlib*[[16]](#footnote-16), *requests*[[17]](#footnote-17)and *fastAPI*[[18]](#footnote-18)) are core programming concepts found within the Python lectures: allowing for practical examples of how said concepts and libraries could be integrated into real-life software projects.

Future functional additions centering around machine-learning and artificial intelligence during the later stage of Avant’s development cycle with libraries such as *TensorFlow*[[19]](#footnote-19)and *PyTorch*[[20]](#footnote-20)would also benefit greatly from use with the Python programming language as Python is a dominant language within the *machine-learning* (ML) and *Artificial Intelligence* (AI) domain; making the language an ideal choice for such future expansions. The ability to pretty much natively integrate ML models into an existing Python codebase without major restructuring ensures that future releases of Avant can continue to expand while remaining technologically coherent.

Lastly, we’ll briefly touch upon some of the internal and external Python libraries used during the development cycle and further explain their place within the software’s hierarchy. Considering that a development cycle consists of many twists and turns both from a functional and library-based perspective, only the most important libraries that are sure to be embedded within the AvantZero algorithm will be further introduced.

We’ll briefly touch upon the following internal- and external Python libraries:

* OS
* Sys
* DepthAnything
* Tkinter

OS

The built-in Python library *OS* is mandatory for AvantZero’s development cycle. It allows us to communicate with the operating system through several ways. For instance, we’re able to list all the files and folders within a certain directory, create the AvantZero folder structure, automatically create the proper folder structure for any of the projects made with the algorithm and save, adjust or delete any of the created files. Thanks to the OS library, any form of communication between Python, the operating system and its files or folders becomes a streamlined process.

Sys

Another built-in Python library which contains a few features that allow us to communicate with the operating system from a more technical point of view. However, it must be noted that sys and os are not the same. The *sys* library features functionality with which we can directly terminate the application and free the allocated memory, start the application with additional arguments from the command line, and present more technical information regarding the system AvantZero is being executed on, such as hashes, hex codes and information regarding the operating system we’ve used to run the algorithm.

DepthAnythingV2

A monocular depth estimation model written with Python and PyTorch, using machine-learning to estimate depth of field inside images and video’s. Recently, a new groundbreaking model has been released called *VideoDepthAnything[[21]](#footnote-21),* which provides the user with consistent depth of field map generation for longer types of video content (5 minutes or longer).

The model has been extensively documented by several means: through academic writing[[22]](#footnote-22), GitHub[[23]](#footnote-23) and its official website[[24]](#footnote-24). Furthermore, the developers have provided a demo version of DepthAnythingV2 on HuggingFace[[25]](#footnote-25); allowing users to experiment with the model directly.

A person with glasses and a neck brace

AI-generated content may be incorrect.

*Figure 6: Picture of the author presented within a heat-map style depth-of-field map.*

*Captured with the Depth Anything V2 demo found on Hugging Face[[26]](#footnote-26)*

Tkinter

Tkinter is a built-in Python library used to create Graphical User Interfaces (GUIs) for applications. Since Python is often executed and interacted with through the command line, Tkinter enables developers to build engaging and clear visual interfaces for software programs. The library offers numerous tools for creating splash screens, buttons, selection menus, loading bars, and customizing the design of these elements—allowing AvantZero to stand out with a polished user experience.

Because everything runs within the Python ecosystem, using Tkinter allows the application to scale gracefully without relying on external interfaces or additional libraries.

**Chapter 3 – Goals and Objectives**

The main goal for the AvantZero algorithm is to fully automate various tasks found within Quake III machinima (post)production pipelines; tasks that would otherwise need to be conducted manually. Keeping the algorithm’s artistic and technical origins in mind, it aims to achieve the following creative and technical objectives:

Generation of unique, pseudo-randomized and experimental video sequences

AvantZero enables users to generate pseudo-randomized experimental video sequences by reorganizing pre-existing footage in an algorithmic way that defies repetition. Each composition generated by the algorithm is to most extent unique, even when using the same source material. This process of automatically generating new content reflects the algorithms underlying commitment to avant-garde principles, such as spontaneity, variation and non-linearity instead of embracing traditional and narrative-based ways of filmmaking.

Rather than following predetermined editing patterns, AvantZero outputs randomized post-production data (in formats such as XML and EDL) that can be imported directly into a non-linear editing (NLE) software suite. The algorithm acts as a generative and creative partner, proposing unexpected visuals and rhythms that challenge conventional filmmaking logic.

