DUPLETOR

PROVIDING EXISTENTIAL PERSPECTIVES TO VIRTUAL GAME ENVIRONMENTS

CODESTORY - A SELF-CONSCIOUS UNIVERSE

São Paulo
2019

DUPLETOR

PROVIDING EXISTENTIAL PERSPECTIVES TO VIRTUAL GAME ENVIRONMENTS

CODESTORY - A SELF-CONSCIOUS UNIVERSE

Undergraduated thesis presented to Polytechnique School of Engineering from University of São Paulo for the Title of Bachelor in Computer Engineering.

São Paulo
2019

DUPLETOR

PROVIDING EXISTENTIAL PERSPECTIVES TO VIRTUAL GAME ENVIRONMENTS

CODESTORY - A SELF-CONSCIOUS UNIVERSE

Undergraduated thesis presented to Polytechnique School of Engineering from University of São Paulo for the Title of Bachelor in Computer Engineering.

Area of Application:

Game Design, Artificial Consciousness

Professor adviser:

Prof. Dr. Jorge Luis Risco Becerra

São Paulo
2019

To all the people who are yet to become true self-conscious beings in the essence of the mind of millions I dedicate this pioneer project.

ACKNOWLEDGMENTS

Firstly, I thank to every single teacher responsible for my personal development across my line of existence, for the knowledge provided.

Mainly to my Profesor adviser, Prof. Dr. Jorge Luis Risco Becerra, for accompanying my progress through the development of this thesis.

In special to Prof. Dr. Ricardo Nakamura and Prof. Dr. Romero Tori, who accompanied the project during its foundation and its latest builds.

I thank to robotics team ThundeRatz for my development as an engineer.

I thank to every person who helped this project reach its current state.

"If there's anything you want to learn, don't be stopped by thinking it'll be too hard! Try it! Game-making, music, art, language, anything!"

-- Toby Fox

ABSTRACT

This paper will describe the demonstration project of an exploration-based humanistic role-play game, in which the characters question in high details about their own existence, welcoming the player to their new body. Just as notably as the player learns about the world, the characters of this world get knowledge about themselves and the player, through the chosen actions and yielded consequences. The main objective of this project is to demonstrate original relations between concepts of consciousness and perspective. Being unlimited in possibilities, the concept will study existence from the beginning. Consciousness is the main concept to be widely explored in the project, of the philosophical concepts recently idealized by engineering paradigms, determinedly overrun by the concept of Intelligence, creating a brand new environment for the study of emergent properties in thinking machines.

Keywords: – Existence, Perspective, Emergence, Consciousness.

ABSTRACT

Este paper descreverá um projeto de demonstração de um RPG humanista orientado a exploração, no qual os personagens questionam sobre a própria existência em grande detalhe, trazendo o jogador para um novo corpo. Tão notavelmente quanto o jogador aprende sobre o mundo, os personagens deste mundo adquirem conhecimento sobre si próprios e o jogador, através das ações escolhidas e consequências acarretadas. O principal objetivo deste projeto é demonstrar relações originais entre os conceitos de consciência e perspectiva. Ilimitado em possibilidades, o conceito estudará a existência desde o início. Consciência é o principal conceito a se explorar amplamente neste projeto, de todos os conceitos filosóficos idealizados recentemente por paradigmas de engenharia, determinadamente sobrescrito pelo conceito de Inteligência; criando um ambiente novo de estudo sobre as propriedades emergentes de máquinas pensadoras.

Keywords: – Existência, Perspectiva, Emergência, Consciência.

LIST OF FIGURES

CONTENTS

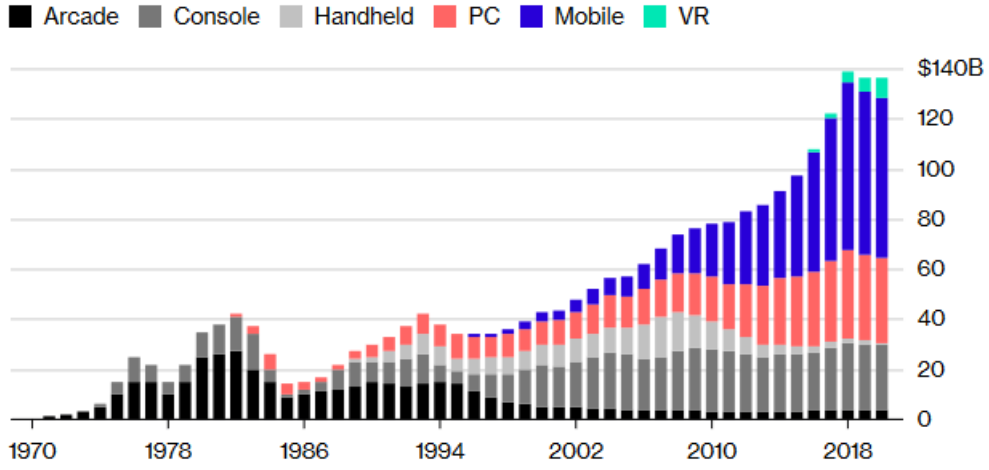
1 INTRODUCTION

Within the past two decades, the Video Game Industry has continuously grown tremendously, becoming one of the most iconic industries for entertaining technological application.[1]

While the oldest video game structures depleted, people started affording personal options for gaming, and the market divided itself well into specialist consoles, PCs and mobiles by 2010, when the technology of mobiles became powerful enough for popular gaming.[1]

Through the last decade, handhelds were reduced and mainly dominated by Nintendo, consoles stabilized as a mainstream technology, pc games became more popular, and the popularity of mobiles exploded through brand new concepts to the market.[1] Recently, an increase on the tendencies of VR and saturation of Mobiles arrived, as observed in the following graph.

Figure 1: Development of gaming industry since 1970



bnnbloomberg.ca

The purpose for development has been widely varied across technological distribution platforms, and built acknowledging the market tendencies through all development stages as an primordial factor to better suit its goals. [2] While the development for mainstream media (2000-2010) runs either around the concept of specialization or generalization, respectively

for Consoles and PCs, the development for mobiles exploded through the popularization of gaming around the concept of hypercasuals, specially designed for mass data collection. [3]

Across the individual development between PCs and Consoles, each divided itself in philosophical regiments for developers, mainly known as the fight between Big Company games and Indie games. [3]

While Corporation games have a wild financial capacity of employing the world's best staff and tools for development, they already have a big expected market that affords to buy their products, which means their product must be safe enough to not disappoint guaranteed sections of society. As a consequence, it is very common for companies to avoid major changes in successful titles, specially in mechanical perspective, and therefore the major technological changes are applied into Animation, complexity (diverging from the hypercasual concept in Mobiles), Artificial Intelligence and World generation, and countering the safety usually has extreme consequences.[2] One recent example of a Big Studio game in this condition with Overwhelmingly negative repercussion is Artifact, from Valve, which paid itself on the Hype and disappointed the majority of playerbase, when a company famous for huge titles like Portal and Dota 2 makes a bad game. Even though it became a complete failure, it lead to profit, because the company had too much of a playerbase to lose money. One recent example of a game that changed its design intensely and got an Overwhelmingly positive response was God of War, by Santa Monica Studio, which remastered every bit of gameplay from the franchise in the last God of War game (2019) surprising the audience with huge success.

