Implementing Crafting Mechanics with Neural Network Pattern Recognition

GDEV60001 GAMES DEVELOPMENT PROJECT

Colin Wood
SUPERVISOR: PETER COOPER
SECOND SUPERVISOR: LUKE BARSBY

Contents

Abstract

Human nature shows evidence that human interaction and its results are at the core of the enjoyment of a video game. Player creativity is also proven to be actively used if the player is given an open-ended task, though it is more likely to happen if they are taught the game linearly first. (Díaza, et al., 2020) Scribblenauts took a different approach to creativity by applying a handwriting recognition system to take in the words of what a player wanted the game to spawn and do. However, the game failed to make handwriting input stable enough. (Kashtan, 2014)

Human creativity and language can be found in doodles, or cave drawings, and have represented human perception and symbolism for many millennia well before written language. (Davidson, et al., 1989) Therefore, the game idea was to recognize singular doodled symbols to craft items in the game, instead of words. Using Python and Keras, the project trained a small, local neural network to recognize a doodle based on pixels of a digital canvas. Using Unreal Engine 5.3, C++ and Epic's Neural Network Engine Plugin with NNERuntimeOrtCPU, the neural network's external thread fires an event with its inference back to the game to spawn the item.

The code proved to be functional, though surrounding game design suggested the mechanic alone cannot make a good game if the other mechanics are flawed. The artefact failed to prove player creativity by having the player draw many items to experiment, but it did find that the nostalgic and natural feeling of drawing one's desires made the player feel creative even if data suggested they were not. This proved the crafting mechanic with doodle recognition is possible and is in demand. Even a small independent games studio could develop it with very low cost, but Unreal Engine in its current state may not be the optimal solution.

Introduction

In the interest of gameplay, players approaching a title may find different reasons to play the game. Some may be looking for a title that provides a linear experience where it's consistently about finding tools and solutions in the environment, whilst others may enjoy an open-ended experience that caters to making their own choice to achieve a goal. Further still, some may take this open-ness to an extreme degree; vouching to have giant open worlds they can play within like a sandbox. Such a term has even coined a genre.

As time moved forward in the video games development field, mechanics such as these have been quite broadly explored. Open-ended games ended up splitting into several subgenres or popular mechanics, such as open worlds, crafting mechanics, character customisation, RPG-style character building, and shelter or factory construction. In the case of these games their software is constructed to simulate the creation of items or characters using rather expected computer graphics user interfaces, which end up reducing the mechanic to either a simple menu, or in-game feedback from the objects in its world.

However, as time moved forward and more mechanics have been explored, a lot of open-ended-ness methods have been considered tried-and-true, sometimes even staples to their genre. Many games have spawned and donned a crown considering themselves the epitome of their respective mechanic types. "Minecraft" for example gives the player to freedom to construct buildings and craft new materials on their own time, giving them no specific goals. (Fan, et al., 2022) And "Scribblenauts" provides players with a simple notepad to write the name and descriptions of the object they want to be spawned into the world to solve puzzles with.

As these games run on computers expecting a wide range of hardware performance, their implementations absolutely tend towards a digital user interface designed to turn the experience into a series of menus that interrupt the actions of the characters in the world. This can, however, deter from the immersion of the experience, or even so far as making the creativity of the player feel rather fake. (Myers, 2003) This could be accounted for in the way user interfaces on a computer may encompass a long learning curve, especially in these games which provide the player with many buttons and menus to have to encounter at a time. This could suggest a gap in the open-ended video games field that such mechanics should be simplified.

This project begins with a focus specifically on the open-endedness of a crafting mechanic, wherein the player should be able to spawn things into the game's world using other resources. However, it goes a step further to research on the idea that "Scribblenauts" brought to light; if a player could suggest what item they wish for with simple human interpreted means, such as handwriting, it may affect the game's design. (Kashtan, 2014) It could do so not only by slimming down the scope of the game itself, but quite possibly the amount of computerized user interfaces it would require learning it, which can reduce the skill curve needed to employ a crafting system.

Aims and Objectives

This project aims to claim evidence that a small independent games development studio can utilize a small-scale neural network to create a game that promotes creativity through drawing mechanics. The main goal is to prove a commercial demand in games being developed with the mechanic. A neural network will be utilized to interpret the player's input regardless of their language of choice and art style and create an item for in-game world interactions accordingly. This has been split apart into several research questions (RQ's) that is hoped to influence the limitations and requirements that would need to be considered.

[RQ1] What is the definition of player creativity and how creative can interpreting doodles allow the player to be? The objective is to gauge creativity with the aim to prove that interpreting a drawing would let players explore more creative solutions possible compared to other traditional open-ended crafting mechanics. In extension to the objective, research will be done to see how creativity is quantified.

[RQ2] What performance hinderances does the implementation of the neural network inhibit? What dataset size is needed to accommodate for players of different doodling style and shape choices to increase the accuracy of the neural network? The objective is to create a neural network for input to a video game crafting system. Its aim is to find a high accuracy measured. The study will dive in depth on if the number of items possible in the game change how accurate and time-consuming the input doodle's processing would be.

[RQ3] What affects to the game's design does implementing doodle-based crafting cause? This will dive more in depth into the idea of reduction of items that must be collected and understood to create another item. The objective is to show in which way does it change the scope of level design to accommodate for a small independent video game studio's development team, and if players have enjoyed the game with a neural network-based crafting system mechanic. The aim is to prove that such a title would be in demand.

Literature Review

Player Creativity Thrives with Open-Ended Crafting Systems

Science has previously defined creativity under two categories, the first of which is a "sociological definition" which explains that its creativity must be approved as such by a wider populous and usually coincides never having been done before by anyone in the world. However, it also gives a secondary "individualist definition," which is defined as "a new mental combination that is expressed in the world". This definition, in contrast to a sociocultural definition, does not require that the created work be not done before, and is not limited to validation by a large group of people. (Sawyer & Hendriksen, 2024) A paper has exemplified this idea with, "In contrast for creating a painting of a creature never designed before, individualist creativity could be to brush one's own teeth with their finger in the absence of a toothbrush." (Fan, et al., 2022)

In the case of video games, a great example of creativity being a mechanical requirement would be Minecraft, whose mechanics require players synthetise information from a virtual world using a broad range of possible interactions. Players of Minecraft must utilize what they have crafted in trial-and-error scenarios to get results. (Kaufman & Baer, 2012) This suggests that learning from success and failures during play along these rules implicitly encourages thinking creatively.

Research proposes that creativity can manifest by a Problem-Solving Mindset which can be established by a task. It defines two types of problems to solve, ones that are "well defined", and ones that are "ill defined". The structure of these two problem types is noted by the differences in three regards; (1) the initial state (the problem itself) and (2) the set of operators (rules and strategies) that allow a user to achieve (3) a goal state (a solution). (Davidson & Sternberg, 2003) Well defined problems exhibit a fixed solution with a set path to that goal, whilst ill-defined problems propose many tools to reach a goal with an open-ended interpretation to success. (Kitchner, 1983) A Problem-Solving Mindset becomes divergent when provided an ill-defined task as it involves experimentation and expanding the problem space's scope for ideas, whilst it becomes convergent when given a well-defined task which emphasises pruning the problem space using "speed, accuracy and logic". (Cropley, 2006) Therefore, a divergent mindset could be said to be employing creativity on their tasks.

However, it also can be said that Creativity is rather a cycle of Convergent Tasks being used to identify the best options for a Divergent task, meaning both mindsets must be deployed at some point for creativity to take effect. (Lubart, 2001) This suggested a mindset is more likely to remain the same after completing one type of task once, suggesting that once a person has achieved a divergent mindset after completing convergent tasks, they will keep that Divergent mindset when completing subsequent tasks. However, the paper that furthered this idea also argues that when a player is presented with more tasks to choose from, they are more likely to choose a task that aligns with their current mindset, rather than to change mindsets and adopt a different kind of task. This promotes an idea that consumers will not only take part in goal setting but will also participate in goal-choosing.

In the case of Minecraft, this is proven to result in a divergent mindset attempting to use several methods and resources. In one assessment in the study (Díaza, et al., 2020), they had found an increase in divergent activity when instructed plainly to "build a house of your dreams". In an extension via another assessment (Fan, et al., 2022), it showed evidence that a player exhibiting divergent activity and being given ill-defined tasks may let them grow accustomed to it, which may

inspire them to take on more divergent tasks in the future. These findings support an argument that a task that encourages a convergent mindset and nothing but that has an absence of creativity.

One might derive the idea that creativity must be enforced from the very start of the player's experience for that player to continue acting creative throughout the rest of their gameplay experience. However there comes the argument that requiring creativity in a game's design too early may reduce the target audience. Specifically, the Minecraft divergent mindset study (Fan, et al., 2022) found that players with more experience and skill in Minecraft were more likely to utilize creativity in their gameplay, whilst those with less experience and skill were more likely to choose well-defined tasks to learn the game. This may also suggest that a game with a creative crafting mechanic may wish to tutorial new players to broaden their target audience to both those with experience and those without. This suggestion specifically idealizes only gradually introducing divergent tasks, rather than heaping them all at once on the player. (Reiser & Tabak, 2014)

A Link Established Between Creative Expression and Handwriting

Diving deeper into player creativity, one known method for this is handwriting within the game. Originally handwriting whilst playing video games was established as a toolset for the player outside of the game, to assist them further within the game. One such example established it as a mechanic in its play, known as Wizardry (1981), with which the game "shipped with a detailed set of instructions on how to draw the map on grid paper, stating that 'mapping is indeed one of the most important skills that Wizardry players possess'." (Barton & Stacks, 2008) Through this the player is expected to personalise their experience with the game's world by "drawing" their interpretations of it. (Myers, 2003) Yet, as technology evolved, the game world could render more of its intended details via the program itself, rather than expecting the player to draw up their own visions. For example, Myst, which takes much inspiration from text adventure games, vouches to render the scene instead completely without words, and uses clickable objects for actions instead of interpreting language. (Kashtan, 2014)

This kind of transparency for a long while had resulted in the reliance of the computer hardware and software to maintain it. Automapping had become a function of most games with large and confusing maps. In exchange, it widened the umbrella for user friendliness, welcoming more users of varying levels of skill to play the same experience without having to learn how to write. Additionally, it promotes a feeling of immersion, as the game world is doing such a difficult task inherently, rather than having it occur outside of the machine. Some games are designed to go beyond this transparency of in world information, to even eliminate per-character information, such as a Heads-Up Display. For example, Dead Space had integrated the player's health and ammo as glowing lights directly on the player's avatar and weapon. Even though a Heads Up Display still renders this information via the machine; by integrating it into the world instead it may result in a more immersive experience as the UI is now actually inside the world. (Kashtan, 2014)

However, two scientists named Karen and Theresa Tanenbaum, argued that the creation of these games must be careful. The games must be capable of drawing a line between what is the in-game world and what is meant for human display, by capturing the idea of "design on meaning" rather than purely designing on freedom. This argument claims that a game that employs freedom without meaning can only result in boredom, as that world does not propose if the rules of the game have been met with actions made by the player. Thus, a game, no matter how open ended its input can be, whether it be by changing the hardware interface, or how the game interprets the input,

must still clearly define rules and constraints to define the boundaries of its experience. By declaring boundaries, the player experiences fun, because then are they able to understand the pleasure derived from playing the game. (Tanenbaum & Tanenbaum, 2009)

This can be linked back to the idea of handwriting. If the player was given complete free reigns over their usage of the pen, such as given an infinitely large sheet of paper, how would the human know where to stop and where to end? It seems feasible to suggest that such a user would find boredom of the task eventually as they had no true goal to achieve. Games by formal definition need an end, and if handwriting is employed within the game, it must have a use case to meet some goal. (Kashtan, 2014)

Kashtan's work was dedicated to finding good implementations of handwriting into a video game medium and had discussed three video game titles for its overview of the technology's evolution. The first of these, considered an entry point of study, was Jet Set Radio. This game's main mechanic tasked the player with tagging property with graffiti in tandem with their gang, whilst escaping the clutches of rival gangs and the police. However, what was most interesting of its play was an offered Spray Editor, which provided the user with a crude painter software where they could edit the visuals of the tag, they would later spray into the game world. However, this game clearly ignored what the user had written, simply copying it pixel for pixel without any extra reward, which ended up making the technology feel useless; it was given no reason, or rather, a reason that fits the niche of players who want their visuals customized, rather than mechanics. (Kashtan, 2014)

The book followed this up with a study on Okami, a game that provides the player with what it calls a Celestial Brush, a tool in the game world that can be used to draw symbols to interact with its world. When the player holds a button, the screen is filled with a paper-like image and then expects the player to draw lines on the screen to instruct the game on what to do, and where in the camera view. For example, a straight line allowed the player to cut the object, or a circle would cause a tree in the game to grow. (Kashtan, 2014)

However, this falls into another short of the technology; it had reduced the player's input created by human handwriting means into an action very simplified, so much so that the drawing completely disappears afterward. Even further still, the game's response did not reflect the subject's uniqueness, instead reducing it down to its own rules. This is reasonable for digital technology; the computer must be instructed to understand the input. However, the game took its restrictions too far by designing its game to expect only one solution for each problem. There were only two areas in the game where any symbol could be drawn, but in both of those areas, the outcome was the same. In all other areas of the game, such as combat, each obstacle required a specific drawing to overcome. (Kashtan, 2014)

Kashtan proposed that his study on a third title had solved many of the issues with handwriting interpretation that the previous two titles had failed to complete. Scribblenauts was designed with a lexicon of tens of thousands of words, which interpret human language input and translate it to spawning an object in the game world and attaching attributes to it via the idiosyncratic nature of the input phrases. It was designed with puzzles that were open ended and could be solved through many means, which was done by programming the objects' attribute components to respond to several input objects and provide multiple kinds of outputs. (Kashtan, 2014)

It accomplished this with two modes of input, one which was a simple keyboard, and the other utilized the touch screen of the Nintendo DS as a drawing pad to write letters one at a time. However, this design was flawed as commonly the game misinterpreted the input letter. This was due to hardware limitations, as the algorithm designed to interpret the drawn letter was rather simplistic and expected specific shapes in its data, which could lead to mild differences in handwriting style being forgotten. For example, it could misinterpret the letter "I" for the letter "L". This suggests a gap in the technology where entire objects could not be interpreted by the system, that it had to be systematically broken down into one that was still flawed. The developers realized that "no handwriting interpretation software could perform as fast as a simple keyboard" (Tringali, 2009), which states a gap in said software to find an implementation that is smooth enough to feel just as fast.

However, this link between creative freedom and handwriting in games discovered by Scribblenauts found its limitations in its sequel Super Scribblenauts. Super Scribblenauts added a system that allowed action words, which translated to actions taken on an object in its world, rather than just spawning the object. When done so, they also designed levels that were almost entirely open ended. This resulted in the player being able to choose just a few words and to rely on them. This suggests that a designer creating an open-ended game with such creative input must design the game with only a set of a few solutions (McShea, 2011), as having too many solutions causes overlap between puzzles that can be used again and again, which can decrease difficulty and increase boredom.

Doodles as a Language

As we explore human input with handwriting, we come across a globally recognized problem. Every person has a specific language they are capable of writing, which even if they share some Latin lettering, may be done in a completely different font. This opens a need for games that recognize human handwriting to take input via shapes of some construct more globally recognizable. Through a study looking at the interpretations of art (Riley, 2019), Professor Howard Riley agrees amongst other psychologists, including most recent as Maslen and Southern (Maslen & Southern, 2014), that drawings can be used as language as well.