The algorithm aligns with experimental filmmaking traditions that aim for abstraction and viewer interpretation. By automating the visual structure through the algorithm and leaving the auditory domain to the user (such as sound design, audioscaping, music and/or dialogue), AvantZero introduces a highly interesting form of collaboration between human and code. While this approach of algorithmic machinima-making may seem closed-circuit in a sense, it must be noted that the filmmaker will always remain in control of their creative choices.

The EDL and XML files are not locked in any shape or form: allowing the user to change, improve upon or revert changes conducted by the algorithm. Any output generated by AvantZero may be fully accepted by the user, rejected, or used as a starting point for further improvements. Therefore, avant-machinima, or algorithmic-machinima is an exciting and continuously evolving body of work where no two outputs are identical.

Automating the assembly process of diverse montages

In virtually every filmmaking workflow, the act of assembling and trimming raw footage remains a foundational process. Some might find it a repetitive process. This is no different in machinima. Machinima artists are often tasked with selecting, sequencing and refining a vast array of gameplay clips, and sit through large quantities of video frames. To address the elephant in the room: AvantZero is not seeking to replace the manual editing process, neither will it propose or promote automation as a substitute for artistic expression. On the contrary, the manual curation of footage should be valued and preserved as rhythms, contrasts and visual leitmotivs appear organically throughout the process. It is within this very hands-on manipulation that a film’s unique sense of style begins to take shape.

Rather than suppressing this process, AvantZero serves as a complementary pillar. It is a system of visual experimentation designed to challenge traditional conventions by primarily emphasizing focus on experimentation and randomized footage selection. It embraces the idea of randomizing its footage selection, and adoption of uncertainty and unpredictability. It avoids the familiar pacing found in traditional cinematic convention. So much more so of uncertainty and unpredictability is found in the full version of AvantZero (the Avant algorithm), in which the user is only able to witness their final result once they import the XML/EDL file into the NLE for the first time and load up the generated sequence.

AvantZero introduces a new audiovisual syntax that privileges experimentation over story; variation over repetition. It supports an exploratory mindset, where the algorithm serves as a collaborative tool inside of the editing suite. The user can adjust the proposed edits to their liking at any given time, or, if desired, immediately accept the presented composition. The algorithm serves as a provocative starting point for further refinement, disruption, distortion (by utilizing the Practice of Distortion Framework[[27]](#footnote-27)), reinterpretation and cinematic expression. AvantZero both embraces and restructures the creative loop between intuition, interpretation, tradition and experimentation.

Broadening the roles of the User (Post-Editor & Interpreter)

By default, during most film post-production processes, the user acts as the author. While using the AvantZero algorithm, the user gains an additional role: that of *interpreter*. The editorial agency is initially shifted from the user towards the algorithm, since it is the algorithm that presents the user with a preliminary and randomized sequence. While doing so, the user’s ability to intervene, reject and/or reinterpret remains preserved. By shifting the roles, the process of editing transforms into a form of dialogue between human and machine; between suggestion, interpretation and creative desire.

Multi-Format Compatibility and Integrated Depthmap Generation

AvantZero was designed with flexibility in mind. Thanks to a multitude of external software libraries implemented within the algorithm, such as the DepthAnythingV2 model, it accepts both image sequences and traditional video formats, such as, but not limited to: JPG, PNG, TIFF, DPX and H264, Apple ProRes, MXF, HEVC or AVI files. The algorithms wide range of compatibility ensures that artists can work across a broader spectrum of visual material.

While the codecs above exemplify several codecs that will enjoy support from the first initial version, it remains a possibility that initial support will be extended with additional codecs during later stages of development; considering that minor updates will still be considered during AvantZero’s significantly shorter development cycle (due to its proof of concept status).

AvantZero’s multi-format file compatibility allows for deeper integration into diverse post-production pipelines and, in tandem with its XML and EDL generation capabilities, for direct compatibility with several NLE’s as well; such as Adobe Premiere Pro, Blackmagic Da Vinci Resolve and/or Avid Media Composer.

In addition to its compositional functionality, the algorithm contains the capability to generate frame-by-frame depth of field image-streams alongside (or in a standalone type of fashion) its randomized video output thanks to the groundbreaking *DepthAnythingV2* model. These depth maps provide valuable metadata that can be used in a variety of technical workflows (such as 3D compositing, generating manual depth of field masks) or for further experimental and artistic intent.[[28]](#footnote-28)

Organized Data and Clear Visualizations in a multi-format fashion

While the core focus of AvantZero revolves around generating randomized experimental data and video sequences, the software is also equipped with tools for organizing and outputting generated and randomized data in a structured and accessible format. Generated datasets are automatically compiled into clear PDF reports, and exported as CSV files for further processing or analysis in conventional environments such as vanilla Python, conda or Jupyter Notebooks.