In contrast to the safe development of guaranteed profit from Big Companies, Indie developers arrive with lesser technologies and hiring capacities, but with the possibility for development directed by, as the name suggests, independent ideas for free development[2], capable of developing the craziest ideas and earn prestige through creativity and sheer dedication. Even though this sector is oversaturated by poor quality, copied mechanics and bad design, it is where new ideas can shine, specially the ones who bring philosophical questions about real life.[3][8]

1.1 Problem

This project is an Indie exploration-oriented Role-Play Game, intended to demonstrate the consequences of conscious minds in a virtual world. A self-conscious Universe with true desire of discovering the true nature, the reasons, requirements and objectives of Existence. Each individual Non-Player Character lives along with their own personal existential perspective,

living their lives around their concept of justification for existence, and reconsidering their life choices according to the level of chaos around them.

These properties will require proper study and application of Etymological concept of Consciousness, advanced control over the development capacity, a solid reactive structure for identification of incoherence and innovative uses for Arbitrary Screen Control technologies.

1.1.1 Inspiration and Determination

This project has been inspired by Undertale, by Toby Fox, on a philosophical view, though it will explore the existentialist aspects of the self-awareness of a game, instead of the deterministic ones as Undertale does, providing new possibilities and even more intense questions about the perspective of what we are. Also, it was technologically inspired by an Internet Gem from the Flash Era, Animation vs. Animator, in which a stickman fights for their life causing problems and damage to the computer of the animator who designed it.

1.2 Objectives

This project's intention is to show to the world the perspective of nine different ways to justify existence through eight civilizations, and abuse the most varied properties of computing to create very exotic effects and properties, and so it has to provide the player with deep characters, capable of analyzing the world according to their own perspective, and show the player why they believe in what they do, how they react to chaos and what they think about their existence.

In order to completely question society through every single possibility according to Existentialism, nine different perspectives will be used, all incomplete by themselves, but complete as a whole and capable of justifying Existence. The model generated by this paper will be the first Existential line of thought, through the first civilization: A Naturalistic view of Existence. Under the perspective of the first civilization: "Existence is real, for we exist and something is real. The most trustful source of knowledge is Existence itself through empiricism, and so we justify our own by learning the true behavior of things."

1.3 Methodology

Across the development of this project, a systematic approach for engineering design, aligned to all of the the researched subjects related to game design and consciousness will be required, as not even the entire functional structure can be considered a definitive answer to how the development of the project system should be lead in order for it to be considered an engineering project. The provided techniques and classifications used will then actually determine a system that is called a "game development software engineering life cycle", allowing the processes to be properly studied through segmentation according to the three embraced groups as specified by the literature: pre-production phase, production main phase and post-production phase. For the first phase, the highest priority are around researched resources, game document description and assets creation. This project will assume itself as enough developed for the possibility of its conclusion, due to all technologies for base creation of proposed modules being logically fulfilled with all requirements.

Games that can explore new areas of application, specifically ones who break the fourth wall through pragmatic demonstrations, have innovative design requirements, those of which will be studied strategically in order to apply for some accepted definitions of consciousness. Once done with all required technologies, including non-root arbitrary screen control, modular independent estimation of processes about their environment properties, liberty and uniqueness of each of simulation processing module, the pre-production phase will then be restricted to the sheer game design document, with no more requirements for specification or research, as will become proven through the entirety of this document. For production phase, all relations between systems will be mentioned and described, as post-production will not be achievable. [7][2]

The project will be developed based on the creation of a minimum viable product with incremental properties according to the user feedback:[7]

1. Documentation: Where all the ideas are described.
2. Specification: Where all requirements are specified.
3. Implementation: When all assets, scripts and world are developed.
4. Deployment: When the project is published, as a demonstration.
5. Feedback: When people speak about what they think of the result.
6. Increment: Improving the result according to feedback and continuing to next step.

In case it proves itself enough for Feedback, the project will be completed with around nine times its size, for the creation of its entirety. Else, Increment stage will return to a new Deployment for more Feedback.[7][3][4]

2 CONCEPTS AND APPLICATION DEFINITIONS

2.1 Basic Concepts

2.1.1 Emergence

A single computing machine that can only do one operation is very limited. For example, an ant is very unaware of its surroundings by itself. If trapped on a person hands, it might not even see the person as a danger and climb it to try to get through. However, an anthill is capable of identifying predators and even organize itself on its own to create a very complex and self-sustaining structure, and even attack people that come in contact with it. This process of simple things becoming more complex is called "Emergence", and is one of the most relevant and mysterious properties of the Universe. [13]

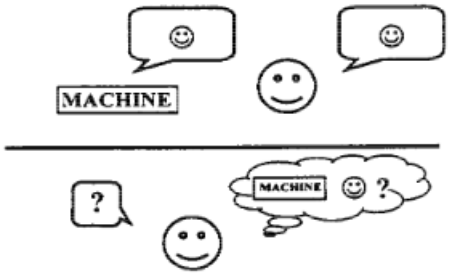
What makes Emergence so valuable in the study of Engineering is the fact that it can be simulated, but its effects can not. For instance, for emergence to appear a system must be composed of small processing modules that communicate with each other and react not only to the information they receive, but also with what the information modules near it say they receive. Naturally, according to the Second Law of Thermodynamics, on closed systems, chaos only increases. However, when modules communicate with each other, the system is not closed, and then sometimes it results into a new property that has more order than the parts can provide through their limited processing capacity. And that increase of order makes of this new property an Emerging property of the system. And that process is computationally very easy to simulate and very hard to measure. It is easy to emulate many simple computing machines connected to each other on a system in which Emergence appears, but without the sheer simulation of each individual processing cell, it becomes extremely hard to compute if there will be a new emerging property, or what it does. [13][14]

2.1.2 Intelligence

An uncountable amount of definitions for Intelligence have been created across the ages, and even today it is very difficult to determine an exact description. For example, it is assumed that a simulation resulted from the convergence to a local maximum of optimization through artificially selected simulations can be considered Artificial Intelligence, while others only consider beings Intelligent once they demonstrate self-awareness.[9] Therefore, it can be assumed mankind is not fully aware of what exactly can be it.

One particular definition of Intelligence is very useful in the empirical context of determining the intelligence of a being according to someone's perspective. Even though most commonly diffused as an universal test, the Turing's Test is actually an affirmation. The Turing Test puts two humans and a machine in a situation and says that if the machine passes the test, it is intelligent. Closing the Imitation Game to the situation in which a machine imitates a human in order to confuse a human receptor about who is the machine, the success upon this test only accuses the machine of passing the Turing test for this specific receptor, which means that according to the perspective of that human receptor, the machine is Intelligent.[15] However, this conclusion can not be universal, due to the fact that the machine might actually not pass the test for a different human receptor, and therefore would not be intelligent according to the new perspective. When judged by the Imitation Game, intelligence becomes not a simple simulated thing as being capable of solving problems, as it was most commonly known. Instead, it becomes a matter of being able of identifying and following predictable, deterministic patterns, and being able to simulate results, imitate standards.