Firstly, Howard paraphrased discussions from a psychologist named P. Salmon, who clearly defined an idea that humans culturally go beyond the survival instincts of the animal wherein being able to recognize objects and their changes in the environment but instead can also utilize these visuals to "convey a meaning or reasoning." (Salmon, 1978) This is the basis of language; designed by human intuition to solve the problem to fill the need to "make sense of everything in the world." (Bannister & Fransella, 1980) Humans accomplish this by utilizing signs, which are "used as syntax to represent something else as agreed upon by a community" (Riley, 2019). Usually this is compressed down to an evolved and critically decided upon language of speech or lettering. This also proves a capacity for humans to reflect on what they have experienced, without having to present those items directly.

Along with the notion of language comes its history in human development. New dates are estimated on archaeological findings commonly that prove time and time again that sometime after humans started speaking (Noble & Davidson, 1996), they had started using drawings. (Hoffman, et al., 2018) This form of communication did not compress down the experiences of the person to a repeated, tested, and standardized language, that came sometime later (Schmandt-Besserat, 2014);

instead, the humans would attempt to draw a likeness to what occurred and what was involved as closely as possible, or to “trace” the object, further known today as “observational drawing”. (Davidson, et al., 1989) In extension, though, humanity began to try to represent things that did not exist or are not clearly done with an object to trace. This is where symbols come into play; a drawing may be a shape that must be reinterpreted into something else for the drawer’s point to get across.

This is most interesting for the development of a video game attempting to interpret drawings as a global language, as it aligns with a fact; the object being requested may have a specific shape, and that shape is synonymous with all persons regardless of the language they use every day, as most societies should agree that such a symbol fits that idea. Furthermore, this way of thinking of the evolution of human language suggests that a computer could absolutely interpret the symbol as informational data, rather than sensical data. For example, the user may draw a sword, but the computer needs to be able to conclude that its symbol is indeed a sword, and that its own computations must be able to acknowledge the symbol as such when interacting with it, when “reading” the player’s input. This theoretically would emulate making that computer agree with the user in a social sense that the symbol is the same between them.

Hypothesis on Neural Network Performance Quirks

We may infer from the depictions of objects as doodles that the computer is also needed some agreed-upon rule set to interpret the drawing. More specifically, by instructing a computer to interpret a doodle, the programmer may designate a calculation to translate the incoming doodle to a specific language that the programmer uses. Because the user is human, and can understand the expected symbol, the user could create a database of the symbols and assign the computer to respond with a specific word to represent each generally expected image. For example, if the player draws a hammer, the programmer may instruct the computer to spit out the word “hammer” as a string. How a computer could best accomplish this is discussed further by looking into technologies that scientists have already deployed for interpreting human-drawn symbols, most known as Image Classification.

A thesis on Image Classification contained a state of art that defines some important technologies related to the idea of image recognition (Sorrenti, 2023), which is a technology suggested for handwriting recognition by Kashtan’s research (Kashtan, 2014) and his reference to the technology’s definition (Tappert, et al., 1990). This technology has a lengthy history which first came to digital computing after the invention of a theoretical mathematics model of the human neural, called the Perceptron by Frank Rosenblatt (Rosenblatt, 1958). This is the simplest model of a neuron, with a variable number of inputs, one neuron, and one output. Later technology evolved to stack together multiple neuron simulations into multiple layers crossing between each other, which is what became known as Artificial Neural Networks.

An Artificial Neural Network’s design for neurons is its most important facet. The thesis defined the idea by following a three-step paradigm. First, the dot product of the value of each input and the weights connecting the input to the neuron (also known as the input’s importance), are summed together to form an output. Next, a bias of that neuron is added to its output. Finally, the output is passed to a non-linear activation function to filter the results into something the target will be able to understand. This flow is defined as “Forward Propagation”, and it demonstrates that the computation of the typical Artificial Neural Network is entirely mathematical. (Rosenblatt, 1958)

However, another major facet to the implementation of Artificial Neural Networks has equal if not more processing power poured into it, the idea of learning. This allows the simulated neurons to change their output as they are exposed to more and more inputs, and to give themselves new weights based on the mathematics of training algorithms. If done correctly this can significantly improve the accuracy of the Neural Network and is usually done in three phases. The first is to compute the Loss, which gets the result of the current state of the model. The second is to perform Backpropagation, which is intended to suggest how the weights should be tuned next to minimize Loss, obtained by using a negative gradient Loss function. And the third is to Optimize, which implements the desired tuning based on a training rate that also considers the previous values of the weights in accordance with current output solutions. (Rosenblatt, 1958) These calculations are complex enough to warrant a necessity for more powerful hardware the more neurons are in the field, considering each formula must be executed on each neuron.

The reason to study Artificial Neural Networks is down to their proven significant performance in image recognition (also known as image classification). Inspired by the Visual Cortex of the human brain, a Convolutional Neural Network is evidenced to be most useful for this task. (Rosenblatt, 1958) This model of neural network layout utilizes a Convolutional Layer, which is a set of neurons intended not to directly translate data as the input and output nodes would be, but instead to pick out the features of the input node, or to tell the output node about a feature of a previous layer it had calculated against. It features layer techniques such as Dropout Layers, which inherently cause some neurons to be unable to contribute temporarily to prevent overfitting. Overfitting is the highest risk to image recognition as it can cause the neural network to fail when presented with an image that looks like what is expected, but the neural network cannot understand it as the image might be missing exact details or have a shape not like its trained counterparts.

In the thesis on Image Classification, the paper proposed the idea of Edge Computing to alleviate many factors that affect neural network computation performance. (Rosenblatt, 1958) By bringing the software as close to the client's hardware as possible, there reduces a need to transport sensitive data back up to the cloud just to operate on the task. It also significantly decreases the latency of the application, usually by executing the "edge" on the client's own hardware. Additionally, it reduces cost for deployment as less network traffic is used to transport data between the cloud and the consumer, which is a costly affair especially for smaller businesses. There exists a recent paradigm being officiated as the Artificial Intelligence of Things (Zhang & Dacheng, 2020) which encompasses all these benefits.

This study hypothesises going further than edge computing to fit the game development scenario. More consumers every day vouch for the games they purchase to be installed locally on their machines without the need for internet connection especially if the experience does not require multiplayer. Understanding this, we can confer that the game's development should focus on a platform whose internet connection is purely to transport the game to their PC, then becomes disconnected as the user wishes to play alone. With that the Neural Network needs to be able to run locally on the host's machine.

Keras and Python Benefit for Prototyping Neural Networks

Implementation of the Neural Network itself therefore becomes the most important factor when designing a video game meant to utilize one. In extension, if a smaller independent studio were to take on the project, they would need to consider many costs to adapting it to their smaller

workforce size. The one main aspect that is popular in such a company structure is to focus on tools that are completely free or as close to free as possible to use to decrease monetary pressure on the company. (Lipkin, 2013) This is laid in good example by the business models of the sales for Unity Engine or Unreal Engine. These are much improved in 2024 compared to ten years ago, wherein even these tools were costly to developers looking to start a business using them. (Epic Games Inc., 2024)

Looking at the broad umbrella of tools for the use of neural network development quite a few libraries which have seen source ports to many programming languages promote themselves as publicly usable and even low-cost tools. (Vasilev, et al., 2019) However, amongst them only a few seem reasonable to a small development team looking for easier to use software. Assuming that the development team is comfortable with making the tool modular and sit on its own separate to the video game's repository, one language that has a large following in the neural network field is Python. (Atienza, 2020) Python takes this crown due to notable ease of use especially with its large suite of simulation, analysis, and mathematics prowess. (Hassam, et al., 2023)

One library getting the most focus in this paper is Keras, which is a Python API that sits on top of the library TensorFlow, meaning Keras is developed as a high-level language for working with, generating, training, monitoring, and testing neural networks. "It is designed to allow for speedy experimentation while also providing a delightful development experience". (Ahmed, et al., 2023) This would be of most interest to the development of this project not only due to such a speed factor in implementation, but also because Keras has been proven to be very efficient in computer vision systems, including image recognition and even handwriting recognition, such as the Arabic language within decorated environments, which is difficult if not impossible to do without neural networks. (Al-Barhamtoshy, et al., 2023) This could lead to a reasonable assumption that Keras is well beyond powerful enough to be trained to recognize simple doodles with less clutter in its provided input.

The only things the developer will have to worry much about when working with Keras is what layers to use, the resolution of the input and output shapes, and providing the training data using another library such as Numpy. The input resolution could quite closely coincide with the input resolution of the doodle image itself as a convolutional neural network is intended for the purpose of directly inputting an image into a neural network for processing (Sharma, et al., 2023), though it is worth considering if performance uplifts exist if the input image is of a lower resolution. The output resolution should provide one node per category the network needs to convert the image to, which may change as a developer adds more items to their game that the neural network needs to recognize.

For the purposes of training the neural network, Numpy is an algorithm and mathematics package in Python that can be leveraged to deliver data to Keras in the form of pixels arrays, or even arrays of the doodle draw input as an array of vectors representing the lines drawn by a finger onto a screen. This vector data is in line with a doodle dataset published by Google known as *"Quick, Draw!"* (Fernandez-Fernandez, et al., 2019) This means that an independent games developer can utilize the same, if not a similar or stripped-down version of, the dataset to develop a video game that must categorize doodles of those objects as input, especially since this dataset may provide important variations to train the neural network against that the developers may not have thought of at the time. There are many examples of this dataset being used by developers which can assist in a team's learning on how to use Keras for this purpose. (Kothawade, et al., 2020)

Evaluating and Adapting Performance of a Locally Running Neural Network

The time to compute of the neural network after shipping becomes one of the last considerations for the game's developer. It is expectable that computations must be done in a way that does not stop the game's main thread, lest the engine start to lag, as neural networks are computationally expensive and take a very long time to complete. The game should be able to leverage asynchronous multithreading technology to process the neural network and subscribe to when it has completed for the game to give feedback sometime later. Multiple technologies in games development prove asynchronous code to accelerate the processing time of an application, a great example of which is the Source engine which places the physics engine off thread and then lets it run on its own asynchronously. (Leonard, 2007)

This project should estimate that the accuracy of the neural network must be rather high to reduce player frustration when learning to play the game. For the sake of player creativity, if a neural network is to be deployed to support such thinking, the game must be able to accept as many interpretations of input as possible. For example, if the player were to doodle a sword with several jewels and trinkets and instruct the in-game neural network to turn it into a sword that can interact with the in-game world, that neural network must be able to still interpret that drawing as a sword no matter its decoration. Meanwhile, that neural network must not be able to mistake it for some other object, such as a rock, which is not at all in-line with the intended use of the item the player is looking for.

Neural network accuracy can be measured in a quantifiable manner, especially during gameplay testing internally. Sorrenti's thesis for Image Classification suggests several algorithms that address how well the model can make correct predictions. Hypothetically, if outputs could be classed binarily as Positive or Negative, a model may exhibit results that are True Positive, False Positive, True Negative or False Negative. A Receiver Operating Characteristic curve chart can be used to plot the results of the model by bounding its Recall (how often the neural network reports a Positive) versus its False Positive Rate (how often the positive reported was false). The closer the curve is to an inverse negative logarithmic function, the more accurate the model is (by proving its False Positive Rate is as low as possible with as few reports necessary). (Sorrenti, 2023) Furthermore, a generalized Accuracy function can be utilized instead, which considers all four possible output connotations and suggests how likely it is to output a True Positive result.

The last thing to note on the implementation of a Convolutional Neural Network is the issue of overfitting which is introduced in every neural network model. Overfitting is defined as the process of a neural network's learning algorithm resulting in it giving results rather rigidly and less capable of fluidly transitioning between multiple inputs. For example, if a neural network is trained against a flower and a bee, but is given an image of both, a model experiencing overfitting will only output one or the other, whereas a properly trained model will output half assurance in both a bee and a flower. Inherently, Convolutional Neural Networks thrive by being wider in the network per layer rather than deeper, which results in less overfitting, however it is still a risk. (Sorrenti, 2023) This risk could coincide with miscategorising one item for another if the doodle is rather messy. Therefore, the developer must reduce overfitting as much as possible to reduce cases of the neural network getting confused, which would decrease accuracy.

Research Methodologies

Technologies Used

As noted by research, there are a lot of different tools that could be used to accomplish the same goal, that being of a small studio developing a doodle-based crafting mechanic in a video game. However, for the sake of scope, this project established chosen toolsets for its artefact, there in before proceeding with this project's own research via user feedback of gameplay testing. These are quickly noted before further explained in their relevancy, design use, implementation ideas, and testing method regarding each of the three research questions laid out before.

- **Unreal Engine** – This will be the game engine of choice as it is a popular tool amongst many studios, sometimes including independent ones especially those looking to make action games with lots of performance clearance overhead. Specifically, the experiment will be using Unreal Engine 5.3 with Lumen, Virtual Shadow Maps, and Nanite turned off, which will widen that overhead significantly, allowing the project to measure neural network performance more accurately.
- **C++** - The experiment will be using Unreal Engine's C++ coding suite to build the game application, including integration with Unreal Engine Neural Network Engine, which allows the developer to integrate ONNX formatted neural networks directly. This is the most efficient method to do so in Unreal Engine.
- **Python and Keras** – Taking the recommendations of Hassam and Vasilev (Hassam, et al., 2023) (Vasilev, et al., 2019), this study will implement Keras with Python for training. Later the resulting network will be installed into the video game's environment and utilized for back-end processing of the doodle data to instruct the game on which item to create. This is done by compiling the network to ONNX, and then loading it with the Neural Network Engine library.
- **Microsoft Office** – Microsoft Word will be used to assist in the creation of designs for each of the game's objects so that the experiment can have plenty of planning ahead of time to flesh out the required ideas before wasting any time within the Unreal Engine project. Microsoft Forms will be used to develop feedback questionnaires for gameplay testing users to respond and to easily collect and categorize the data.
- **Game Resource Websites** – SketchFab, Fab, Kenney.nl, Mixamo.org, among others, are common resource sites to grab free and easy to use 3D models and sounds that can be used within games. Considering the nature of this project to be non-commercial, it aligns even further by giving more license opportunities for such assets. Further fitting still, considering independent video games studios prefer to prototype games using such assets if possible or required, (Lipkin, 2013) it fits with the theme question of this project.
- **Google's Draw This! Dataset –** To train the neural network according to doodles, the massive archive of Google's dataset is the best opportunity for it contains tens of thousands of individually submitted drawings of the same category of item. This theoretically will massively increase the accuracy of our doodle recognition mechanic. There are many examples of this being done before, (Fernandez-Fernandez, et al., 2019) however the idea is to create a new network from scratch, one stripped down to only the items the game will implement. This will be achieved using Keras.

Approaches for Implementations

[RQ1] Inciting Player Creativity through Gameplay and UX

As science suggested, the project's first goal should be to spin creativity up. As defined before, creativity from gameplay is achieved by promoting the use of ill-defined tasks to create a divergent mindset. (Cropley, 2006) To align with Reiser and Tabak's estimations, the project should introduce these ill-defined tasks gradually (Reiser & Tabak, 2014) using a tutorial and a progression-based learning curve, which will introduce the player to the game's main mechanics, and the resources they have available to them gradually. These lessons need to be able to be looked back upon by the player even in later gameplay, to reduce the skill curve of memorizing all possible shapes one could draw.

For a task to be truly ill-defined, it must adhere to the two most important distinctions between a well-defined task and an ill-defined task, that being that each task's goal should not be defined as only one, but to provide multiple, and that such tasks should be achievable by multiple means, (Kitchner, 1983) even possibly as far as to not suggest them directly to the player after tutorial had already completed. However, in video games it can be quite tricky to properly establish a ruleset via open-ended-ness, (Tanenbaum & Tanenbaum, 2009) so there are multiple regards as to restrictions the prototype should establish on the player's freedom whilst not sacrificing a feeling of open-ness. This study should provide a single final goal to the player but allow the player to split the path to that goal into several sections. These sections should vary between well-defined tasks and ill-defined tasks based on progress.