In addition, AvantZero supports the automatic creation of visualizations: such as line graphs, bar charts, and scatter plots; using libraries like *matplotlib*. Generated plots are exported both as part of the PDF reports and as individual image files (e.g., PNG), allowing for use in presentations, publications, or external dashboards.

All organizational and visualization functions are tightly bound to each individual project. This means they can always be recalled, regenerated, or updated at any stage of the workflow, ensuring consistency throughout the experimental process.

**Chapter 4: Data Structures & System Workflows**

AvantZero will utilize the Object-Oriented Programming paradigm during its development cycle, which lays emphasis on the use of classes and instances (named objects) to create extensive and complex relationships between the different cores found within the software. In AvantZero’s design, each larger encompassing aspect of the software will be considered a ‘*core’*, or in OOP-terms: a class.

Relationships between cores are visualized with the aid of UML (Unified Modeling Language): a set of standardized diagrams used to depict the structure and interactions within object-oriented systems. Several times throughout this chapter, we’re referring to what is called a *sequence diagram.* A sequence diagram is a visual representation of the relationships and connections between objects and how they interact with each other over time.

In the case of the AvantZero algorithm, the diagram depicts all cores found within the software, how each of these parts consecutively interact with one another and visualizes the data exchanged between each of the various cores to reach a specific type of output. Considering each of the core’s interconnect with one another several given times during the data generation process, the diagram required some dilution to visualize the phenomena and processes at play in the most clear and concise way possible. After introducing the diagram, all cores will be briefly further elaborated upon, with the chapter’s conclusion centring around a brief and concise explanation of the processes that are at play within the proposed sequence diagram.

The diagram is divided into two distinct areas, referred to as *‘Boot’* and *‘Main’*. All functionalities found within the boot section of the diagram is linked to the initialization and booting process of the algorithm. The main section centres around all other form of functionality, such as the project manager (PROJM), data analysis generator (DANLY), depth generator (DEPGEN) and the main system (SYS); which concerns itself with all higher-level functionality and connectivity between all cores found within the algorithm.

**A diagram with blue lines

AI-generated content may be incorrect.**

*Figure 7: Partial representation of the AvantZero Sequence Diagram: visualizing the*

*initial checks conducted by the project manager after boot[[29]](#footnote-29)*

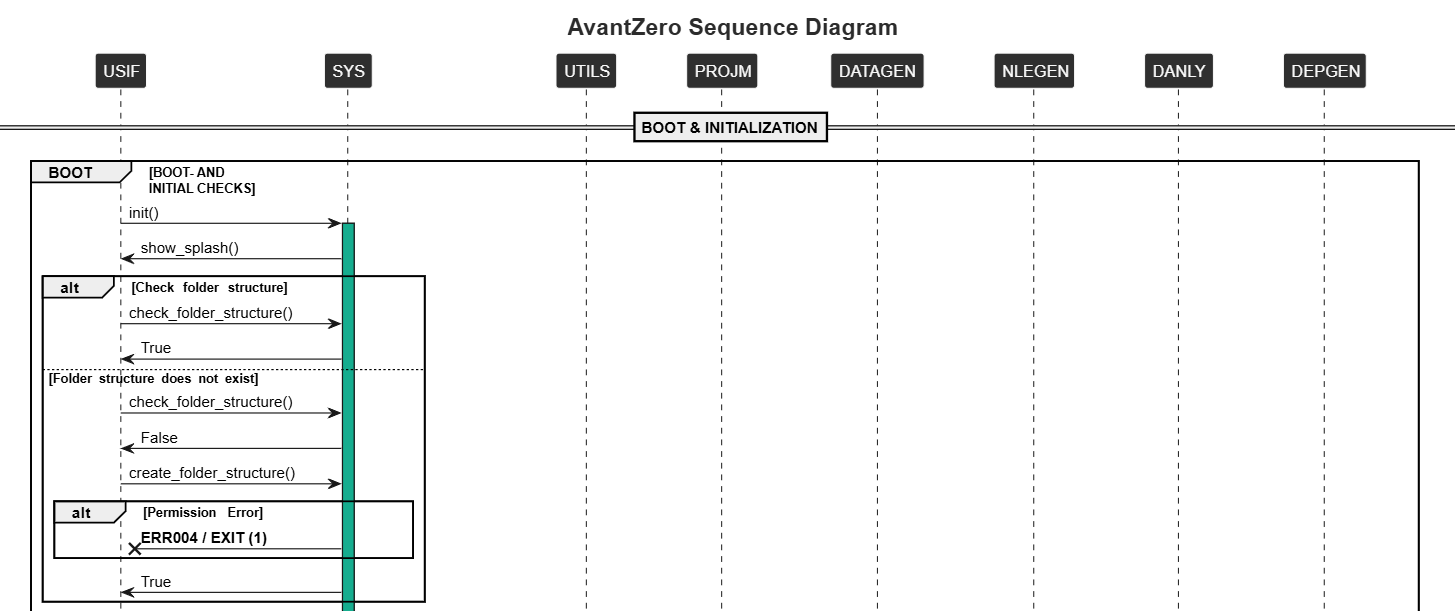
Considering the large size of the diagram, each of the different sections found within the diagram will be represented, introduced and elaborated upon individually: keeping digestibility and conciseness in mind. For the full diagram, please refer to the AvantZero GitHub Repository.[[30]](#footnote-30) The source-code for the sequence diagram, and visualizations of the entire diagram in various formats (SVG, PNG and PDF can be found within the *docs* folder.