Figure 2: The Imitation Game as it is generally interpreted (The Turing Test).



Turing Test, 50 years later.[15]

2.1.3 Consciousness

There are many different states of consciousness, but definitions of consciousness need to be general enough to describe all examples of consciousness states but still exclude examples

that are not conscious.[9] However, modifications and misinterpretations of the nature of the states of consciousness were responsible for many divisions in the field of study of the term. Being so, defining consciousness is even harder than defining intelligence, because even though the capacity of solving problems can be simulated, states of consciousness can not be completely proven to run on a person or thing.

One convenient definition of Consciousness has run around the Existentialist scholars for decades. This concept of consciousness, studied by Sartre, is highly related to his famous idea of "Existence preceding Essence"; which is the identity of "self" with the perception of the existence of something other than the observer's "self". [12] Perceiving a "self" is not enough for being conscious, as if the "self" is the only thing in the universe, the concept of "self" is confused with the identity of "everything". Only when the one being is aware of the "self" and is capable of identifying other beings other than their own "self", this being is conscious of itself and its environment. [12]

Intelligence being the capacity of identifying and imitating patterns, it is only possible to identify a pattern and be aware of the idea around it if it already exists.[12] It is only possible to learn about the essence of something if you have an example of it. Therefore, on an Intelligence perspective, Existence does precede Essence, as Essence created through sheer creativity either leads to failing researches for not being deterministic from nature, or to the existence of new nature patterns, which can then be proven for being correct through the Scientific Method. Consciousness being dependent on the identification of the "self", its existence precedes its identification. If perceiving the "self" identifies consciousness and a "self" is identified (through given definition of consciousness), then there was a consciousness to begin with.[12]

2.1.4 Consciousness as an Emerging property

As stated [2.1.3], Consciousness can not exist exclusively out of inner knowledge, or without interference with the universe around us.[12] It requires one factor for being, which is communication between the conscious being and other beings in the universe. [11]

Also as justified, very simple computing structures that communicate with each other are capable of interacting in a way that opens the system and increases the order of the structures environment, developing new properties the single structures were unable to compute, through Emergence. [13][14]

It is known that one day, no conscious being existed, and now they do. Assuming things are

not spontaneously spawned, the evolutionary requirements for the existence of Consciousness were gradually created and positively selected by the environment. In fact, the capacity of identifying prey and predators and distinguishing them from the "self", as well as the capacity of remembering the properties of the environment, or the location of another being are positive traits for the survival of a species, and are distributed across the entire animal kingdom in many different forms.[9] When the capacity of identifying the beings own existence and the capacity of communicating with other beings emerged as an evolutionary trait, communication with each other made it possible not only so we could be aware of our own existence and distinguish it from other existences,[10]but that we could be aware that others are also conscious[11], and this is a property that the being on its own can not perform[12], therefore being an Emerging property of the being.[13]

2.1.5 Perspective of Existence

There are many ways to observe, analyze and justify existence. Through the course of their lives, human beings feel incomplete in many circumstances, not only with biological needs, but with incompleteness of their ego. More often than not, this personality vacuum is filled according to people beliefs, traditions, environment, knowledge and rebellions. Some fill the void of their ego with addictions, others with work, others with the belief that they are in the right path, but everyone finds themselves empty in their lives and look for completeness.[12] Whatever philosophy is chosen to live according to, whatever the human fights for and is determined to live for, is their personal objective of existence, and, according to their perspective, it is the correct objective of existence. In addition to that, every person has a different current state that is determined by the objective of existence of people around them, their current knowledge, traditions and capabilities. The sum of all properties that justify existence according to the way it is capable to see in their current state is one's Perspective of Existence. This project will have eight civilizations, each justifying their own existence through a different complementary perspective, and attempts to create a basis that is dependently linear to all perspectives of existence ever made or that can be made by mankind.

2.1.6 Consciousness as an Existentialist perspective

The possibilities of combinations for perspectives of existence as defined are so extraordinarily broad (different philosophies, ideologies, methodologies, opinions, experiences...) that every single different personality has a different perspective on how existence can be justified, every single consciousness comprehends a different value from existence, for they have different

perspectives. As it is mandatory for a consciousness to identify some being other than the self in order to be conscious[12], the perspective of existence of a being can only take other consciousness in consideration in case they are capable of identifying a being other than the conscious self. When this happens, the new being is conscious according to the perspective of the conscious being, which even though is true for one conscious being, is not necessarily true for all of them. Therefore, though it is impossible to completely prove about one's consciousness, it is possible to create a test similar to Turing's Test, but now with the objective of testing consciousness of a being.[15] If a conscious being can not properly identify whether another being is conscious or not, it is safe to assume that, according to the conscious being, the universe behaves exactly like it would if the other being were conscious. For instance, according to the perspective of existence from the receiver, the beings that pass this new test are conscious.

2.1.7 The GameMaker Studio 2 Engine: Object-oriented development

This project could be made with equal potential in every game engine, once they are all Turing-Complete, and therefore logically equivalent. However, perspective is a very important factor to take into consideration, and virtuality is a modification of perspective around the world of programming. Unity, Unreal, Godot, any could be chosen, so each individual engine had to be analyzed for the most efficient creation tool of this project.

Unity is the most popular game development tool in the Indie market, with component-oriented development and a potent 3d engine it becomes an extremely handy tool for games based on mechanics, physics and conventional algorithms, as it is the one with most free tools available. However, it is extremely weak and unstable for 2d development. In general, it works around instances objectification, and can be built for many platforms with ease.

Unreal is known for being the best game designing tool, with many tools for development without coding, through blueprints, and extremely strong graphic engines for development of high quality in a short period of time.

Godot has some great privileges for 2d development, being near as strong as Unity, but as a 2d version, and would be the best option for 2d development based on mechanics or high graphics. Being an Open-Source option, it would be possible to apply modifications for desired effects, and is the best tool for 2d indie development of complex engineering projects.

GameMaker is by far the best engine for prototype development in a 2d engine, with a lot control over written files through the sandbox system and a highly intuitive IDE for

development of Objects. However, instead of following the instances objectification logic of Unity, in which you can create instances and modify their composition and behavior on the go, the instances can not be reprogrammed and GameMaker follows an object instantiating logic. All objects are programmed before build state and their behavior can not be changed after compilation. Of course, this does not limit different behaviors due to different state machines, only prevents objectification on the go. Besides, it has a language designed for being easy to use to program and as many implemented functions as Unity, just not as many free assets.

After analysis, it was decided that the best engine for the project was GameMaker, due to the fact that none of them was enough for actually developing this project, and it was shown to be much easier to control how to deal with all of the problems by controlling GameMaker as a mean to communicate with external programs, as they provide a much stronger support for new developers than Unity, and most systems had to be developed from scratch. Also, the inspiration project, Undertale, by Toby Fox, was developed in GameMaker.