Within the context of a crafting mechanic, the doodle recognition system being used to provide the several resources should be trained with a wide number of items that can be spawned into the world. Some items should share at least a little contextual use cases with each other, such as a group of weapons, a group of tools and a group of key items, as did Scribblenauts (Kashtan, 2014), but each item should still have a distinct use case even with such shareable reasoning. Hypothetically, this would provide the player with the freedom to address their next goal as is suggested with the definition of an ill-defined goal. (Cropley, 2006) It is also important to note that the items the network has been trained for must be implemented into the game to satisfy the entire loop of the mechanic's intended functionality.

[RQ2] Maintaining Accuracy of a Neural Network for Interpreting Doodles

The project should implement a Convolutional Neural Network or any relevant variants of it to implement doodle recognition for the crafting mechanic. (Vasilev, et al., 2019) With this technology comes its quirks especially with the model's fallacies toward what the human player has drawn on the screen. Therefore, several aspects of the network need to be considered when building the framework to train it and any performance concerns regarding its execution at runtime specifically on about its weight on the processing power of the machine, in addition to its accuracy.

First is the idea of designing a specific item set to train the data against, by stripping away the extra materials within the Google Draw This! Dataset. Since the original repository has 330 different categories, (Fernandez-Fernandez, et al., 2019) picking out only a few will not only reduce the scope of the project but should also reduce the time it takes to train the neural network. The data for this should be measured based on a timer implemented in Python, and a timer implemented

in Unreal Engine, to count both how long the data took to train, and how long it took to process in real time in the game.

Second comes the concern of having to execute a neural network in a video game scenario. Considering the intended genre of the project's artefact is to be action oriented, the study should consider the impact of the time it takes for the neural network to finish its calculations. This could be linearly affected by the resolution of the captured image canvas used to interpret the player's doodles, so a reasonable resolution should be decided upon, which may be further swayed by the time it takes to train the data with such a resolution as well.

One solution to help avoid stuttering or freezes in the game during calculation would be to employ multithreading, as suggested by Valve's study on putting their engine's physics on a separate thread, (Leonard, 2007) but this should not deter from a good resolution choice as we want to minimize the amount of time that extra thread is alive and waiting for. This project should not employ Edge Computing, (Sorrenti, 2023) even though such technology would be like the asynchronistic execution multithreading provides, as it would involve a secondary machine that is out of scope for the project's testing phases. Instead, the neural network should be optimized to execute locally.

[RQ3] Effects of Doodle Based Crafting on Game Design and Enjoyment

Game enjoyment and game design are rather complicated and open to interpretation fields in the realm of video game development, especially within the independent studio market. Following the theme, as such a small studio looks towards implementing doodle-based crafting into their game, they will also need to consider how it will affect the game's design, and what changes to the typical action platforming game will be required to implement such a mechanic cleanly.

The game loop of a crafting mechanic usually interrupts the flow with a menu that provides the player with a screen that gives them time to create the tool they're after. Two implementations variate ways to do it, one that pauses the game whilst crafting such as in Scribblenauts, (Kashtan, 2014) and one that does not such as in Minecraft. (Fan, et al., 2022) The artefact should decide on which makes more sense for reducing difficulty of developing the game, as a smaller studio may find trouble in implementing more mechanics surrounding just the one.

Considering the performance of the neural network itself in how reducing the item counts can increase its performance, whilst also considering that an open-ended game should inherently include a lot of options for the player to craft with the mechanic; the artefact needs to strike a balance between the two. It should have just enough items to give the player a choice of at least two or three options for each task slice within the entire level.

Testing Methods

The questionnaire and artefact will each need to collaborate on data collection via communicative means. This will be done with playtesting of the artefact with each test subject in the group, which will first be provided specific data on the screen at the end of the game, then expected not only to mirror that information into the questionnaire, but also provide some quantifiable feedback in different forms. The rest of the questionnaire's details are explained in the list below.

- **The creativity of the player** – Measurements need to be taken out in a quantifiable manner. This should be accomplished by providing the artefact with simple telemetry that should count how many times the player crafted each item type, and how many times they used them. It could be possible to go a step further with this by comparing with the game's intended design, like the testing done in the Minecraft "build a house of your dreams" study. (Díaza, et al., 2020) However, the analysis of the data must also consider the final game design's usability of the item, and at what point in time did they unlock the ability to draw the item via progress. If the game allows a specific item for only a short time towards the end of the game, it should relate the data to such, or the game should allow the item to be used a lot more at such a time.
- **Accuracy of the Neural Network –** The artefact's neural network needs to be trained with an accurate variant of neural network choice, but it also needs to consider taking more processing time to increase its accuracy during training. Additionally, careful programming must be utilized when integrating the network into Unreal Engine, as mistakes on the handling of nodes could mis-wire which response the neural network wishes to give to an input. This accuracy will be measured with rather traditional neural network accuracy measuring means as done in Sorrenti's study. (Sorrenti, 2023) Data collection for this testing should be carried out by asking the user at the end of every doodle creation how accurate the neural network was by categorizing its response as "Positive or False Positive." Afterwards, the data will be totalled and displayed in the telemetry at the end of the game.
- **Measuring Enjoyment and Demand –** For an action platforming game of the artefact's kind, it needs to understand the familiarity of the user to the genre, and their relative skill level. This could significantly impact their enjoyment of the game's design inherently by their tastes alone, which is given evidence in the study on Minecraft (Fan, et al., 2022) and its relation to creative gameplay. This effect on the data could change the future of a doodle-based crafting mechanic and could inflict a shrink on the scope of the target audience. The play testers' opinions should be considered as important data, beyond just testing their creativity and the performance of the neural network, as it will make evidence of the results of the prototype's game design. So, the policy should be to first collect factual data on if the player has had experience with the genre before, followed by collecting their opinions with quantifiable ratings based on their enjoyment of the artefact's implementation of the genre, and finalized with an opinion on if they would purchase a game with the doodle crafting mechanic. This would tell if a game studio could see success in fully funding a project of its kind, and if they need to reduce the project's complexity in level design.

Results and Findings

Design

The first consideration required understanding how the interaction to the neural network should be laid out to minimize performance impact, including regards to memory usage, performance hit from the neural network being executed, and the amount of time it would take to grab screen data from the game to interpret a doodle with. The optimal solution was achieved by making sure only a single instance of the Scene Capture Component was ever present at a given time, followed by moving its transform onto a specific easel based on the player's interactions. This was reasonable as the player would only ever be required to draw on one easel at a time. Additionally, to provide the level design with a way to make easel interactions flow with a rewarding feel, inkwells were created which were used as pick-up resources for the player spend as currency on drawing doodles.

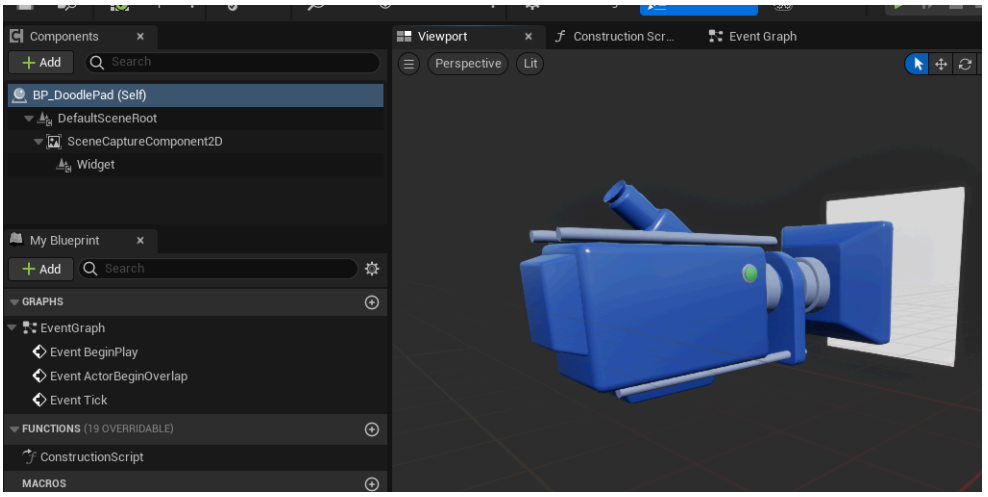


Figure 1: The render target used to get pixel data and doodles from the player

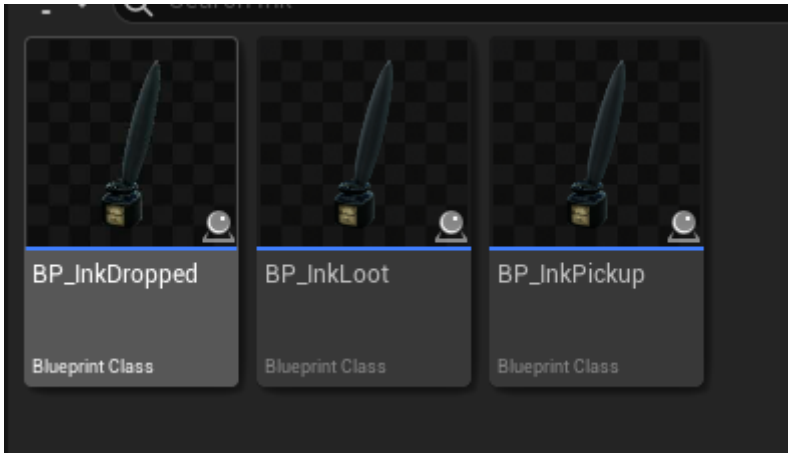
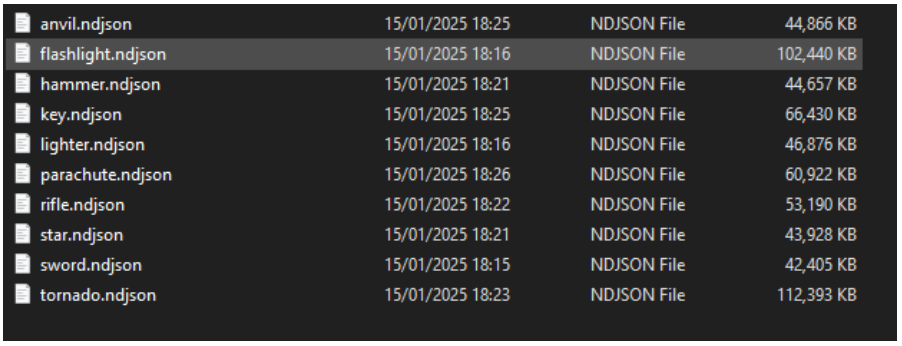


Figure 2: Ink Actors for the player to fuel their doodles

Once the idea of how the code would flow was confirmed, the next approach involved selecting items from the dataset that could be both as unique as possible in shape, whilst also providing unique experiences for what a player could expect from the shape. For example, one might be able to interpret a Tornado as a Spring if drawn simply by circles, an estimation provided by deductions of "symbols" as noted in Riley's study on simplified art (Riley, 2019). Then, items present in the Google Quick Draw dataset chosen to provide a broad selection of routes to approach a problem, such as destruction, combat, platforming, puzzle solving, or providing lights.



anvil.ndjson	15/01/2025 18:25	NDJSON File	44,866 KB
flashlight.ndjson	15/01/2025 18:16	NDJSON File	102,440 KB
hammer.ndjson	15/01/2025 18:21	NDJSON File	44,657 KB
key.ndjson	15/01/2025 18:25	NDJSON File	66,430 KB
lighter.ndjson	15/01/2025 18:16	NDJSON File	46,876 KB
parachute.ndjson	15/01/2025 18:26	NDJSON File	60,922 KB
rifle.ndjson	15/01/2025 18:22	NDJSON File	53,190 KB
star.ndjson	15/01/2025 18:21	NDJSON File	43,928 KB
sword.ndjson	15/01/2025 18:15	NDJSON File	42,405 KB
tornado.ndjson	15/01/2025 18:23	NDJSON File	112,393 KB

Figure 3: Select NDJSONs of specific items from the Google Quick, Draw! Dataset.

The level's design was next. Each room had two or three items involving it so that multiple instances of the same problem could be approached differently from the other. As mentioned in Tanenbaum and Tanenbaum's study on game rulesets and freedom (Tanenbaum & Tanenbaum, 2009), this artefact needed to restrict the number of solutions so that its paths would make sense and changing paths over the course of the level would have a use case.

To help reduce the chances of the player getting lost in the new design, regardless of their previous experience in video games of the same genre or its subsets (related to crafting and platforming), tutorials were thought out. Initially the level laid out its tutorials to introduce the player to each of the mechanics of the game, including different methods of jumps, how to light up darkness, how to drop items, how to interact with objects, and how to combat enemies. These tutorials were done with symbols instead of plain lettering wherever possible in hopes to reduce the language barrier, as stated before.

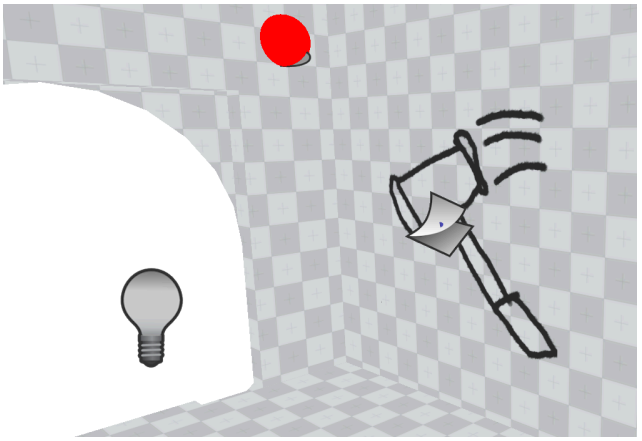


Figure 4: An example of a tutorial, this one being for the Hammer item.

And finally, a user interface was designed for the game to provide information to the player on their current state, such as their inventory and collected ink pots. This mechanic gave them information on when interaction with an easel was possible. Once the player interacts with the Easel a new user interface would display, asking them to confirm when they are done drawing, whilst also displaying what the network currently thought of the doodle on the easel over a set interval. This allowed the network to respond as quickly as possible so that the player could adjust or erase their drawing to avoid situations where the network would not spit out an item they wanted. Additionally, for the later purposes of testing the neural network's accuracy, a button was added so the user could declare a network inference as incorrect.