The diagram was created inside Visual Studio Code with the aid of the *PlantUML*[[31]](#footnote-31) library in a bit over four-hundred lines of code. PlantUML is a Java-based tool allowing for the streamlined creation of various UML diagrams, such as sequence diagrams and activity diagrams.

Together with the PlantUML extension found within Visual Studio Code, users can create clear, neat and customized diagrams with code-based components inside .*puml* files, easily visualize and/or export them to tangible files for later use.

**AREA 1 - BOOT**

**1.1 – Initial Boot**

****

*Figure 8: AvantZero Sequence Diagram: Visual representation of the initial boot*

*functions[[32]](#footnote-32)*

Figure 8 above contains a partial representation of the first distinct area found within the AvantZero algorithm; called *BOOT.* The functionality described mainly revolves around functionality concerning the boot and initialization process of the algorithm. A few examples include creating the splash screen as a form of visual indicator that the algorithm is booting, conducting initial checks of all required Python files, verifying if the AvantZero folder structure exists and has remained in place since the last boot, and whether all required Python libraries have been successfully installed and rightfully called upon. It is not until the working of all vital components has been successfully verified that initial control is given back to the user and the main menu is presented.

[Afterwards, continue with dependency and package checks]

While AvantZero is considered a niche algorithm by trade and primarily serves as a proof of concept for a broader and updated version of the algorithm (named Avant) with much more added functionality; AvantZero, as a prototype, automates several tasks and generates a multitude of data streams that may be of interest for a number of user-bases, such as but not limited to:

* Machinima Artists willing to conduct experiments with Depth of Field maps within their workflow or create machinima solely based on Depth of Field maps;
* Machinima Artists wanting to generate randomized and experimental compositions of their project for either final use or demonstrational purposes;
* Video-Editors wanting to generate randomized compositions of existing footage (of any kind) for either final use or demonstrational purposes;
* Machinima Artists wanting to combine original footage and depth of field streams altogether in an experimental manner to create highly artistic point of views;
* Video-Editors and Machinima Artists in need of generating storyboards or timeline reports in PDF format of their projects;
* Machinima Artists, Programmers and/or Data Analysts who would like to view, analyze and visualize the generated randomized data (in CSV format).

The broad range of potential users for the AvantZero algorithm reflects the hybrid nature of the algorithm itself. It’s embedded equally primarily focuses between artistic exploration and technical automation, while offering utility among a multitude of disciplines. It’s core functionality lies in experimental machinima production, yet it offers additional capacity in the form of randomizing compositions, depth map generation and handling structured data. Because of its wide applicability, AvantZero is perfectly usable for both creative and analytical workflows.

Machinima artists now can utilize Depth of Field streams throughout several ways, and more importantly: throughout an setup-based automated process. For instance, the artist can export ‘depth maps’ alongside their footage for later use within their post-production pipeline, such as for the addition of manual depth of field, color grading effects or focus to a shot as demonstrated in earlier video demonstrations.[[33]](#footnote-33) Secondly, one could use the ‘depth map stream’ as primary footage to create exciting and experimental point of views. Lastly, the depth of field streams could provide valuable data for users interested in analyzing or visualizing the random generated data, or simply provide a low threshold to dive into experimental depth of field map use for the very first time.

The inclusion of automated PDF- and CSV report further broadens its appeal to users seeking documentation, visualization or post-production analysis of their projects and allows for randomized brainstorming or rapid prototyping. Though AvantZero is not intended to function as a final product, the algorithm has emerged as a versatile and exploratory production tool; one that merges creativity with automation and aesthetic control.