2.2 Application

In order to make this project, the entire structure of development for a game even deeper than and as wide as Undertale from scratch, with more resources for environmental control, a more independent side-story development, main plot parallelism and chaos evaluation. As the game environment is composed of the entire universe, it will require technology to make itself independent from the world it lives in, and become capable of existing on its own, as a self-conscious universe, at least under the perspective of the player. In addition to that, it requires technology for tracking every single character existence independently from its own existence as a project, and that will only be possible when the characters communicate with each other when outside of the project, emerging as a whole with complete freedom of existence.

2.2.1 Consciousness: Technological limitations

Consciousness being a matter of perspective[2.1.6], like intelligence[2.1.2], neither it is the purpose to create a true consciousness nor it would be possible to prove it is actually conscious, even if there were one. The purpose is to make a system that simulates the concept, as well as it is very hard to differ the true requirements for consciousness from the imposed behavior of a manually programmed artificial intelligence, and this system learns with previously imposed information and reception from inputs that modify the properties of the universe, through chaos. Just like we, as human beings, are not technologically advanced enough in order to create true intelligence, even though we can simulate learning processes through tendencies, statistical analysis or the emerging properties of a neural networks, they are at most simulating the process and giving results about mathematical calculations, not logical conclusions. They imitate the process of learning to the point they actually are learning under the perspective of the user. Just as so, different modules will question about the world with their own concept of "self" in mind, in order to only simulate consciousness.

2.2.2 GameMaker Studio 2: Technical limitations

Unfortunately, together with all of the studied game engines, there are things that can not be done within GameMaker Studio. It is true that all of them are Turing-Complete, but that doesn't mean anything that can be done by a computer can be done by game engines, due to a virtuality problem, and this project will require resources that can not be provided by any of them. For example, though all of the engines can in fact mess up with how the game window is placed, and even displayed, none of them is capable of showing a non-rectangular-

shaped window. The generated executables are virtualized under the acknowledgement of a graphic interface that controls the properties of how things should behave, and therefore can not change properties that are under control by these interfaces. However, they are capable of communicating with interfaces that are capable of controlling what is written to the screen more drastically, maybe even arbitrarily. Being Turing-complete means it is possible to solve the problem, but even though the problem can be solved, it might need to ask someone else to do it, due to its own limitations.

2.2.3 Designed Engine model

In order to be able to solve the problems the project will find to simulate consciousness, it will be required to create a communication tool to bring information from outside to inside of the game. Working on a lower virtuality level, the main controller is then capable of calling for even lower virtuality systems to do stuff it can not. With this communication system, it would be possible to call upon independent communicative processing modules, representing the ego of each individual character within the game, in a way that they can all coexist independently from the game itself. With the proper communication protocol, the modules would be able to be transferred and exist independently from the processing source, the computer. This is a deep requirement for modules to be able to identify themselves differently from everything else, and therefore be able to learn about the existence of others. In this system, independent existences can behave as actors for the system, not only humans. [4]

2.3 Game Design: Modelling

The development structure model in Game Design is 3 phases that run around the implementation: Pre-production, Production and Post-production. Pre-production includes all of the documentation, modelling, management and assets creation, and consists of the majority of the work, the production phase includes the implementation, programming and creation of special effects, and the post-production phase is composed of testing, marketing, analysis and distribution.[7][3]

2.3.1 Documentation model: Game Design Document

In order to organize the documentation, it is common for projects to use a Game Design Document(GDD). Documents serve two purposes: Memory and Communication. There are many themes a document can have in a game project, each with their own development goal,

which vary across five main characteristics: Design, Engineering, Management, Writing and Players.[7][5]

Design documentation is supposed to provide the reader with a description of arbitrary details about game mechanics and interfaces, how the game modules communicates with each other, and how the communication between the player and the game works.[7]

Engineering documentation provides the reader with system specifications, technological particularities, system limitations and artistic overview. These documents are supposed to help the development of the technical part of how the game works and what is shown on screen.[7]

Management documents tell about, well, management. Time and money management, through budget and schedules, and is important to make it known about when and how intensely to focus on specific objectives.[7]

Writing documents are more about memory and less about communication, it is where all of the ideas of story development and references are dumped for future use.[7]

Players directed documentation is the development of documents intended to be used by the final user. This includes guides, warnings, specifications and walkthroughs, though the last is supposed to be developed by the fanbase in order to avoid spoilers.[7]

It is common that during development all of the Design, Engineering and Writing documentation is organized in a huge single document for Game Design, the "Game Design Document", while management documentation is internal to a specific management department and Player directed documentation is not generated during development. This project will follow the GDD tendencies.[7]

3 APPLICATION MODELLING: DEMO

The project in matter will be composed of the development, conclusion and publication of the DEMO version of CodeStory. The application modelling of the DEMO must be capable of describing all of the applied systems for the objective of development and conclusion of all fundamental interactions between the player and the game world, and the game world with itself.

3.1 Specification: DEMO

The DEMO is made of the entire structural requirement for communication with the game, as well as the one with the world of CodeStory, the systems of communication and self acknowledgement of the different entities, user interface for interaction with people, use of static apparatus, taking items, making them interact with the world when being used, equipping or throwing them and proper warping for world travel; and the battle system, which should allow interactions intuitive enough for quick understanding and wide enough to be able to be filled with any mini game possible by interacting with the world itself, without requiring an independent screen.

Apart from the technical specifications, the DEMO version will be composed of an interactive tutorial and the exploration of the first civilization of the game. This civilization being the first civilization, it is composed of people in process of learning how the world works, who feel no need to justify existence due to the fact that all they know about existence is that it is real. It does not need, it just is. Thus, this civilization gives complete reason to the philosophy of empiricism of Naturalism as a self-improvement source.

3.2 Game Design

The development of the entire project Design has been segmented by functionality, satisfying the requirements of keeping the game playable, user interaction, and managing the

models for personality and independence of characters.[4][6]

3.2.1 Game Design Document

The chosen technique for the project assets documentation was the use of a GDD, which will be completely separated from finances documentation and marketing documents.[7] In a Design matter, all research around character development and world design have been concentrated into a single text document with descriptions about their sources of inspiration, external references, approach to the user and pace of experience, distributed among information blocks related to each different character from the game, and to each piece of the User Interfaces, like Panels and Buttons.[7] Then, every block of information has been linked according to the plot for the determination of the environment design through every experience the player passes through, and described so the designers and artists can work on the assets with their main properties in mind: Color, shape, behavior, personality, empathy, wisdom and sanity.