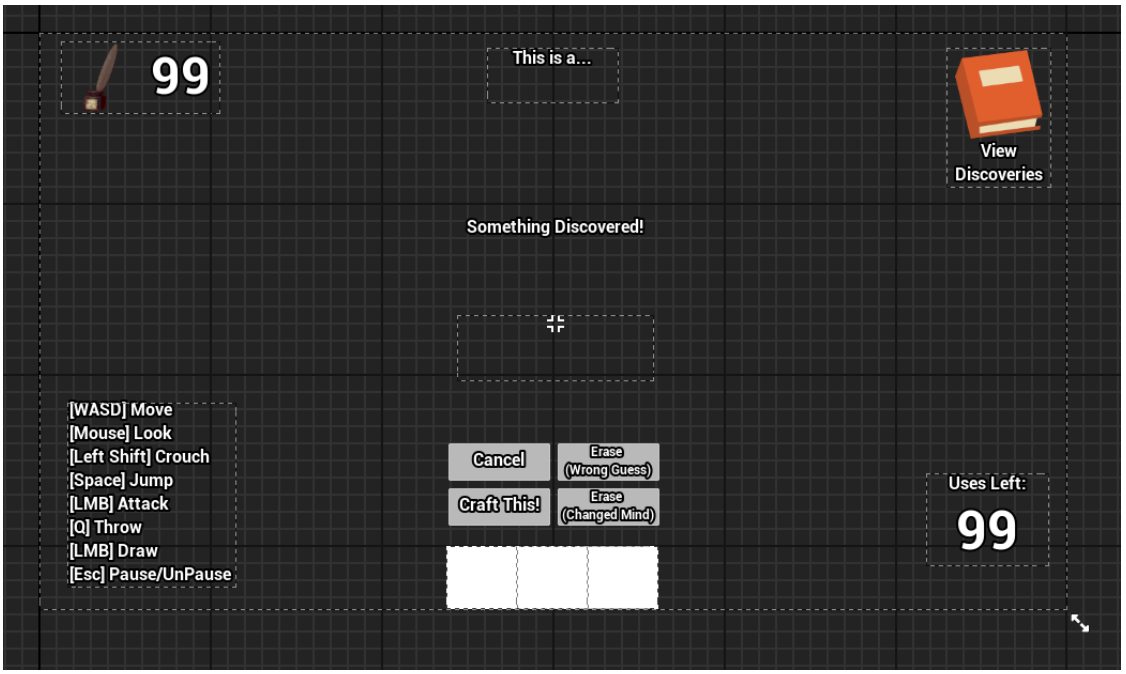


Figure 5: The main User Interface, with Easel controls that get disabled, and Text that gets modified at runtime.

Implementation

Implementation began with the items that the game was going to introduce, as well as the player's character pawn. Alongside it, the Items that the game's design specified were implemented in their fullest, including each mechanic each item brought to the game, and how they should physically collide with objects or attach to the player. These items used a base class for easy item interactions such as picking up and attacking, but the actual behaviour of the class was done with ActorComponents, to let items share behaviours with each other if needed, and to reduce bloating inheritance. This also made it easy to sort the actor's configurations in the editor.

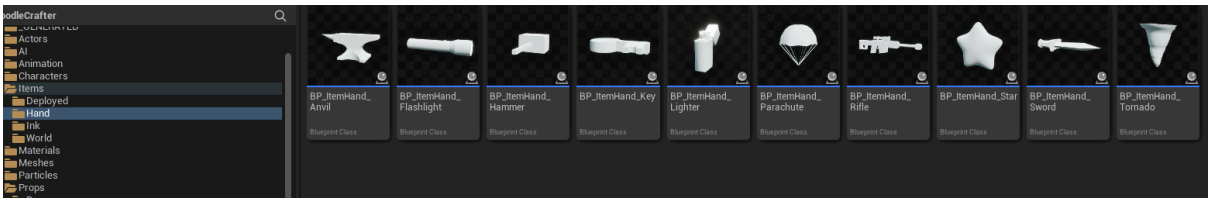


Figure 6: All ten implemented items in their Hand Item form.

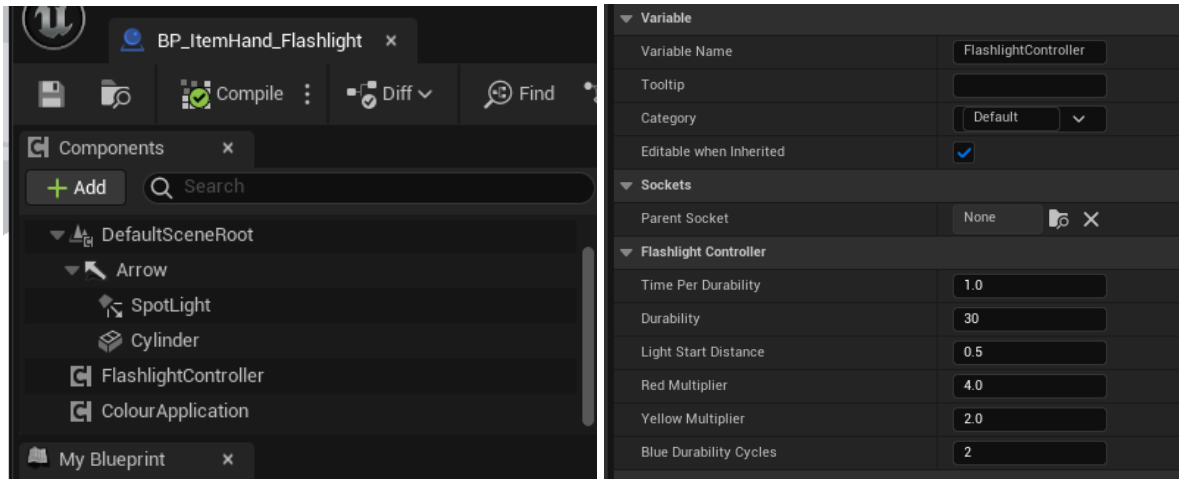
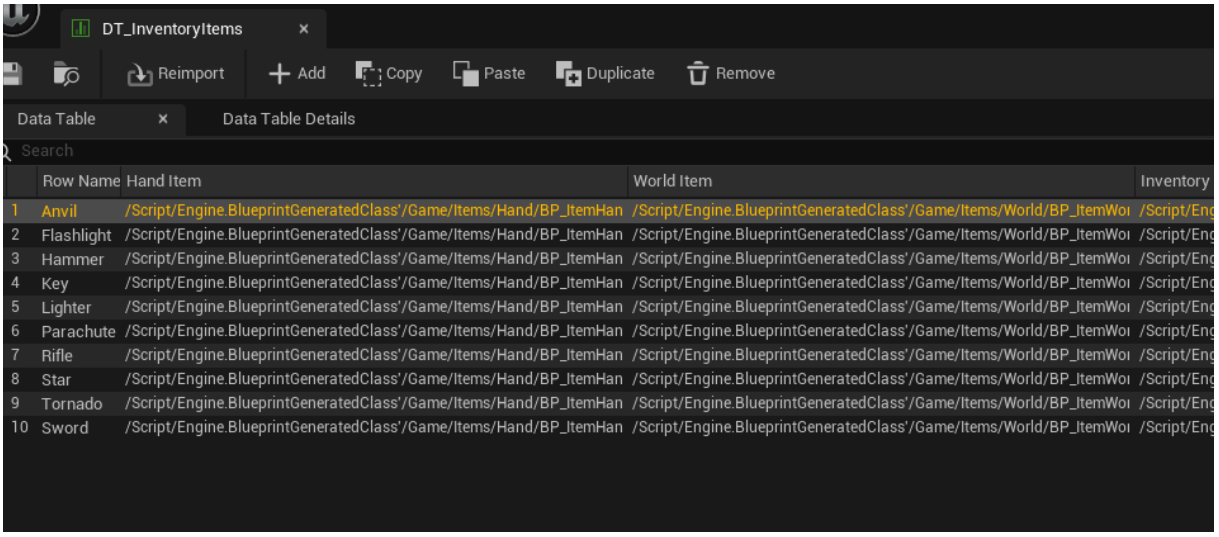


Figure 7: An example of the actor components being used and their configurations

A practice was taken on to implement as much of each actor's code in C++ as possible for the purpose of integration to the crafting mechanic which requires C++ later. This was followed by using a Blueprint to keep memory safety over all interconnections and exposed variables, for later tweaks during the development process. This included the usage of the Item Inventory Data Table, which is a table of all possible items sorted by ID via FName classes, which then points to the two item classes for Item World and Item Hand, alongside a Texture for the Inventory user interface to display.



	Row Name	Hand Item	World Item	Inventory
1	Anvil	/Script/Engine.BlueprintGeneratedClass'/Game/Items/Hand/BP_ItemHan	/Script/Engine.BlueprintGeneratedClass'/Game/Items/World/BP_ItemWo	/Script/Eng
2	Flashlight	/Script/Engine.BlueprintGeneratedClass'/Game/Items/Hand/BP_ItemHan	/Script/Engine.BlueprintGeneratedClass'/Game/Items/World/BP_ItemWo	/Script/Eng
3	Hammer	/Script/Engine.BlueprintGeneratedClass'/Game/Items/Hand/BP_ItemHan	/Script/Engine.BlueprintGeneratedClass'/Game/Items/World/BP_ItemWo	/Script/Eng
4	Key	/Script/Engine.BlueprintGeneratedClass'/Game/Items/Hand/BP_ItemHan	/Script/Engine.BlueprintGeneratedClass'/Game/Items/World/BP_ItemWo	/Script/Eng
5	Lighter	/Script/Engine.BlueprintGeneratedClass'/Game/Items/Hand/BP_ItemHan	/Script/Engine.BlueprintGeneratedClass'/Game/Items/World/BP_ItemWo	/Script/Eng
6	Parachute	/Script/Engine.BlueprintGeneratedClass'/Game/Items/Hand/BP_ItemHan	/Script/Engine.BlueprintGeneratedClass'/Game/Items/World/BP_ItemWo	/Script/Eng
7	Rifle	/Script/Engine.BlueprintGeneratedClass'/Game/Items/Hand/BP_ItemHan	/Script/Engine.BlueprintGeneratedClass'/Game/Items/World/BP_ItemWo	/Script/Eng
8	Star	/Script/Engine.BlueprintGeneratedClass'/Game/Items/Hand/BP_ItemHan	/Script/Engine.BlueprintGeneratedClass'/Game/Items/World/BP_ItemWo	/Script/Eng
9	Tornado	/Script/Engine.BlueprintGeneratedClass'/Game/Items/Hand/BP_ItemHan	/Script/Engine.BlueprintGeneratedClass'/Game/Items/World/BP_ItemWo	/Script/Eng
10	Sword	/Script/Engine.BlueprintGeneratedClass'/Game/Items/Hand/BP_ItemHan	/Script/Engine.BlueprintGeneratedClass'/Game/Items/World/BP_ItemWo	/Script/Eng

Figure 8: The InventoryDataTable filled with all information for all ten items

```
UPROPERTY()
Changed in 0 Blueprints
ADoodlePad* DoodlePad;

UPROPERTY(BlueprintType, EditAnywhere)
Changed in 1 Blueprint
UDataTable* InventoryDataTable;
```

Figure 9: Example code of keeping assets memory safe on actors

Later, the enemies designed were implemented into the project, and a small testing room was created to ensure their functionality within. Again, this included exposing as much of their behaviours and rulesets to Blueprint as possible, whilst keeping actual functionality within C++. Behaviour Trees were utilized to make sure they could navigate within the Nav Mesh Bounding Volumes. These three enemies were given rather basic behaviour trees and animation state machines to make their implementation simple and fast.

Once the actors the game required were implemented, the easel and its neural network integration were put into place. The neural network's Scene Capture Component was decidedly set to a resolution of 64 by 64, which was also used as the value by the neural network itself. This data was then grabbed by blocking the game thread until the texture was copied, and then a for loop was used to get a greyscale representation of every pixel on the texture.

```
void ADoodlePad::GetGreyscaleData(TArray<float>& Values)
{
    TArray<FColor> PixelValues;
    FRenderTarget* Render = RenderTarget->GameThread_GetRenderTargetResource();
    Render->ReadPixels(PixelValues);

    for (int x = 0; x < 64; x++)
    {
        for (int y = 0; y < 64; y++)
        {
            // split into RGBA
            int i = (x * 64 + y);

            // Normalize to 0.0f - 1.0f.
            float FinalValue = ((PixelValues[i].R + PixelValues[i].G + PixelValues[i].B) / 3.0f) / 256.0f;

            //UE_LOG(LogDoodleCrafter, Display, TEXT("Greyscale was %s"), *FString::SanitizeFloat(FinalValue));

            Values.Add(FinalValue);
        }
    }
}
```

Figure 10: Code to safely get the pixels from the RenderTarget.

Additionally, the widget that the image was getting data from had implemented a Line Drawing system that reacted to mouse clicks. Notably this still required that the Widget exist in world space, so the Player Character had a Widget Interaction Component attached and set up. The following diagram sums up a flow chart of the entire crafting mechanic's input system implementation as intended.

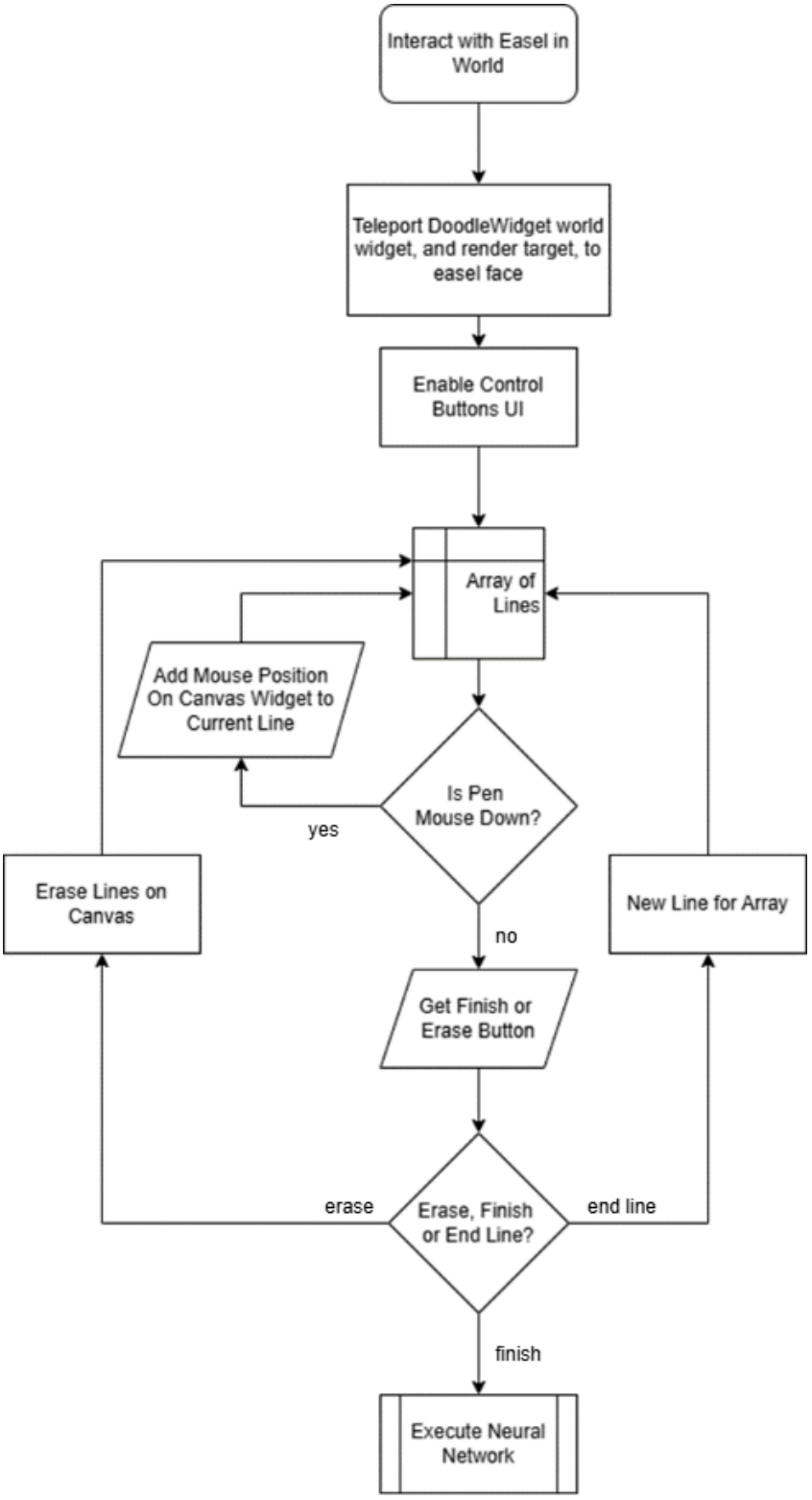


Figure 11: Flowchart showing how to gather lines as doodle data

Afterwards, the Neural Network Engine plugin was integrated into a Game Instance Subsystem so that it could sit in place, be constructed only once to save time, and be called by any actor that needed its data at any moment. The neural network execution itself was placed onto a separate thread pool using the ASyncTask helper function. Delegates were used to alert the player actor of what item was chosen by the neural network.