Looking forward, the upcoming Avant algorithm (the fully completed version of AvantZero) will significantly expand upon this foundation. Unlike AvantZero, Avant is built with full-scale film-data integration in mind, allowing it to actively analyze patterns through machine learning and artificial intelligence, generate corresponding depth maps and even autonomously capture machinima footage.

It includes native support for the game Quake III Arena through the automatic generation of full Q3MME (Quake III Movie Maker’s Edition – a mod to capture video or images within the id Tech 3 engine) project files, including scripts, executables and project files; further bridging the gap between procedural data and hands-on filmmaking. AvantZero not only serves as a technical prototype, but also as a conceptual framework and steppingstone towards a more intelligent and deeply integrated machinima production platform.

**Chapter 4 – Functional Requirements**

In a nutshell, AvantZero should offer the following functionality upon its initial (version 1.0) release:

* Provide the user with an easy-to-navigate main menu;
* Embrace a project-based workflow. Users can create, edit and delete projects. The user can generate random compositions, depth of field streams or generated data on a project-basis and is able to choose whether they would like to execute a single area of functionality or combine freely between multiple functions at will;
* Contain a clear overview of potential errors and provide simple, jargon-free potential fixes for the user if an error were to be thrown by the algorithm;
* Provide a consistent, qualitative, easy-to-read and quick-to-start documentation website for both first-timers and experienced users with Python when they decide to use the AvantZero algorithm;
* Allow the user to create a single or multiple experimental edits of footage of any kind, with an emphasis on machinima. The user should be able to utilize both image sequences and video files alike and alongside one another;
* Allow the user to create depth of field map streams:
  + Create depth of field streams of the users’ entire footage bin;
  + Create depth of field streams of the randomized sequences only;
* Provide the user with both XML and EDL files for use in popular NLE’s, such as Adobe Premiere Pro, Blackmagic DaVinci Resolve or Avid Media Composer – to name a few;
* Generate a PDF containing table-based data on all randomized choices made by the AvantZero algorithm, including timecodes, filenames and thumbnails (somewhat in comparison of a watered-down dailies report);
* Generate a CSV file containing data on all randomized choices made by the AvantZero algorithm, ready for use with data analysis frameworks such as pandas, vanilla Python or Jupyter Notebook;
* Add visualized charts of said data into the table-based data PDF upon request;
* Offer a submenu that allows the user to select their function of choice after selecting a project;
* Offer a submenu with all listed dependencies, a quick how-to, links to the documentation website and further copyright/license information;
* Multiple ways to run the application, such as: with Python commands, compiled executables or as a package on the Python Package Index ‘PyPi’;
* And most importantly: a way for the user to gracefully terminate and exit out of the application correctly;

**Chapter 5 – Limitations**

Within its specified scope of functionality, AvantZero currently knows the following limitations:

* AvantZero can only create randomized compositions from existing footage to present proof of concept of the current ‘RNG’-based workflow, which encompasses the heart of both the Avant and AvantZero algorithm. AvantZero cannot capture randomized footage; functionality that Avant will have with the implementation of the Quake III Movie Maker’s Edition mod.
* The Avant algorithm will eventually have a deeper, more sophisticated menu structure than the AvantZero algorithm.
* AvantZero will most likely not receive any further updates after its initial version 1.0 release, except for small numbers of patches to ensure full functionality. Avant, on the other hand, will receive long-term minor and major updates.
* Avant will contain broader means of functionality, such as, but not limited to: artificial intelligence, machine learning, an full-fledged external API (current codename: TRIII – which is pronounced as ‘tree’) containing extensive information of each existing Quake III map (relieving the algorithm from these repetitive duties and thus speeding up the overall production process) such as spawn points, screenshots, general information and an auto map-downloader function.
* AvantZero should be seen as a ‘console-first’ application as it is developed first with the terminal in mind. A minimalistic *Graphical User Interface* will be implemented during later stages of development for easy reference and initialization. The full version of AvantZero – the Avant algorithm – will contain two separate and dedicated modes for the user to choose between either CLI or GUI-mode upon boot; allowing them to swap modes on the fly.

**References**

Altar of Gaming (n.d.). *Quake III Arena*. Altar of Gaming*.* Retrieved July 10, 2025,

from <https://altarofgaming.com/game/quake-iii-arena/>

Merriam-Webster. (n.d.). *Avant-garde*. In *Merriam-Webster.com dictionary*.

Retrieved July 9, 2025, from <https://www.merriam-webster.com/dictionary/avant-garde>

Oxford Dictionary (n.d.). *Machinima*. In *Oxford English Dictionary*. Retrieved July

15, 2025, from <https://www.oed.com/dictionary/machinima_n?tl=true>

Veenstra, J.P. (2020). DEPTHMAP NEXT [Video]. YouTube.