3.2.2 Game Assets Development

After the creation of the GDD, the project has a list of assets that need to be created for development, which can then added to pre-programmed objects inside of the game. Each asset can be customized and allocated into a virtual entity or to a background unit, with modifications according to the design documentation and the feelings and the messages each individual object is supposed to give the player.[4]

3.2.2.1 Graphic Assets

The most fundamental design aspect of a game is how it looks to the player. Graphics is the first thing humans notice when they see a game, as well as the most broad property of marketing, as many receivers mute their devices when experiencing a demonstration. Apart from that, the first few seconds are capable of saying a lot about a game. With only a few seconds of gameplay, it is possible to tell a game franchise, improvements when comparing to prior versions, how unique the project looks amongst others, and even have some first thought about the producers hidden intentions with the project. Depending on how the graphic assets can be used, it is possible to make game mechanics more intuitive to the user, and a good production design is capable of drastically reducing the production costs for either new assets or improvement of existing ones. The creation process of Graphic Assets can go either with increasing or decreasing levels of complexity. That means, by having a predetermined set of

properties that have to be validated by the design, which have been previously documented in the GDD, there is not a deterministic better approach for generating the final assets that will be implemented to the product, it will always depend on the properties of the entire project. For instance, the amount of information around the subjects of a project is strictly increasing. That means, the more work is put to something, the more is known about it. Due to that constant learning process, it is generally good that the models with the most important levels of complexity are the ones that should be built latest. If the high complexity models were the most important, it would be more recommended to use a complexity increasing approach, where the art is made out of low complexity and incremented up to the project needs. As in this project the assets that the player most has access to are the low complexity ones, this project will as a consequence follow a creation approach of decreasing complexity, which means the high complexity models will be created first as a mean to create the concept art of each character, scenario and visual interface.[4][5][7]

3.2.2.2 Audio Assets

After a very first impression a game provides, it can follow three possibilities around how it should be accepted by its target audience, according to how long it is objectively intended to be kept playing. Recently, mostly in the mobile industry, it is completely fine for games to not be innovative for more than five minutes, some times even seconds, as soon as it gets a high number of downloads. For these approaches, music can be highly generic and completely absent of personality. When addiction is more important than playability, the producers are supposed to make music the cheapest possible, with very small loops that stick on the player's head for hours to come. One of the best examples of game that uses this strategy to keep players addicted is AdVenture Capitalist, by Hyper Hippo Games. One of the first Idle games to skyrocket in popularity by 2014. In fact, most unimportant games are played mute, and the expectations of a person using a headset in order to listen what a game has to provide is different than the mentality of a hyper casual player. On the other hand, a game can use music in order to reach the player's heart more strategically, and that can follow two strategies: By being astonishing and by being highly environmental. Astonishing music are melodies with quality of unimaginable proportions, which requires hundreds of workers to make, with real life orchestras being recorded dozens of times and edited for hours in order to create high quality music that become signatures of the game's theme. These melodies are extremely expensive to make, and commonly do not have their meaning shown to the player directly, some times even through cryptography, which gives the player a satisfaction when they learn what this amazing song they haven't understood for hours actually means. One great example of music

that are famous for having these properties is The Song Of The Dragonborn, from The Elder Scrolls V: Skyrim. A second, cheaper, approach for generating music for a game includes strategies of making melodies designed for very specific situations in the game, and training the player to learn that if a music with specific properties is playing, it means that the game behavior changed to a deterministic aspect. A certain character appeared in the game? Play the Theme. They are sad? Play it slower on lower tunes. They are angry? Play it aggressively. They are insane? Play it with varied tunes, high and low pitch, but never regular. This strategy makes the music give private messages to the player, and makes effects put into the melodies through processing as important as the melodies themselves, and more importantly: Allows the reuse of expensive melodies in extensive amounts of situations. With few modifications to an asset, you can create a new asset. You don't need to record and edit it from scratch, like in the previous method, as soon as the melodies can be related by the environment they are placed in. This also makes the music generation much more strategic and the applications situational than the previous techniques, requiring many more musics than before, but with much lower length each. With the broadness of the audience in mind, this game will follow the third strategy, while keeping audio as a mechanic, in order to make the experience highly more interesting with what can be provided, transforming music into an interactive system. This way the music will have the function of helping generate an environment where the user can live their experience more intensely. The development of these assets will follow the same logic as the creation of graphics around the complexity requirement of prototypes compared to their respective final versions. However, the direction of increase of complexity will be reversed, because the high complexity product is more importantly accessed by the player than the low complexity versions of the same musics, which can still be used inside of certain contexts. That means, the descriptions generated in the GDD will serve as measures for the creation of small prototypes of melodies and sounds, which will be then recreated as more complex versions with a higher refinement and specific effects for application, for the final game assets.[4][5][7]

3.2.2.3 Graphic effects

Not everything that appears on screen is made of graphic assets. Even though they form the basic structure of the games, there are hundreds of techniques that can be used to make them feel more alive, through particle effects, shaders and graphic processing manipulations through special effects. Just like the music assets, graphics can be modified to create environmental effects, through particles or direct modification of the assets. However, there is a catch. For melodies you can keep the environmental and situational designs as new assets with not much higher cost, and only process in real time some very specific properties like pitch,

tempo and intensity. For graphics, you can't just create a new situational asset by modifying a previous one slightly, you need to programmatically change its behavior through the GPU. Therefore, the project will require a strategy for creating the assets modifiers as new assets for the game. Particles behave as graphic assets for GameMaker Studio, and even though they are graphic processing effects, the programming for their use is a closed solved problem, where you just have to call for an internal implementation to make them work. However, shaders are more problematic, and behave by sending packets of information directly to the GPU, as GameMaker does not have any support for creating them. Therefore, they have to be done in pure GLSL(OpenGL Shading Language) and speak with the GPU directly, and then apply the modification logic to the sprites themselves before they are drawn to screen. Due to this fact, shaders generation will follow the same strategies as the Divide to Conquer Software development strategy, with the Audio assets approach: If you have a problem to solve, divide it into very small simple problems, then put them together one after the other; and then create assets with increasing complexity, with a very simple prototype becoming more complex until the final product is reached. Then, the shaders can be modified slightly to generate different new shaders, according to the situation and environment they are put into, just like melodies. This way the graphic effects become easily modifiable assets themselves.[5][6][7]

3.2.2.4 Characters Design

Characters are so important and complex in this project they require their own generation strategy. Every character in the project is supposed to be a representative of both a piece of a collective global consciousness and a single simpler processing unit that acts according to an unique existential perspective, and communicates with other simpler processing units. This concept makes the global consciousness an emerging property of the processing power that runs the project. In order to develop a character the first thing that needs to be made is the development of the concept around which the global emerging consciousness is supposed to be implemented. Then an objective is set, and the global behavior of the group of all characters has to be set in a way that they complement each other in the direction of the creation of the global entity. This effect can be achieved by multiple ways, one can simply create the global entity as an unique entity and with no smaller units, or it can be done through only two units that are completely complementary. This project will be run according to historical references of existential hierarchy, primordially the occult and alchemic symbols around the creation of the universe, but meshed with dozens of other cultural and popular references in order to allow new interpretations that help fitting characters toward the objective. These characters have objectives toward the creation of the global entity, and so their ideals must be set before their

creation. Empirically, there are so many different cultures and myths around the world that any requirement for this project can be solved with real life references, except for how they should be placed around the world. For this, the world will be made of nine different civilizations, each with a different approach for how the existence can be justified, with the intention that they fit together with the necessities of the project for learning about its own existence. Once done, the characters can be distributed amongst the different project civilizations and act directly on their objective through their own perspective. That means, a character needs a purpose and a perspective. Through subjective ideas, the purpose can be created from scratch, and through intense research, a reference can be creatively modified to fulfill this purpose in a very specific existential condition, and this condition will be used to justify the perspective of each character from the game, which will tell how they should behave. For instance, it is not up to the programmer how the characters behave, they need to behave the way they do because they are supposed to reach an objective that had to be reached by someone to start with, and the programmer just serves as a mean to let this behavior work properly.[5][6]