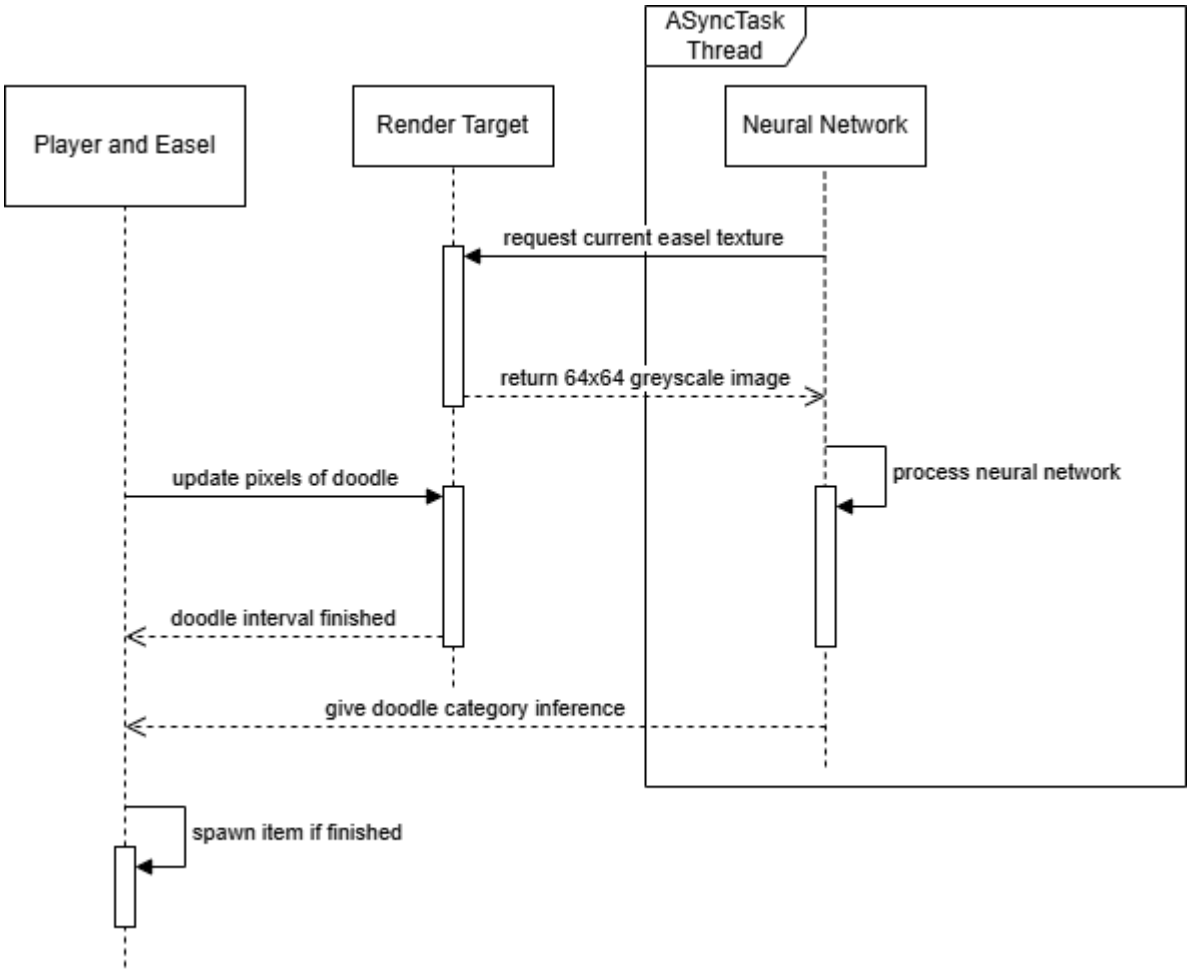


Figure 12: Sequence Diagram showing the execution of the neural network off-thread, and how the Render Target blocks the entire game thread to calculate its pixels.

The neural network itself was finally implemented with Keras. PyCairo was used to translate the NDJSONs provided by Google's "Quick, Draw!" dataset, into 64 by 64 pixel black and white PNG image files. Afterwards, Keras loaded the data and generated a dataset with it. This utilized the deprecated, yet still available, ImageDataGenerator to categorize the images automatically based on the name of the subfolder the PNG was placed in. As a second iteration however, the data was handpicked to reduce cases of doodles that were too like other item types.

Figure 13: Initial PyCairo results of all data from Google Quick, Draw!'s NDJSON.

Figure 14: Hand-picked data that decided on a uniform set of expected details (two large fingers, and a circle base)

Lastly, a network was compiled using Keras' Sequential type, and the Adam optimizer. The network in total had four layers deep of information and used 5 by 5 two-dimensional filters for each node. This was the second iteration of this network as further depth was required to increase the network's accuracy, as was reasonable due to the results of a previous debugging test.

```
print("Getting images (this may take a very long time!)")

train_generator = train_datagen.flow_from_directory(
    'training/',
    target_size=target_size,
    batch_size=8,
    color_mode="grayscale")

num_filters = 32
filter_size = 5
pool_size = 2

# Build the model.
model = Sequential([
  Conv2D(num_filters, (filter_size, filter_size), input_shape=(64, 64, 1), padding="same"),
  MaxPooling2D(pool_size=(pool_size, pool_size)),

  Conv2D(num_filters * 2, (filter_size, filter_size), padding="same", activation="relu"),
  MaxPooling2D(pool_size=(pool_size, pool_size)),

  Conv2D(num_filters * 4, (filter_size, filter_size), padding="same", activation="relu"),
  MaxPooling2D(pool_size=(pool_size, pool_size)),

  Conv2D(num_filters * 8, (filter_size, filter_size), padding="same", activation="relu"),
  MaxPooling2D(pool_size=(pool_size, pool_size)),

  Flatten(),
  Dense(10, activation='softmax'),
])

# Compile the model.
model.compile(
  'adam',
  loss='categorical_crossentropy',
  metrics=['accuracy'],
)

# Train the model.
model.fit(
  train_generator,
  epochs=25)

# Save the model to disk.
print("Exporting model to {}".format(os.path.join('output/', 'model.onnx')))

input_signature = [tf.TensorSpec(model.inputs[0].shape, model.inputs[0].dtype, name='digit')]
tf2onnx.convert.from_keras(model, input_signature, output_path = os.path.join('output/', "model.onnx"), opset=13)
```

Figure 15: Python Keras code for the neural network

To finish off the experience, the level's design was implemented using Unreal's modelling tools to provide CubeGrids for the level layout. To reduce the performance hit that dynamic lighting may provide, static lights and stationary lights were used wherever possible. Another step used to increase performance was the process of baking these lights into lightmaps using GPU Lightmass. Virtual Shadow Maps, Lumen and Nanite were turned off to further raise the performance of the final product, to ensure the only bottleneck would be what is created by the player, both doodles and items. (See Appendix C) for the final gameplay result.

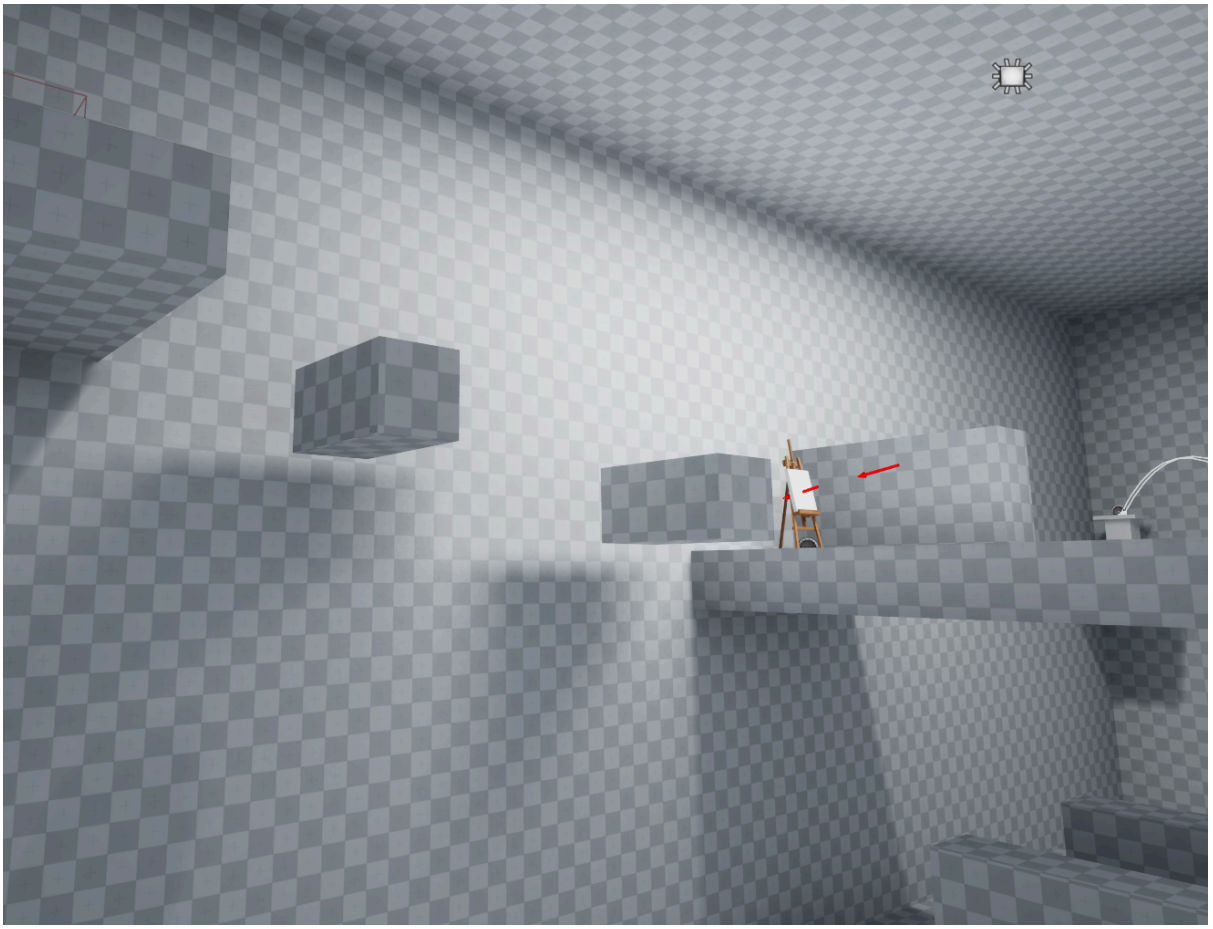


Figure 16: Some examples of lighting and final layout in the scene.

Testing

Addressing player creativity, statistical data during gameplay was collected which simply added the total amount of items together or graphed each one on their own. This then looked at how widespread the use count was for each item, gauging if some were affected by the level design, weapon design usability, and other factors. A total of 15 users submitted their json data, whilst 3 others opted out of doing so, and 3 other users were unable or opted out of attending testing altogether.

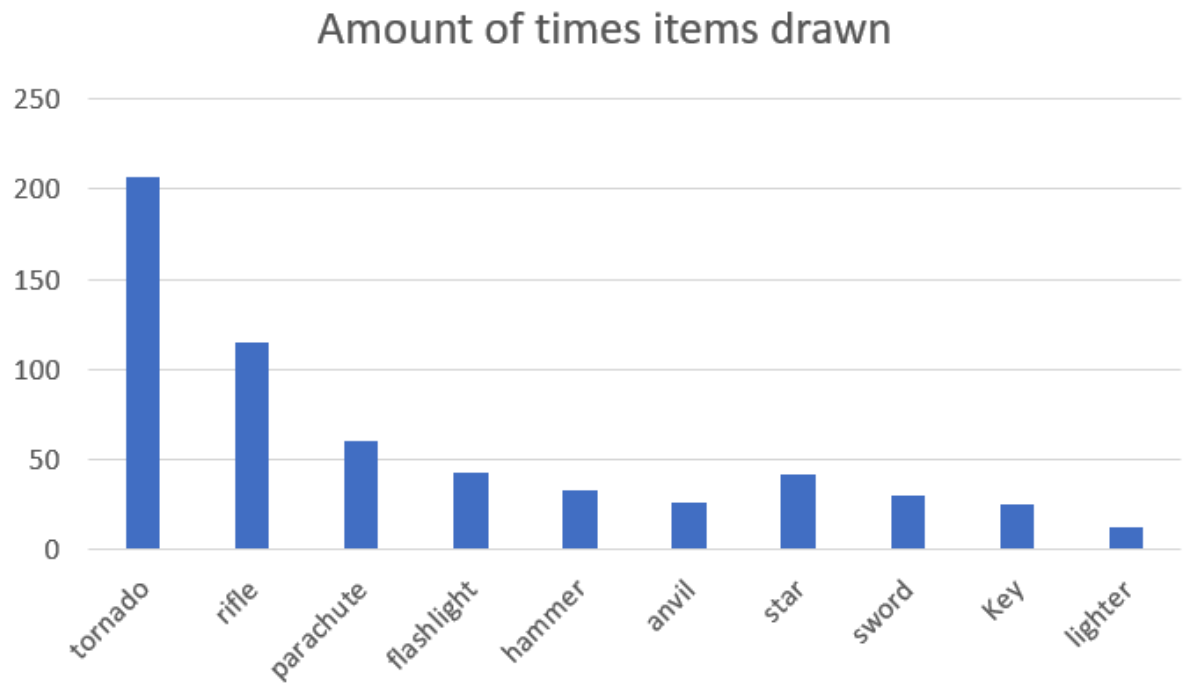


Figure 17: Bar graph showing commonality for the Tornado item, a preference for the Rifle in combat, but a lack of expected appreciation for the Hammer or Sword, alongside an expected distaste for the Lighter

The neural network's performance was tested by allowing players to hit a "Erase (Wrong Guess)" button in the event the neural network did not give a reasonable result to their doodle. This provides "False Positive" data for the network, which is used to decrease the accuracy in the resulted formula. The values reported by the game have been graphed and its average was calculated. For raw data (See Appendix B).

The calculated average found that players reported the network had approximately 68.2% accuracy.

Players were asked to give their previous experience on games in the genre that included a crafting mechanic by any means. This was used to ensure a removal of that bias from the data, as it would prove to be important for the later questions. A very large amount in the amount of 76% of the selected set of people reportedly previously played Minecraft.