<https://www.youtube.com/watch?v=c2FT_a34vw4>

Veenstra, J.P. (2022). Quake III Arena: Experimental Depth Map use [Video].

YouTube. <https://www.youtube.com/watch?v=7cu76qMIWIQ>

Veenstra, J.P. (2024). Dominion Algorithm. Bitbucket. Retrieved July 16, 2025, from

<https://bitbucket.org/appov/dominion/src/main/>

**Appendix A: Development Roadmap**

The table below outlines the development roadmap for the AvantZero algorithm in a concise and summarized manner. Please refer to the *Technical Design Report* for a more detailed description.

|  |  |  |
| --- | --- | --- |
| Version | Functionality | Summary |
| 0.0.0 | Documentation | * Functional Design * Technical Design and software diagrams * Design mock-up for AvantZero docs * Development Roadmap |
| 0.1.0 | Preparation | * File and folder structures ready * Documentation Website Loading Screen and Homepage layout ready (accessible and responsive) * Implementation of ‘Internationalization’ on the documentation website – support for both Dutch and English language * Outline of algorithm functions and helper functions defined in software |
| 0.2.0 | Splash Screen / Main Menu | * Completed Loading Screen (with helper functions to ensure all required files and packages are installed) * Completed Main Menu, all possible selections are working and lead to their respective functions * All functions can return to the main menu. * The user is able to gracefully exit out of the algorithm * If applicable, errors are properly documented and jargon-free fixes are described for the user * The homepage of the Documentation website has clickable buttons leading to (for now) empty pages, is responsive, accessible and can be accessed in two languages * The documentation website has a fully functioning loading screen |
| 0.2.1 | Splash Screen / Main Menu (GUI) | * A GUI is created and designed for use with the loading screen and the main menu. * All functions and helper functions make proper use of the GUI |
| 0.2.2 | Splash Screen / Main Menu (UnitTest) | * Manual and automated unit tests of the main menu, loading screen and (helper)functions * Error Database |
| 0.3.0 | Documentation: Background, Requirements, Installation | * Completed the Background, Requirements and Installation chapters of the Documentation website * Content of said chapters is clear, free of errors. * Pages are accessible, responsive and are accessible in Dutch and English. * User can switch languages at will |
| 0.4.0 | Project based workflow | * Users can see, create, edit, select and delete projects * AvantZero automatically creates the right folder structure upon creation of a project |
| 0.4.1 | Projects (GUI) | * GUI implementation of projects menu |
| 0.4.2 | Projects (UnitTest) | * Automated and manual tests of the projects menu * Update of Error Database |
| 0.5.0 | Basic Data Generation Functionality | * Proper implementation of data generation functions and helper functions * User should be able input video and receive a randomized .edl and .xml file |
| 0.5.1 | Data Generation  (Unit Test) | * Automated and manual tests of Data Generation functions * Update of Error Database |
| 0.5.2 | Data Generation (GUI) | * GUI implementation of data generation functions with visual feedback |
| 0.6.0 | Image Sequence Support | * Support for a wide array of image sequences inside the generator functions |
| 0.6.1 | Image Sequence  (Unit Test) | * Automated and manual tests ensuring compatibility with image sequences * Update of Error Database |
| 0.6.2 | Image Sequence  (GUI) | * Full GUI support for both video- and image-based data generation |
| 0.7.0 | PDF Support | * The application is simultaneously able, at a base level, to generate randomized data and output it to a PDF file |
| 0.7.1 | PDF Support  (Unit Test) | * Automated and manual tests of exporting project data to a PDF file * Update of Error Database |
| 0.7.2 | PDF Support (GUI) | * Full GUI support for PDF generation |
| 0.8.0 | CSV Support | * The application is simultaneously able, at a base level, to generate randomized data and output it to PDF and/or CSV files through separate, dedicated functions |
| 0.8.1 | CSV Support  (Unit Test) | * Automated and manual tests of CSV related functions * Update of Error database |
| 0.8.2 | CSV Support  (GUI) | * GUI implementation of CSV related functionality |
| 0.9.0 | Data Visualization | * Implementation of generating plots and saving these as files within the project folder |
| 0.9.1 | Data Visualization  (Unit Test) | * Automated and manual testing of Data Visualization function * Update of Error Database |
| 0.9.2 | Data Visualization  (GUI) | * Implementation of Data Visualization inside of the GUI |
| 0.10.0 | Depth of Field Base Support | * Basic support for generating Depth of Field maps |
| 0.10.1 | Depth of Field Base Support  (Unit Test) | * Automated and manual tests for Depth of Field generation |
| 0.10.2 | Depth of Field Base Support  (GUI) | * GUI implementation of Depth of Field Support |
| 0.11.0 | Advanced Depth of Field Support | * Base support has been implemented. At this stage, the user should be able to generate the depth of field maps from the same, randomly generated pieces of footage, or allow the algorithm to generate depth of field maps for the entire set of footage |
| 0.11.1 | Advanced Depth of Field Support  (Unit Test) | * Automated and Manual tests * Update Error Database |
| 0.11.2 | Advanced Depth of Field Support  (GUI) | * GUI implementation of all Depth of Field related functionality |
| 0.12.0 | Error Database | * The Error Database has been frequently updated and contains extensive and easy-to-execute information on how to solve common problems * The Error Database is implemented on the Documentation website |
| 0.13.0 | License, Credits Pages | * The License page on the Documentation website has been fully completed * The Credits page on the Documentation website has been fully completed |
| 0.14.0 | Quick Start | * The Quick Start page contains concise and clear information on how to easily setup a project and start generating data * The Quick Start page is accessible, responsive and readable in two different languages |
| 0.15.0 | Project Manager | * The project manager page contains extensive information regarding the project manager and all its functionality * The project manager page is accessible, responsive and readable in two different languages |
| 0.16.0 | Data Generation | * The data generation page contains extensive information regarding data generation and all its functionality * The data generation page is accessible, responsive and readable in two different languages |
| 0.17.0 | Data Visualization | * The data visualization page contains extensive information regarding data visualization and all its functionality * The data visualization page is accessible, responsive and readable in two different languages |
| 0.18.0 | Depth of Field | * The depth of field pages contain extensive information regarding depth of field and all its functionality * The page is accessible, responsive and readable in two different languages |
| 0.19.0a | Alpha | * Checks and final adjustments of documentation website * Final tests and final adjustments of algorithm code |
| 0.20.0b | Beta | * Creation of TestPyPi package and installation * QA testing by a body of external individuals * Final code and documentation adjustments |
| **1.0.0** | **Initial Release** | * Completed code and documentation website * Updated GitHub repository * Uploaded all required CS50X deliverables, such as code, readme, presentation and form * Not unimportant: received CS50x certificate * Created Windows and MacOS versions of AvantZero * Released AvantZero on PyPi * Project completed |