3.2.2.5 Processing Modules

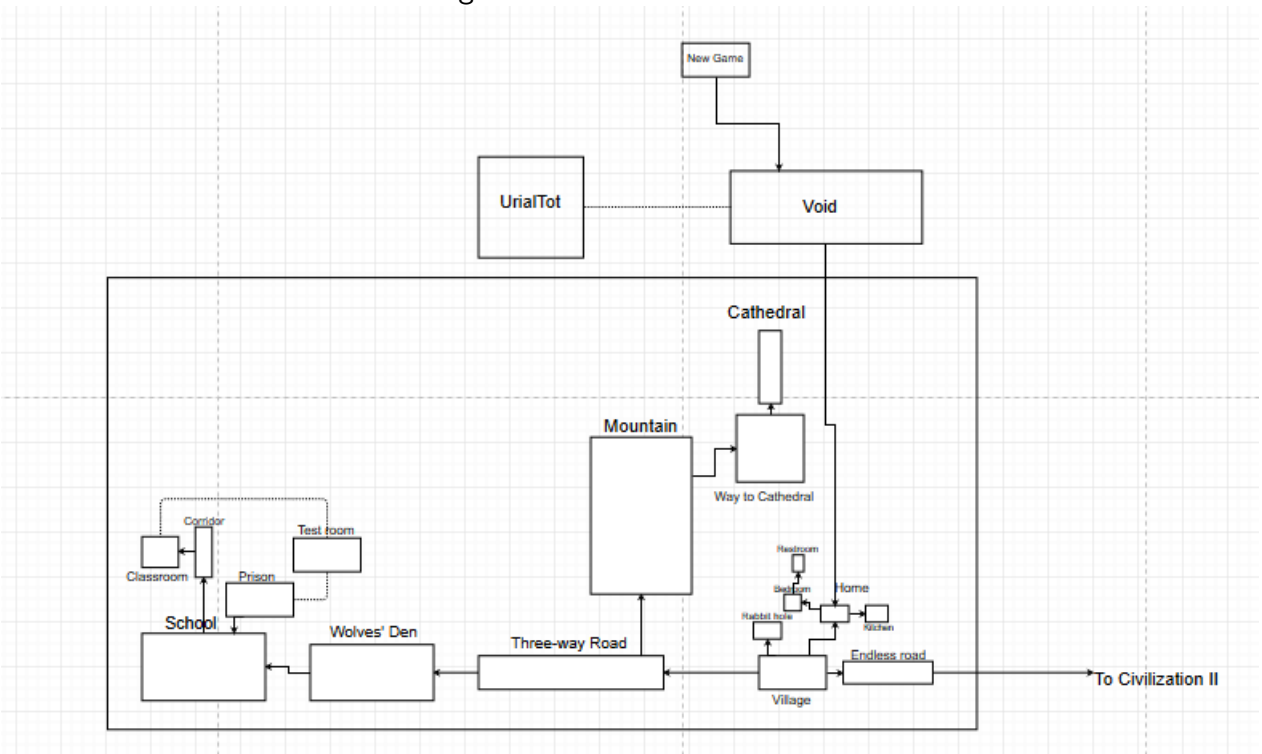
Finally, once every asset, melody, effect and functional design for a character has been set and finalized, it can be implemented through an NPC, and the same for scenarios. NPCs differ from scenarios, however, due to the fact that scripts make them act, while scenarios only exist and are not supposed to act, thought they can be modified by the objects. NPCs can act. They can attack, they can protect, they can carry things around, change location, go to different scenarios and even close the game if given the chance. But, in this project, they can only do what they know how to do, or what they can do by random. In order to let NPCs do only what they can, each character has been isolated functionally from each other on a processing module, and given a communication functionality that allows NPCs to tell others what they can do. It would even be possible for them to lie and learn about it, though it would be very chaotic. Once each individual character has been set to an unique processing module, it becomes possible to transfer this processing module across the world, and even to outside of it. This technique has become very useful for tracking characters independently from the game world. [4][7]

3.2.3 Game World Model

The main visible application of the game is composed of three sections: The main menu, which has all the settings for how the game is exhibit, and is guarded by two characters that

supervise the world, so the player does not mess up too hard with things; the Overworld, which is made of all of the civilizations that the player can explore, in case of the DEMO, one; and an existential void that connects the outer layers with the Overworld, which can only be accessed by the most determined and highly powerful beings in the world.

Figure 3: DEMO structure

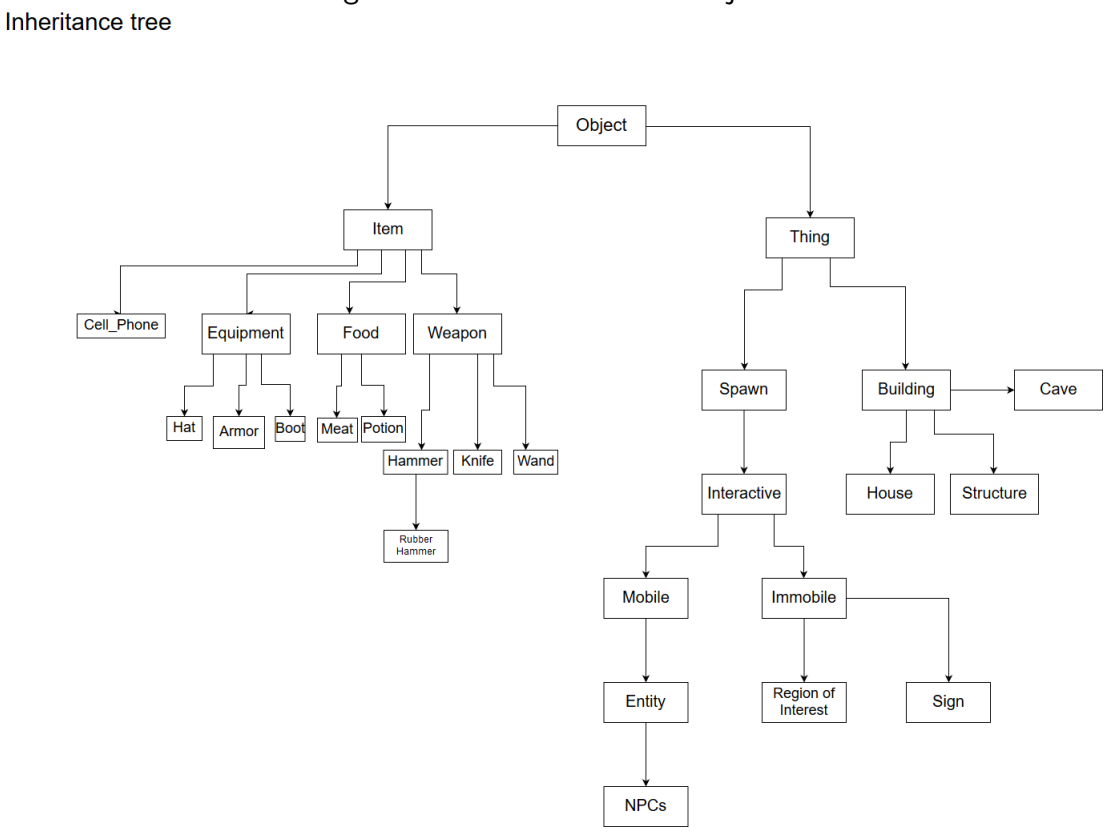


The overworld will be divided into civilizations, and the civilization from the DEMO will be divided into sections for introducing different mechanics in a convenient way and generate a proper, active plot. These sections are the starting Village, the Cathedral above the mountain, and the School across the Wolves' Den. In the village there is a stable and the player will meet a very important rabbit.