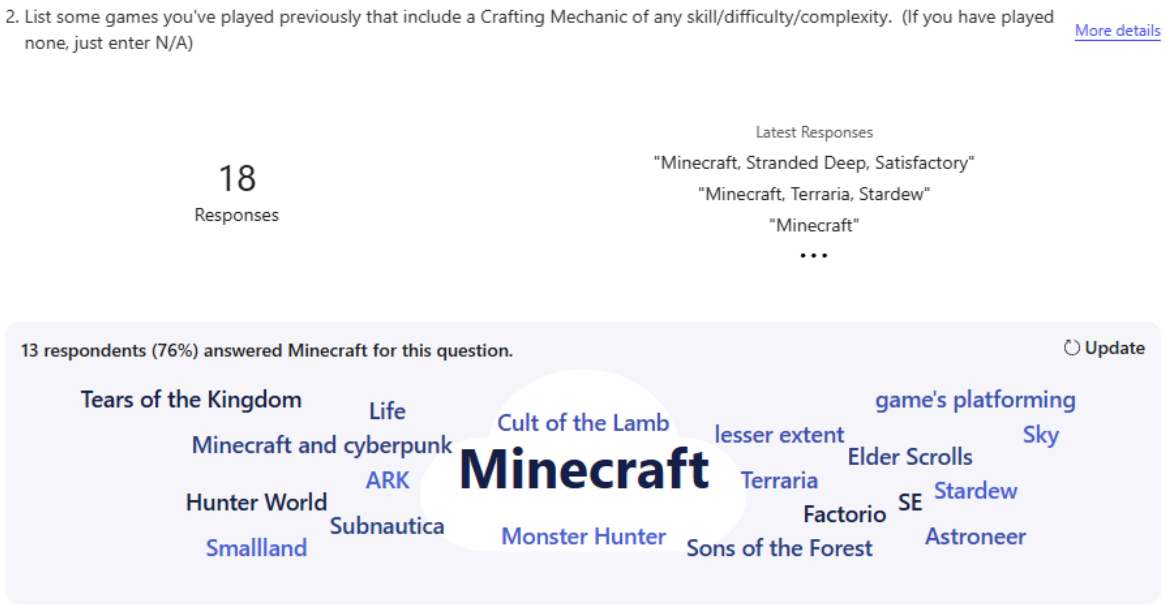


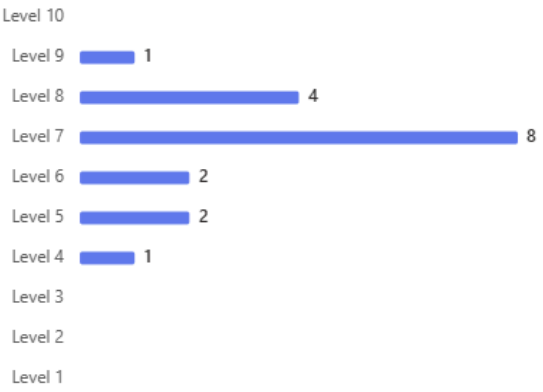
Figure 18: Word Cloud of respondents listing games they had previously played with a crafting mechanic.

Next, the game's design was put to opinionated yet quantitative testing, and compared against the games the players previously had experience in. A very large amount of the players reported Minecraft as a previous experience. The level's design was reportedly mixed in reviews; however, the mechanic's response was marginally better than the level's design. Another question asked how creative the player felt they were during gameplay, which showed a significant portion felt their creative voice was well utilized in the mechanic.

4. How functional and responsive was the crafting mechanic?

6.83

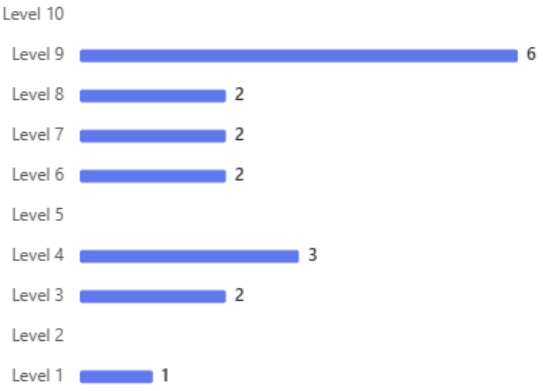
Average Rating



5. How enjoyable was the level design?

6.39

Average Rating



6. How creative did you feel when playing the game?

More details

7.17

Average Rating

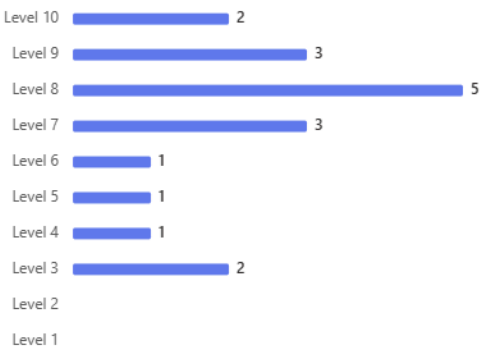


Figure 19: On a scale with 1 being terrible, 5 being unnoticeable, and 10 being outstanding, players were asked to rate the quality of the crafting mechanic, the level design, and how creative they felt.

Players were then asked to give opinions on the mechanic and their experience with it beyond performance, and instead oriented towards enjoyment. The feeling of how natural the mechanic felt to work with showed significant improvement over predecessor crafting systems in the genre. Afterwards, almost all players reported they have demand to purchase a game with the mechanic present and wish to see it produced further.

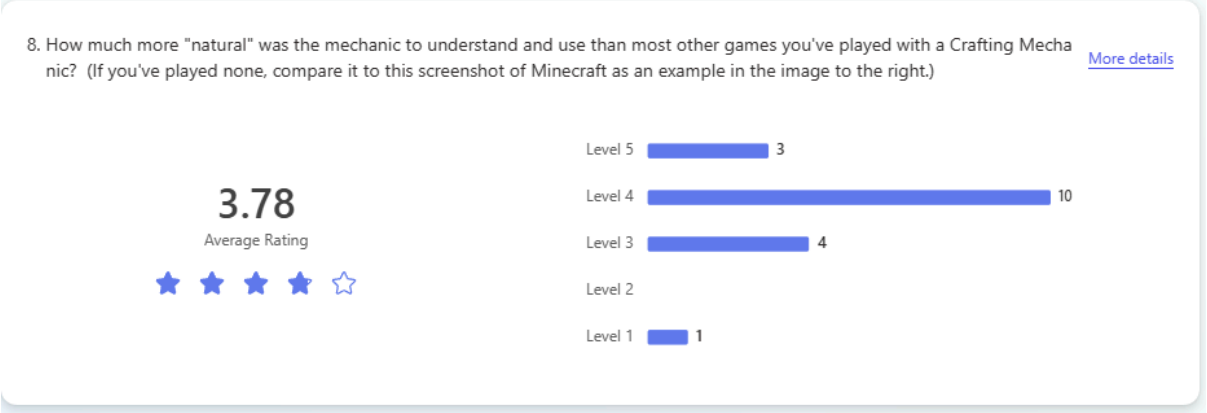


Figure 20: On a scale of 1 being worse and 5 being better, players stated the doodle-based crafting system felt more natural to use than a menu-based crafting system, such as Minecraft.

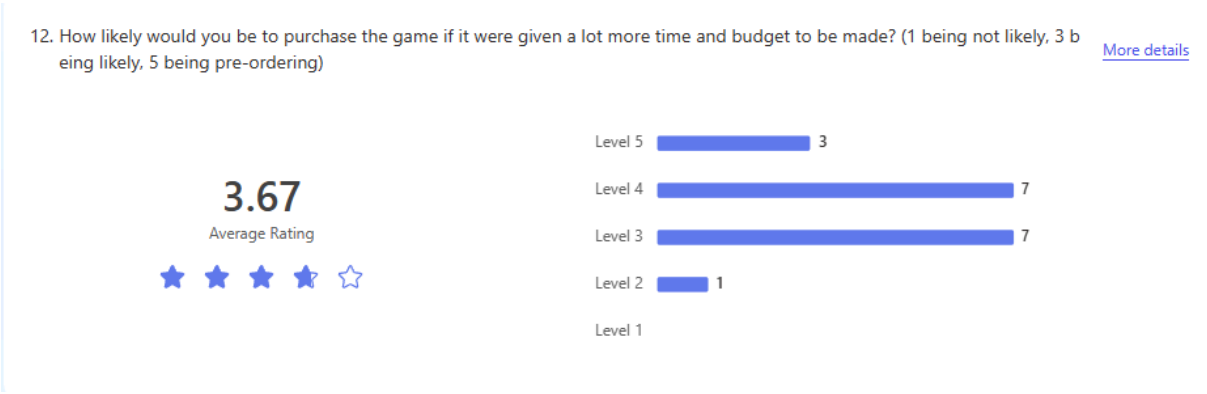


Figure 21: On a scale of 1 being unlikely, 3 being likely, and 5 being actively pre-ordering, most players stated they would be willing to purchase a fully-fledged game with the prototype's crafting mechanic.

Lastly, the exact details of what made the level so mixed on reviews was requested via final dialogue boxes. This included a question for any bugs discovered during gameplay and a question for opinions on the game's design. (See Appendix A) Many respondents reported trouble with the neural network detecting a doodle of a Rifle. Another large chunk of respondents stated difficulty with the level's platforming and enemy respawning, which reduced their enjoyment of the experience. A lot of players also mentioned confusion on the hints and use cases for the Colour Pools in the level. Players found themselves to be stuck from time to time trying to get the

mechanics of the game to work, which is seemingly caused by bad implementation of visuals to indicate things that are breakable and not. Another issue arose when players attempted to combine the action of two items, such as the Parachute and Tornado, as they reported they might only ever end up using one due to lack of “satisfaction”.

Discussion and Analysis

[RQ1] Measurement and Push for Player's Creativity

The main issue with the final product seemed to be related to visuals and indication with the game itself. Additionally, there were failures in getting the items to have multiple use cases. Multiple parts of the level only had one item as a solution for some objective, such as the Tornado to get to a taller platform, or the Key to unlock a single red Door. For parts that did require multiple items, it required both items to be present in the player's inventory. The only room that had any leniency to allow for a proper freedom of choice was the final room.

This regard of pushing the player's creativity seemingly did not achieve its objective to its fullest. Measurements were indeed taken, and the level design did have later rooms with rather ill-defined tasks, but not enough tutorial was put in to ease the player into later experimentation. However, the data suggests an unexpected finding, that players feel creative when using the doodle-based crafting system. It seems to not be related to the number of items they were offered as feedback clearly stated some felt there were too few items to work with, along with the previously mentioned game design flaw of the prototype having such a limited number of items.

Looking back onto the research proposed by Riley (Riley, 2019), it suggests that the feeling of creativity even without experimenting with many item types might have emanated from being able to express one's desire simply by drawing the item. This is further evidenced by feedback that claimed they enjoyed having to figure out what the game allowed as drawings, which can interconnect with the natural human nature to understand signs as representing objects (Kothawade, et al., 2020). This may have had the results for the question that compared the mechanic to another crafting mechanic game, such as Minecraft, eventually giving reports that the doodle-based crafting system felt more natural.

Testing did have its flaws. Firstly, the selection of respondents for the questionnaire were well known to the developer, which may have resulted in a friendship bias. Additionally, asking specifically "how natural" or "how creative" may have been too vague and resulted in tainted data, however for this experiment it could be argued that opinionated data was necessary, as doodles are an ancient human construct (Davidson, et al., 1989) and therefore truly computing it may be removing elements from the data that make up what it is.

[RQ2] Neural Network Accuracy and Performance

The neural network had issues recognizing a Rifle, which can be broken down into one main reason that was clearer after internal evaluation and reflection. The dataset may have provided doodles that were too simple or lacked a detail that could properly distinguish it from another kind of item. Such as the below figure, some of the dataset could easily be mistaken for a flashlight by a human as well. Unlike the flashlight's data, which was heavily weighted in having "three dashed light beams of lines", the Rifle rarely had any distinguishing data, when it probably should have had scopes, magazines, or the like.

Figure 22: Provided by a hand-picked pass on Google's Quick, Draw! Dataset; on the left is a rifle, and on the right is a flashlight, which looks too similar.

Beyond the flaw with recognizing Rifles, the network actually had reasonable strides toward success in achieving the set objective, which was to reduce the performance overhead as much as possible whilst maintaining accuracy. The final average reported value of 68.2% was quite high, as previous implementations during development were much lower. However it is not perfect, and the network certainly could have been pushed further.

When compared to what the industry expects the results to be when proving the accuracy of a simple inference neural network, the network seems much more like a simple test. Usually a percentage of around 90 or more percent seems to be more tolerable as these make up the majority of accuracy testing results of all experimental networks in Atienza's book of tutorials for Python Keras (Atienza, 2020). What is unknown is if the accuracy issue was due to the depth of the network as well, for not enough study was carried out on how to be careful of overfitting or gradient explosion.

One flaw that surely existed with this testing was that there was no attempt at measuring the framerate and CPU usage impact of the neural network's execution. Even though no issues with performance were reported in the feedback data, measuring it with quantitative data would have been preferred. Besides that, even though friendship bias did exist in the questionnaire and gameplay testing, the factually-based buttons to report a false positive from the neural network may have been confusing. Some players had mistaken it for a simple erase button resulting in much lower accuracy data which may have brought down the final network average accuracy.

[RQ3] Prototype's Game Design and Viability for Audience

The flaws in the level's enjoyment mainly stemmed from design choices on platforming, enemies, and puzzle design. Not enough hints were put into some segments of the level to properly show the intended solutions. The annoyance with the enemies resulted in a major distraction from the more alluring aspect of the gameplay; that being to create items and use them in the world as the player pleases. And with the platforming, it ended up catering to a genre of higher skill players, as happened due to the developer's own experience whilst playtesting. Affordances like these were missing ultimately because the level's design was done from scratch in the game engine, rather than creating a grid or other type of draft with proper measurements.

Even with such a flawed level design, the side of the objective to prove viability as a small independent game studio's project did provide enough to objectively gather needed data. Many players reported they would be likely to purchase a game that uses this crafting mechanic. For the objective of creating an enjoyable level, the main aspect holding it back was game mechanics providing punishment that resulted in players feeling like what they've learned or explored was useless, such as the Tornado not reaching high enough to a platform that briefly seemed available, or an enemy respawning shortly after being killed with a Rifle.

In terms of the allure of the game to the audience, a large amount of the respondents claimed the mechanic reminded themselves of Scribblenauts, which is indeed comparatively the most alike game with a crafting mechanic related to using the hand to draw. What this indicates is a form of nostalgia for such a game mechanic (Kashtan, 2014), which can truly be attractive to buyers looking for games from the indie space. (Lipkin, 2013)

Testing therefore could have been done better in many ways. On top of the friendship bias, and besides the flawed game design, a better valued testing group would have had a much larger number of participants. There was also a bias in picking players who specifically have played games with a crafting mechanic before, which does not consider users who have not encountered such a mechanic. Additionally, measurement of skill level on the subgenre selected by the prototype, such as platforming, and combat, should have been taken as it would have been interesting to compare their previous experience with their gameplay performance in the prototype.

Conclusion

At the start of the research into the tools available for a doodle recognition mechanic, it was discovered that the broad range of them being present in a still rather early age for computing neural networks results in a lot of those tools being lower level or more difficult to learn. Whilst Python Keras did indeed turn out to be one of the simplest languages to use (Vasilev, et al., 2019) especially with its wide range of capabilities (Atienza, 2020), the integration into Unreal Engine was quite limiting due to Epic's decisions to drop OrtCPU (which is short for ONNX RunTime CPU) support in 5.4. By having trouble with the Neural Network Engine on versions post 5.3, it ends up that Unreal Engine as it is now, which was version 5.5 at the time of writing, is not the optimal solution for the crafting mechanic and may pose difficulties in development when programming.

Whilst there were some flaws in the final implementation of the artefact, such as the failure to recognize Rifle doodles, there was still a lot to be said about how it influenced the game programmatically. Utilizing the ASyncTask multithreading helper function proved to be a stable, memory safe way of executing the neural network with no noticeable framerate overhead, whilst it provided real and reasonable data back to the game to spawn items based on the neural network's inference. The mechanic, therefore, is proven to be functional in accordance with its design. However, this artefact did not achieve the fullness of its aim to provide a neural network with enough accuracy, as it was only measured to be around 68.2%, which proved it had noticeable problems beyond specific written feedback from the play testers.