AvantZero might receive a few *patches* and *minor* *updates* after its initial release to ensure full functionality and compatibility for its intended purpose. However, additional functionality will not be guaranteed as development will eventually shift towards the completion of the Avant algorithm. The repository will be publicly archived on GitHub once development on AvantZero has been considered complete.

1. Oxford Dictionary (n.d.). *Machinima*. In *Oxford English Dictionary*. Retrieved July 16, 2025, from <https://www.oed.com/dictionary/machinima_n?tl=true> [↑](#footnote-ref-1)
2. Merriam-Webster. (n.d.). *Avant-garde*. In *Merriam-Webster.com dictionary*. Retrieved July 17, 2025, from <https://www.merriam-webster.com/dictionary/avant-garde> [↑](#footnote-ref-2)
3. Veenstra, J.P. (2024). Dominion Algorithm [Bitbucket Repository]. *Bitbucket*. Retrieved July 17, 2025, from: <https://bitbucket.org/appov/dominion/src/main/> [↑](#footnote-ref-3)
4. Altar of Gaming (n.d.). *Quake III Arena*. Altar of Gaming*.* Retrieved July 16, 2025, from <https://altarofgaming.com/game/quake-iii-arena/> [↑](#footnote-ref-4)
5. entdark. (n.d.). *q3mme* [GitHub repository]. GitHub. Retrieved July 16, 2025, from: <https://github.com/entdark/q3mme> [↑](#footnote-ref-5)
6. Codecademy (n.d.). *What is CRUD?* Codecademy. Retrieved July 16, 2025, from: <https://www.codecademy.com/article/what-is-crud-explained> [↑](#footnote-ref-6)
7. Goodwin, M. (2024). *What is an API (Application Programming Interface?* IBM. Retrieved July 18, 2025, from: <https://www.ibm.com/think/topics/api> [↑](#footnote-ref-7)
8. Veenstra, J.P. (2025). *Dominion Algorithm Repository* [Image]. [↑](#footnote-ref-8)
9. Veenstra, J.P. (2025). *Dominion Algorithm: Code Snippet* [Image]. [↑](#footnote-ref-9)
10. Veenstra, J.P. (2025). *Avant 0.2.9 running first-time setup* [Image]. [↑](#footnote-ref-10)
11. Veenstra, J.P. (2025). *Avant 0.2.9 successfully completed first-time setup* [Image]. [↑](#footnote-ref-11)
12. Veenstra, J.P. (2025). *Main menu of the Avant algorithm. Version 0.2.9* [Image]. [↑](#footnote-ref-12)
13. Python Software Foundation. (n.d.). *os — Miscellaneous operating system interfaces*. Python 3. Retrieved July 19, 2025, from <https://docs.python.org/3/library/os.html> [↑](#footnote-ref-13)
14. Python Software Foundation. (n.d.). *sys – System-specific parameters and functions.* Python 3. Retrieved July 19, 2025, from <https://docs.python.org/3/library/sys.html> [↑](#footnote-ref-14)
15. Python Software Foundation. (n.d.). *random – Generate pseudo-random numbers.* Python 3. Retrieved July 19, 2025, from <https://docs.python.org/3/library/random.html> [↑](#footnote-ref-15)