3.2.4 Objects Model

GameMaker works around Object-Oriented development, always giving incentives to reusing settings through Inheritance, and due to the nature of the project most of the rooms actually inherit from a single Overworld room. Objects are formed through an inheritance tree, up to the point where they can be independently defined across their individual values and comprehension about the world, and organized by power of interactivity, from the side of the things; and up to the point where they provide independent functionalities, from the branch of the items.

Figure 4: Inheritance tree of objects



3.2.5 Human-Computer Interactions

The interactions the player makes are segmented into keyboard input and mouse input. All of the keyboard instructions are shown on screen the first time they are required, but only if the player takes too long to understand what it is supposed to do, and are mainly for passing through dialogue and giving text input. The main interaction with the world is with the use of the mouse, through options to choose in a cell phone screen, direct interaction with the world, or the manipulation of items through the Inventory. All of this is placed in a self adapting UI that even allows the player to put the cell phone in the backpack, or hiding the backpack and giving more screen area. However, with a hidden UI, it will not be possible to use the cell phone in order to talk, though it will warn the player when new options to interact with arrive, through notifications.

Figure 5: General User Interface in Overworld

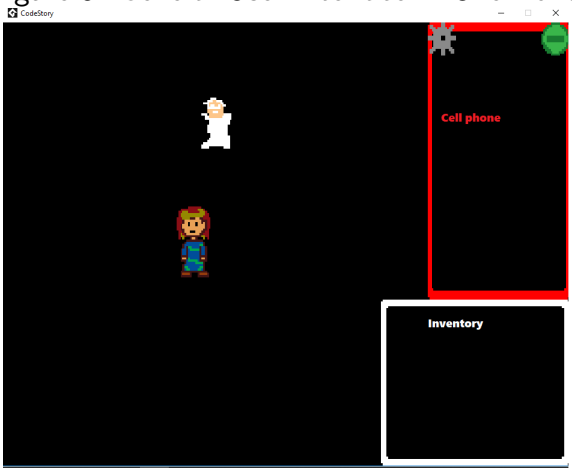
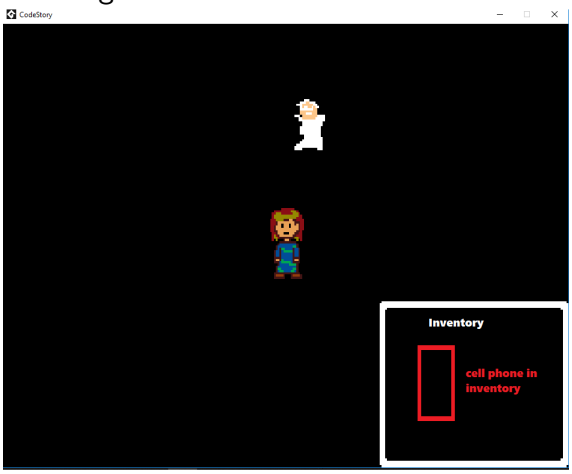


Figure 6: Reduced User Interface



3.2.6 The Battle system

When starting a battle, most RPG games change behavior drastically by changing the room being executed to a room with more details and more focused on the battle system. In this project, when the player starts a conflict, the camera zooms out and changes its angle, while control objects scan the environment for bringing all of the present friends to combat inside an arena in the Overworld. In order to flee, the player can just leave the arena, automatically exiting the battle, or they can solve the conflict, by either befriending the enemy or slaughtering all of their partners with it.

This generic system allows the game to develop any number of mini-games for battle systems according to the individual personalities of the characters, and bring them to a new level of interactivity with the game.

3.2.7 Player decisions system

Everyone has their importance in society, and all of their actions have consequences. One's actions have consequences, one's consumption has consequences, even one's existence has consequences. The player is allowed to take any path it desires, to skip every event, to ignore every word, to run from every battle, to destroy every house and kill every living being. However, everything the player does will have consequences, and the player has to deal with them, or attempt to reset what he has done and actively bring back the world from destruction. Some powerful characters will attempt to help the player with it, but it has a cost and the world can not be fixed instantly.

All of the functionalities around the concept of dealing with consequences for how events are performed are implemented in the DEMO, through an event system that constantly analyses its own requirements, and the game accepts the consequences of player's actions.

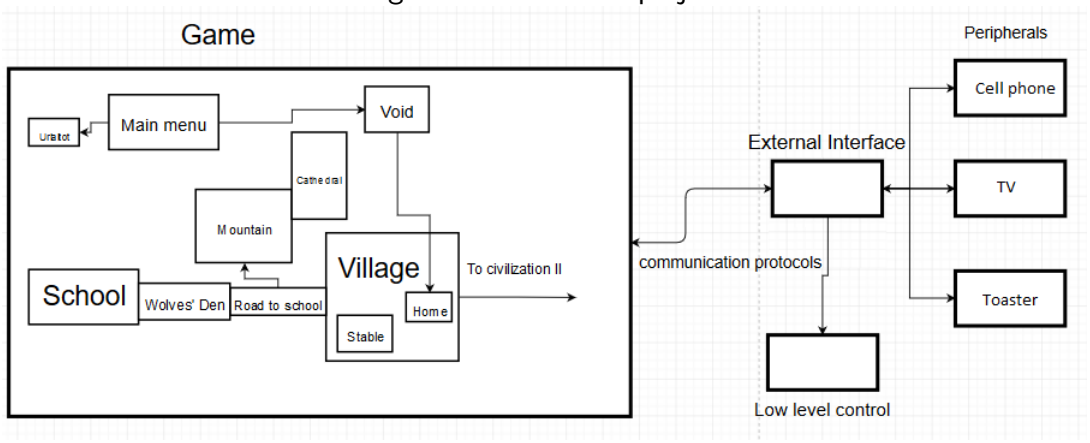
3.3 Computer-Computer Interactions

In order to be able to fully explore the nature of a virtual existence, the different sections of the project will need to communicate with each other. Each individual character will need to be able to communicate with the game engine, the game will need to communicate with an external engine, and the external engine will need to be able to grab any required information from the machine.