In terms of the game's design, an interesting point to talk about would be the item selection. It is conclusive that the number of items provided to the player did not provide enough to stoke creativity within their gameplay, even if the latter parts of the prototype's level had more and more open-ended tasks. The high amount of uses on the Tornado and the Rifle proved that these items felt superior to all the others which resulted in no necessary need to use the other items unless a puzzle in the level specifically called for it. Feedback even stated that players wished there were plenty more items to choose from as it would help them experiment much further beyond the linear scope of the level.

However, there was an unforeseen product of the mechanic that showed up in the results, which was initially irrelevant. Play testers reported they felt quite creative when using the mechanic, which as earlier suggested, could be the result of using the hand like a pen to draw what they desire. Even the comparison to other menu-based crafting systems such as Minecraft suggested they felt this mechanic was superior when aiming to feel more natural to human behaviour. This concludes that this mechanic must be much easier for even newcomers to the genre to understand and even proves it reduces the difficulty of planning one's strategy to collect resources, as the Ink Pots in the prototype were designed for.

When asked about their enjoyment of the level it proved the artefact had a significant failure to create an enjoyable level. However, it was also proven that this was due to the nature of mechanics introduced into the game that were not the crafting mechanic. Instead, these issues were shown by the lack of affordance in the level and enemy designs. Additionally, there was proven evidence that many players felt a sense of nostalgia for the game Scribblenauts, suggesting seeing the mechanic's return as a doodle-based crafting mechanic would be in the interest of customers willing to purchase the game. It is concluded that this also contributed to the enjoyment of the game overall, even if the level design had lacked quality, the crafting mechanic made up for it by being interesting to the players.

To sum it up, the project was programmatically functional, and enough skills were learned to get the game stable enough that such a negligible performance hit proves the mechanic has a wide scope of hardware it could be executed on. This alone widens the possible target user base for the independent studio working on the title. What the mechanic needs to be careful with is what sort of game and scope it is placed beside. This artefact's results concludes that a game with a very small scope is not feasible, as it would restrict creativity to a point that the mechanic starts to lose its potential. It also is conclusive that having a difficult level and difficult combat system, regardless of what crafting mechanic is implemented, results in a loss of enjoyment.

The project approaches its final question of if a small independent games studio could develop a game with this mechanic. In short, the answer is yes. The tools required are all low cost, and if carefully picked, will be low difficulty to use as well. If the studio is willing to take the time to be deeply considerate of player affordance and to increase the accuracy of the neural network solution to the highest degree possible, such a game absolutely could exist. There is even highly suggested demand for the mechanic as 17 out of 18 play testers stated they would be likely or more than eager to purchase such a game. It is concluded that this desire is both born out of the nostalgia for Scribblenauts, and the allure of how the mechanic feels more natural to use than a menu of squares to click.

Recommendations

The small studio should first consider the toolset and alternatives for both performance and ease of use. Whilst Python does give promising results, it may be worthwhile to look at other languages as Python will always be slower than a language that compiles directly to machine code. Another tool to look at is the engine of choice. Due to the conclusion that Unreal Engine 5.5 does not make sense for the project, Unity's Barracuda neural network suite may be the better option. From a glance it seems promising that its syntax, and in-engine training toolkit, makes a better out of the box experience for creating this mechanic.

A short letter should be written to Epic explaining the significance of their unprecedented deprecation of NNERuntimeOrtCPU. Whilst their argument to support IREE is justifiable, due to it translating the network into machine code, what it does not do kindly is keep support for lighter weight neural network projects such as this one, and the ease of use for developers to export the network into a packaged build of a game. An argument could be made to reinstate the package, even if the control over how many threads it uses has yet to be completed. Forcing developers to work on an older version of the engine to push the industry's products further seems counterproductive.

More game designs should attempt to implement a similar mechanic to see how far a neural network of this fashion could get pushed. Games with exotic hardware such as VR, AR, even, could push the mould of this mechanic to its limits. Can it be used to recognize advanced gestures in 6 degrees of freedom? Can it be used to recognize a 3d sculpture made in the game by the player? There are several aspects of how a neural network can be used to recognize shapes that represent something the human desires.

References

Ahmed, D. S., Ibrahim, R. S., Hashim, F. A. & Hussein, N. M., 2023. Keras Deep Learning Package in Python: A Review. *European Journal of Interdisciplinary Research and Development,* Volume 19, pp. 24-29.

Al-Barhamtoshy, H. M., Jambi, K. M., Rashwan, M. A. & Abdou, S. M., 2023. An Arabic Manuscript Regions Detection, Recognition and Its Applications for OCRing. *Transactions on Asian and Low-Resource Language Information Processing,* 22(1), pp. 1-28.

Atienza, R., 2020. *Advanced Deep Learning with TensorFlow 2 and Keras: Apply DL, GANs, VAEs, Deep RL, Unsupervised Learning, Object Detection and Segmentation, and More.* Birmingham: Packt Publishing Ltd..

Bannister, D. & Fransella, F., 1980. *Inquiring Man: The Psychology of Personal Constructs.* 2nd ed. London: Routledge.

Barton, M. & Stacks, S., 2008. *Dungeons and Desktops: The History of Computer Role-Playing Games.* 2e ed. New York: A K Peters/CRC Press.

Cropley, A., 2006. In Praise of Convergent Thinking. *Creativity Research Journal,* 18(3), pp. 391-404.

Davidson, I. et al., 1989. The Archaeology of Perception: Traces of Depiction and Language. *Current Anthropology,* 30(2), pp. 125-155.

Davidson, J. E. & Sternberg, R. J., 2003. *The Psychology of Problem Solving.* Cambrige: Cambrige University Press.

Díaza, D. M., Saorín, J. L. & Carbonell-Carrera, C., 2020. Minecraft: Three-Dimensional Construction Workshop for Improvement of Creativity. *Technology, Pedagogy and Education,* 29(5), pp. 665-678.

Epic Games Inc., 2024. *We are Updating Unreal Engine, Twinmotion, and RealityCapture Pricing in Late April.* [Online]
Available at:
https://www.unrealengine.com/en-US/blog/we-are-updating-unreal-engine-twinmotion-and-realitycapture-pricing-in-late-april
[Accessed 16th December 2024].

Fan, Y., Lane, C. H. & Delialioğlu, Ö., 2022. Open-ended tasks promote creativity in Minecraft. *Educational Technology & Society,* 25(2), pp. 105-116.

Fernandez-Fernandez, R., Victores, J. G., Estevez, D. & Balaguer, C., 2019. Quick, Stat!: A Statistical Analysis of the Quick, Draw! Dataset. *arXiv preprint arXiv: 1907.06417.*

Hassam, G. M., Hussein, N. M. & Mohialden, Y. M., 2023. Python TCP/IP Libraries: A Review. *International Journal Papier Advance and Scientific Review,* 4(2), pp. 10-15.

Hoffman, D. L. et al., 2018. U-Th Dating of Carbonate Crusts Reveals Neanderthal Origin of Iberian Cave Art. *Science,* 359(6378), pp. 912-915.

Kashtan, A., 2014. Forward to the Past: Nostalgic Fantasies of Handwriting in Video Games. In: *Writing Yourself Into the World; Fantasies of Handwriting and Computer Graphics.* Ann Arbor: ProQuest LLC., pp. 373-452.

Kaufman, J. C. & Baer, J., 2012. Beyond New And Appropriate: Who Decides What Is Creative?. *Creativity Research Journal,* 24(1), pp. 83-91.

Kitchner, K. S., 1983. Cognition, Metacognition, and Epistemic Cognition: A Three-Level Model of Cognitive Processing. *Human Development,* 26(4), pp. 222-232.

Kothawade, D. et al., 2020. Recognition of Labels for Hand Drawn Images. *International Research Journal of Engineering and Technology,* 7(5), pp. 1819-1823.

Leonard, T., 2007. *Dragged Kicking and Screaming: Source Multicore.* San Francisco, Game Developers Conference.

Lipkin, N., 2013. Examining Indie's Independence: The meaning of "Indie" Games, the Politics of Production, and Mainstream Cooptation. *Loading...,* 7(11), pp. 8-24.

Lubart, T. I., 2001. Models of the Creative Process: Past, Present and Future. *Creativity Research Journal,* 13(3-4), pp. 295-308.

Maslen, M. & Southern, J., 2014. *Drawing Projects: An Exploration of the Language of Writing.* London: Black Dog Publishing.

McShea, T., 2011. *Super Scribblenauts Review for DS.* [Online]
Available at: https://www.gamespot.com/reviews/super-scribblenauts-review/1900-6281729/
[Accessed 10th December 2024].

Myers, D., 2003. The Nature of Computer Games: Play As Semiosis. *Digital Formations,* Volume 16.

Noble, W. & Davidson, I., 1996. *Human Evolution, Language and Mind: A Psychological and Archaeological Inquiry.* Cambridge: CUP Archive.

Reiser, B. J. & Tabak, I., 2014. Scaffolding. In: R. K. Sawyer, ed. *The Cambridge Handbook of the Learning Sciences, Second Edition.* Cambridge: Cambridge University Press, pp. 44-62.

Riley, H., 2019. Drawing As a Language: The Systemic-Functional Semiotic Argument. *Journal of Visual Art Practice,* 18(2), pp. 132-144.

Rosenblatt, F., 1958. The Perceptron: A Probablistic Model of Information Storage and Organization in the Brain. *Psychological Review,* 65(6), pp. 386-408.

Salmon, P., 1978. Doing Psychological Research. In: F. Fransella, ed. *Personal Construct Psychology.* London: Academic Press, pp. 37-45.

Sawyer, R. K. & Hendriksen, D., 2024. *Explaining creativity: The Science of Human Innovation.* Third ed. Oxford: Oxford University Press.

Schmandt-Besserat, D., 2014. *The Evolution of Writing.* [Online]
Available at: https://sites.utexas.edu/dsb/tokens/the-evolution-of-writing/
[Accessed 12 12 2024].

Sharma, S., Princy, Bhatia, K. & Sharma, R., 2023. A Study on Image Categorization Techniques. *International Journal of Multidisciplinary Research in Science, Engineering and Technology,* 6(5), pp. 1147-1152.

Sorrenti, V., 2023. *Image Classification in the Browser: A Performance Assessment,* Turin: Politecnico di Torino.

Tanenbaum, K. & Tanenbaum, T., 2009. *Commitment to Meaning: A Reframing of Agency in Games,* Irvine: Digital Arts and Culture.

Tappert, C. C., Suen, C. Y. & Wakahara, T., 1990. The State of the Art in Online Handwriting Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 12(8), pp. 787-808.

Tringali, J. M., 2009. Postmortem: Behind the Scenes of Scribblenauts. *Game Developer,* Issue Nov 2009, pp. 22-31.

Vasilev, I. et al., 2019. *Python Deep Learning: Exploring Deep Learning Techniques and Neural Network Architectures with PyTorch, Keras and TensorFlow.* Birmingham: Packt Publishing Ltd..

Zhang, J. & Dacheng, T., 2020. Empowering Things with Intelligence: A Survey of the Progress, Challenges and Opportunities in Artificial Intelligence of Things. *IEEE Internet of Things Journal,* 8(10), pp. 7789-7817.

Appendices

Appendix A

This is the raw data of open-ended questions in the questionnaire regarding explaining bugs encountered and opinions with the level design.

Token ID	Describe any bugs you encountered whilst playing
1	None experienced in this build specifically, save for minor collision issues with the gargoyles. There were also instances of missed crouch jumps, where they would have succeeded in prior builds, though this may have been intended.
2	it was difficult to have it recognize rifle
3	Wonky Platforming, sometimes when creating craft, nothing happens and the doodle closes
4	none
5	I did not encounter any bugs
6	Crashed on first older attempt after Right clicking canvas (Was fixed on current build). painting going off the canvas meant that when mouse hovered back onto canvas, it was still painting without input.
7	I did notice the key disappeared when encountering the red door for the first time when I left clicked instead of right, not sure if that was meant to happen
9	1. The room with the first pressure plate, specifically asking for an anvil, bugged out after I threw the anvil onto the pressure plate, as the anvil bounced off and onto the level below. I tried to place another anvil more gently onto it next time, but it didn't activate. Restarting the game made it work. 2. This is less of a bug and more of a note, the fireball imp enemy sometimes avoided what seemed like a direct shot with the rifle, and I'm not sure if it's a hitbox issue or so.
10	
11	Pausing didn't stop enemies from attacking me; when I made the torch yellow whilst turned on, the light surface area didn't get bigger and I had to turn it off and on again for it to the effect to trigger.
12	Rifle is incredibly hard to draw as nothing logical turns into a rifle, made it very difficult on the star tower as I couldn't draw it and kill the enemies shooting at me half way up. Also threw the star onto the button with LMB and it didn't
14	
15	N/A
16	I dropped a gun on one of the pressure points, and the gun bounced off, disappearing from the world. The plate would then not allow me to trigger it again

	with the correct item, making it impossible for me to continue
17	no bugs
18	The doodle seemed to tend towards "key" as an answer to most in-progress doodles. At one point it classified a single circle as a key. It also really struggled with rifles, requiring a very specific doodle to make that work, often mistaking for flashlights and keys.
20	none that i could see
21	

Token ID	Any other opinions about the game or its mechanic design?
1	The drawing mechanics pretty neat, reminds me of scribblenauts (Even though I haven't played an entry of that series in years). It is rather hit or miss if the game will guess correctly what you're drawing (That might have been why scribblenauts chose a text-based design over and art-one, but still unique nonetheless). Enemies were challenging to where you couldn't really side-step them (Interestingly backwards movement is more lethal then just forwards/strafing), but trivial enough to know how they'd attack within a few moments.
2	difficult and annoying to dodge the goblins while platforming
3	The enemies in this game are not necessary or fun, They are extremely frustrating to deal with, and just make a cool experience tedious
4	Crafting mechanic became intuitive after a while. I hate platform jumping puzzles.
5	The level design needs work, and I think the spray paints should add the items to the discovery book. It's hard to think of the exact items while playing
6	To be organic in it's playthrough and become truly "Open-ended", it would need substantial amount more things to draw with more types of route around the obstacles to account for varied approaches from players. Makes a very fun and engagingly challenging game. The level with the goblins in the dark was the only annoying part where I could only hold the weapon or light individually and couldn't see where I was hitting and ended up soft-locked until I reverted back to the last stage to a canvas to recover items.
7	I was a little confused on what the colours do but I might need to run through it again

9	1. Considering how much jumping and platforming there is, it may be handy to have a double jump that allows you to correct your trajectory, or slightly more control over where you're going in mid air, like with the parachute, but maybe not as much. 2. If there could be an undo or erase button (maybe on the right mouse button), that would be amazing. Sometimes one small line or twitch of the mouse took the result from what I was about to craft to the canvas saying "???". 3. I understand the instructions written on walls were meant to be readable without the use of any specific language (with a few exceptions), and while most made sense, some were very vague, like the messages on the first few colour blobs. I wasn't sure what the fist or "add time" hourglass meant, and the yellow dotted circle made me think it wanted me to craft scissors. 4. Crouch jumping is nice. Thank you for including it.
10	I felt because of the limited set of items my creativity was restricted. There were times I would try to craft a gun or bow for ranged combat but just couldn't. I also tried to combine the tornado and parachute to go really high and cover a further distance while it did work it didn't feel as satisfying as I think it should. I got attacked in the darker section losing my flashlight and making it nearly impossible to continue.
11	Probably a skill issue but I really struggled with drawing a rifle - it likely made up most of the wrong guesses. Other than that, I really enjoyed the game - it felt like time flew by playing it. Big fan of the design of the first 5 triggers area in particular. (hopefully you know what I mean by triggers?)
12	Enemies way too annoying, parkour was too punishing, almost ragequit on the star tower because I fell to the bottom and my wrist hurt from so many small movements (I think it triggered some minor carpal tunnel?)
14	
15	Way to hard
16	The mechanic itself is amazing. A unique idea and definitely a good use for AI, that isn't just using AI for the sake of using AI. With more time and more money, this game could become huge. Even bigger than other games of the same style
17	refreshing
18	I think the doodle mechanic is incredibly intuitive, and given the prompting throughout the levels (which I definitely didn't miss) there wasn't too much question as to what I needed to draw to complete a level. The tornado was a pleasure to work with, I think possibly the most satisfying of the bunch. I think given a more explicit setting (medieval or modern etc.) it may have been more clear what items are available. For example in the dark section, I initially attempted to draw a torch (as in a stick with flame on it) instead of a flashlight.