16. Matplotlib Development Team (n.d.). *Matplotlib – Visualization with Python.* Matplotlib. Retrieved July 19, 2025, from <https://matplotlib.org/> [↑](#footnote-ref-16)
17. Python Software Foundation. (n.d.). *Requests: HTTP for Humans.* Python 3. Retrieved July 19, 2025, from <https://requests.readthedocs.io/en/latest/> [↑](#footnote-ref-17)
18. Ramirez, S. (n.d.). *FastAPI*. FastAPI. Retrieved July 19, 2025, from <https://fastapi.tiangolo.com/> [↑](#footnote-ref-18)
19. Tensorflow (n.d.). *Tensorflow*. Tensorflow. Retrieved July 19, 2025, from <https://www.tensorflow.org/> [↑](#footnote-ref-19)
20. PyTorch Foundation (n.d.). *PyTorch*. PyTorch. Retrieved July 19, 2025, from <https://pytorch.org/> [↑](#footnote-ref-20)
21. DepthAnything (n.d*.) Video Depth Anything: Consistent Depth Estimation for Super-Long Video’s.* GitHub Pages. Retrieved July 22, 2025, from <https://videodepthanything.github.io/> [↑](#footnote-ref-21)
22. Yang, L., Kang, B., Huang, Z., Zhao, Z., Xu, X., Feng, J., & Zhao, H. (2024).Depth Anything V2. In *Proceedings of NeurIPS 2024*. arXiv. Retrieved July 22, 2025, from <https://arxiv.org/abs/2406.09414> [↑](#footnote-ref-22)
23. DepthAnything (2025). *Depth-Anything-V2.* [GitHub Repository]. GitHub. Retrieved July 22, 2025, from <https://github.com/DepthAnything/Depth-Anything-V2> [↑](#footnote-ref-23)
24. DepthAnything (n.d.). *Depth Anything V2 project page.* GitHub Pages. Retrieved July 22, 2025, from <https://depth-anything-v2.github.io/> [↑](#footnote-ref-24)
25. Hugging Face (n.d.). *Depth Anything V2 Demo*. Hugging Face. Retrieved July 22, 2025, from <https://huggingface.co/spaces/depth-anything/Depth-Anything-V2> [↑](#footnote-ref-25)
26. Veenstra, J.P. (2025). *Using Depth Anything V2 on Hugging Face* [Image]. [↑](#footnote-ref-26)
27. See: Bittanti, M., & Veenstra, J.P. (in press). *Los Santos Plays Itself*. Mimesis International. [↑](#footnote-ref-27)
28. Veenstra J.P. (2023). Quake III Arena: Experimental Depth Map Use [Video]. YouTube. Retrieved July 22, 2025, from <https://www.youtube.com/watch?v=7cu76qMIWIQ> [↑](#footnote-ref-28)
29. Veenstra, J.P. (2025). *Partial Representation of AvantZero Sequence Diagram* [Image]. [↑](#footnote-ref-29)
30. Veenstra, J.P. (2025). *AvantZero*. [GitHub Repository]. GitHub. Retrieved July 28, 2025, from <https://github.com/jiyorude/avantzero> [↑](#footnote-ref-30)
31. PlantUML (n.d.). *Quick Start Guide.* PlantUML. Retrieved July 28, 2025, from <https://plantuml.com/starting> [↑](#footnote-ref-31)
32. Veenstra, J.P. (2025). *AvantZero Sequence Diagram: Visual Representation of Boot & Dependency Check* [Image]. [↑](#footnote-ref-32)
33. Veenstra, J.P. (2022). Quake III Arena: Experimental Depth Map use [Video]. YouTube. <https://www.youtube.com/watch?v=7cu76qMIWIQ> [↑](#footnote-ref-33)