20	encourage using the mechanics of the game rather than using platforming prompts. also segregate the action and puzzling mechanics somewhat
21	Colours being introduced later in a longer game would make the mechanics easier to learn. Non-respawning enemies would helpin certain sections where you can get stunlocked without a weapon and amy have to backtrack.

Appendix B

Raw Python generated output of 15 Results JSONs, starting with Network Accuracy and the total average, followed by the number of times each item was drawn.

```
Reported Accuracy List:
1
0.686274528503418
0.3177570104598999
0.875
0.28205129504203796
0.4776119291782379
0.3199999928474426
0.5888888835906982
0.9607843160629272
0.8947368264198303
0.859649121761322
0.6170212626457214
0.8214285969734192
0.6056337952613831
0.925000011920929
Total Network Accuracy:
0.6821225047111511
Items Crafted:
tornado
207
rifle
115
parachute
60
flashlight
43
hammer
33
anvil
26
star
42
sword
30
Key
25
lighter
13
```

Figure 23: Raw output from the console out of Python's print, from code that brought the JSON data together mathematically.

Appendix C

A large batch of screenshots showing what the gameplay looked like in the result of the prototype.

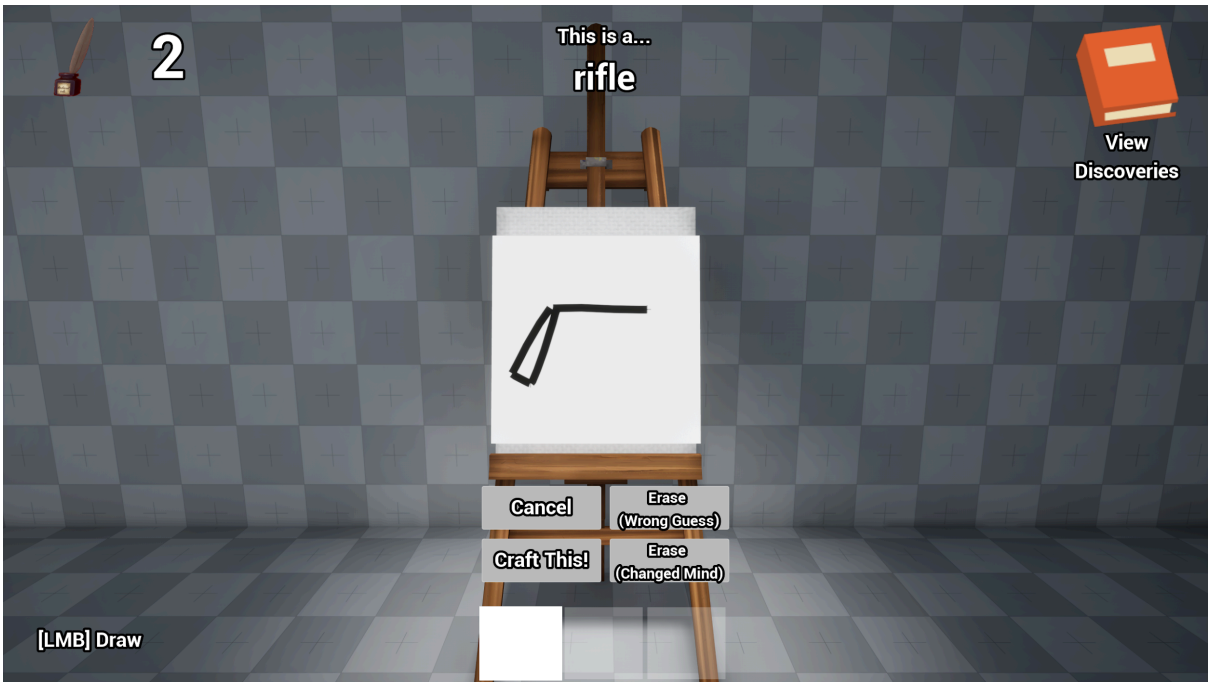


Figure 24: The UI and the Easel with a drawing in progress, showing the neural network recognizes it as a Rifle.

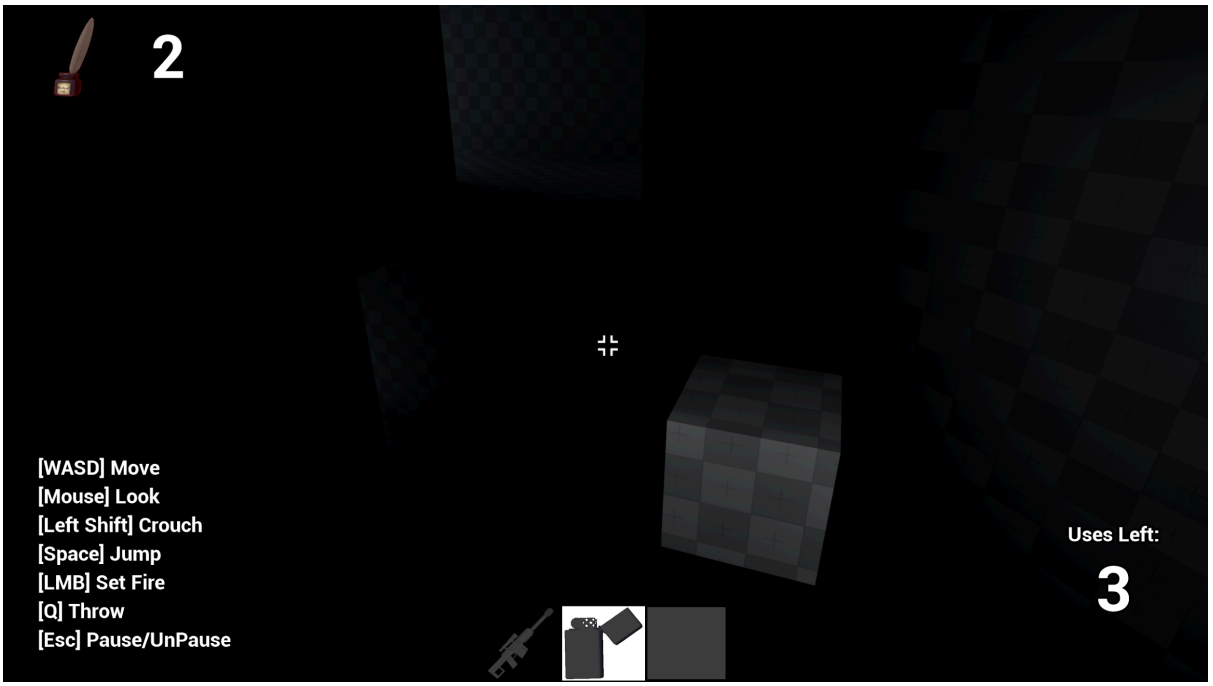


Figure 25: The player using a lighter item to light up darkness to reveal platforms

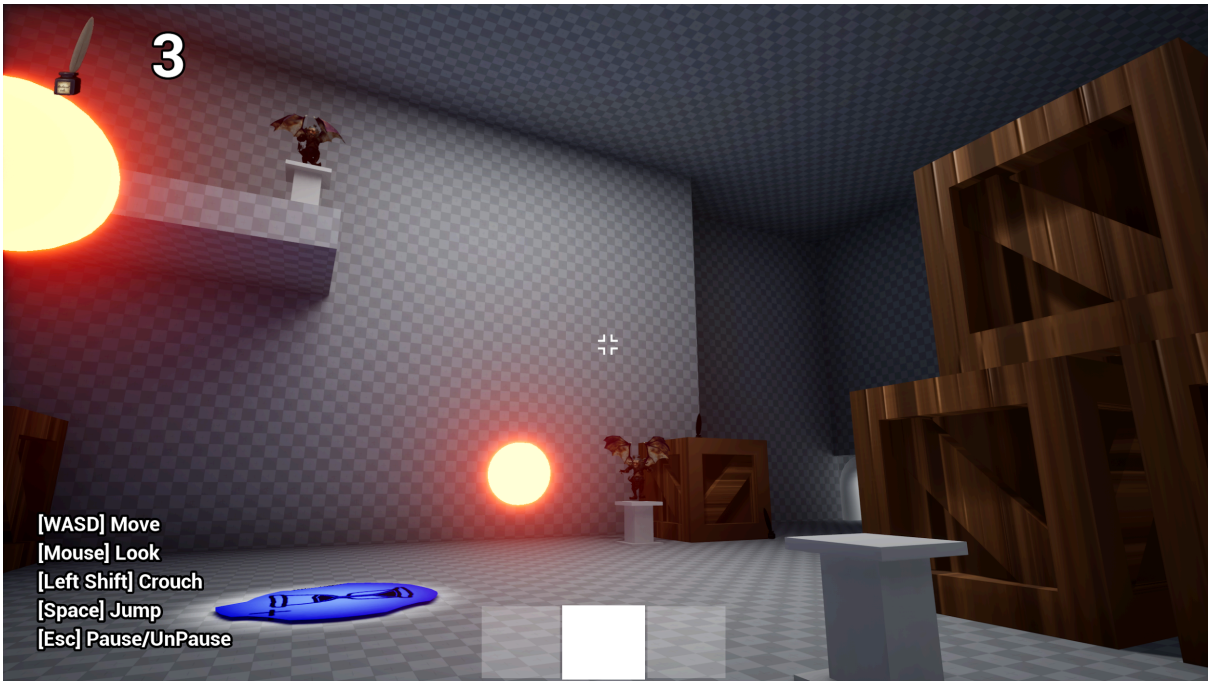


Figure 26: A combat encounter with a colour pool as a reward

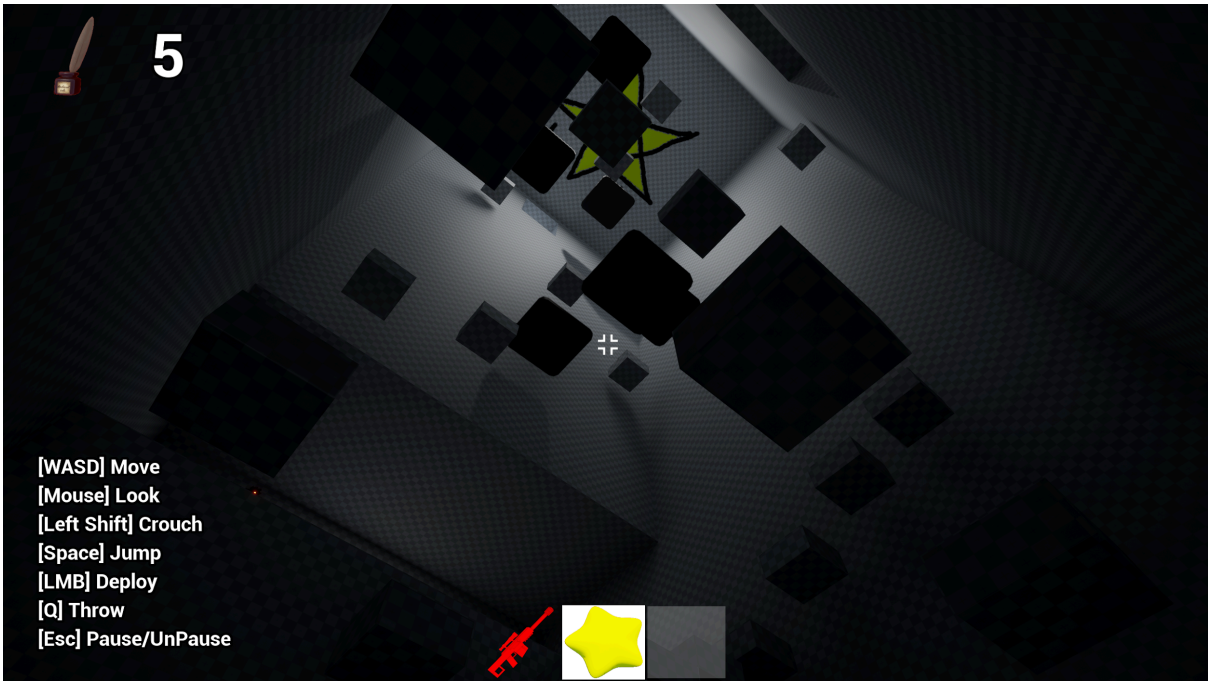


Figure 27: The final room with as much ill definition to its task as possible.

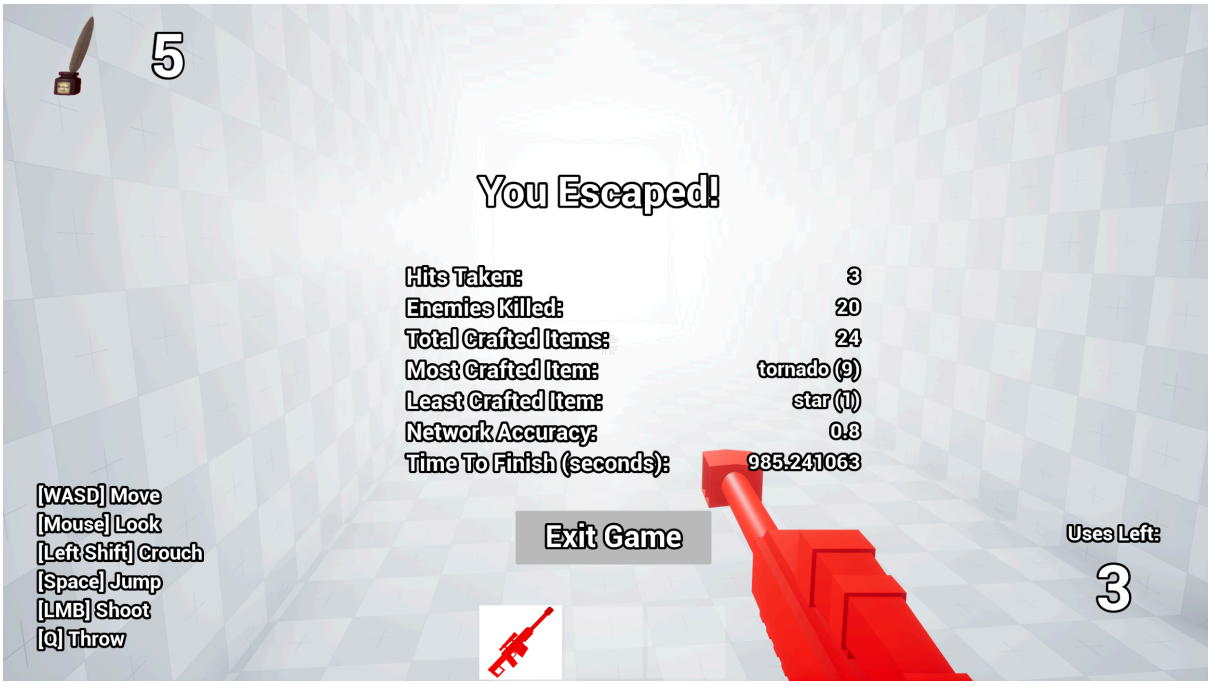


Figure 28: The game results screen