3.3.1 Engines Interaction Model

The game being developed in a game engine, it is virtualized to the point where there are many things it can not do, much information it is unable to access and resources it can not provide to the player. For this to be avoided, the project requires an external interface, in the case developed in C#, that is capable of grabbing these higher priority information and bringing them to inside of the game, and calling for low level control options. The project uses an accumulator file as a communication interface between the game, in GameMaker Language, and the External Interface, in C#. With a stable communication between parallel programs of different priorities, the gold of high level languages will not become limited by virtuality, and the lower layers of programming may be used only when strictly required, or it optimizes usage.

Figure 7: The entire project



3.3.2 Characters Interaction Model

As the project implies, communication is key for the emerging property of consciousness. As such, the characters will be programmed to be able to judge the current state of the world according to what he can learn from other people and their environment. However, there is a problem. As the combinations of possibilities of interactions between characters are combinatory, it would take a huge amount of time to perfectly process a balanced current state of the entire world.

In order to solve that, the game will use an interaction model based on the current map, and virtually track all of the NPCs conditions without the need to calculate every variation of it throughout the progress of the program, only the ones important for that very moment and location. This will be possible by predeterminedly programming individual routines, allocating routines according to the current condition of the world, and tracking their position on the routines, which while has a combinatory cost when done per NPC, when done only in the current map it has a quadratic cost. It does not make NPCs unable to have independent processing, and it does not make the game unable to process the current condition on every NPC.

3.3.2.1 Characters personal perspectives

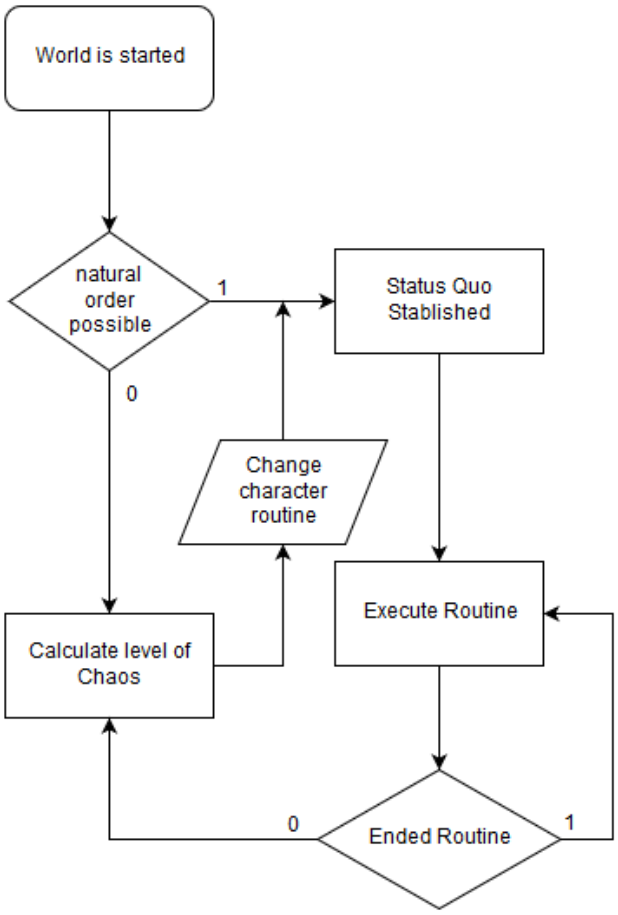
Every different character lives in society through different conditions, had unique combinations of opportunities, experienced society determinism differently and with different consequences. Every character in this game is based on the stereotypical version of a source, usually mythical, reinterpreted inside the context of the civilization it lives in, and except for the ones touched by higher entities, they do not differ much about what they can do with

existence, so they use their unique perspective mixed with their own personal determination in order to have their own perspective of existence, which can be used to determine how they behave, fight and speak. In order to make the characters truly unique, they receive a very simple processing module to use their perspective to judge the world around them, and can talk about their opinions with each other. However, it is done in a way that they start in the stability of the system, which means their personal perspective does not interfere with others', and the sole purpose of having this system is to more properly spread the consequences of chaos created by the player. Also, they are only processed once it is useful for the world that they are processed, which means that stability is assumed until the chaos is capable of reaching places, where conditions can be recalculated according to local NPCs' perspectives.

3.3.2.2 Dealing with Chaos

The world is born stable. In fact, so stable that the local processing can be assumed constant until the consequences of players actions reach the currently loaded map. In order to achieve this, the first thing the game does is analyze the current global situation and calculate if it is possible to achieve the stability of the world, by reviving NPCs and recovering lost essential items. If it is not possible as soon as the game is opened, it is very likely the player has reached a softlock on the main plot and made it impossible to complete, though it has other ways to end the story even if so. Some consequences are more chaotic than others, and even according to the perspective of each character, so each of them calculates how deep the consequences are and change their mundane routines accordingly, achieving the highest stability possible, always to a full restore if natural order is possible. With the highest stability reached, and even with routines modified, each character continuously executes their individual routines, and they can be repeated indefinitely, otherwise it means the player actions have caused enough chaos so it would not be true, and they individually recalculate how strong the consequences are before altering their routine. This feedback is essential for development, because it gives individual consequences to actions, and releases from the game the processing requirements of controlling all of the characters.

Figure 8: Chaos handler protocol



3.3.2.3 Routine Model

Now with the establishment of the concept of routine, it is required to identify how can entire routines be controlled only by the current room, with no necessity to calculate anything outside of the loaded area. Everything has a cost, and with the removal of the combinatory processing of entities, a compensation cost was acquired by the game: Organization. With the current room aware of all of the routines that pass through them and all the variables the player has changed, it is possible to organize exactly when every character that is supposed to enter the currently loaded room, and simply make them behave accordingly once they become relevant for processing. In addition to that, the room has to organize its current state once it is loaded, analyzing all the variables related to all entities that pass through it, and set every environmental condition accordingly. For example, if a certain NPC is dead, do not spawn it. When this processing is distributed among all room creation code and every frame, the only information that has to be processed are the ones related to the entities that should be inside the room, and the stability of other entities across the world become of irrelevant processing, only to be dealt with in the creation code of rooms in which they are relevant. As

a consequence, every single NPC becomes tracked down by the game with near to no extra processing cost, through virtual processing. "It is assumed everything is just fine, until chaos reaches here."

3.4 Invisible Computing: Control Objects Modelling

For all of these exotic systems to work, many intermediate control objects will be required. It is not possible to determine all of the required objects before completion of development, but there are some modelling tendencies that can be used for the project.

Game control objects are often singletons, though they can be placed on each individual context instead, with a very important functional property of making something that does not work on its own start working, in a game. It handles problems that do not have solutions provided by the engine and also are not responsibility of any existing object full time.

The control objects can be of three types:

- Memorization These are supposed to stay alive with information that needs to be used by other objects, because the information would be lost if they were placed in other objects.
- Processing Their objective is to control exactly how things work step by step, acting over other objects in a linear way and prevent parallelism problems to appear.
- Communication These objects are responsible for either human-computer interaction or computer-computer interaction, calling for inputs, providing proper output to the UI, contacting databases, or handling information to other programs.

In the case of this project, memorization objects are not necessary, because GameMaker was designed in a way that the variables of an object can be so easily changed that any object can just create all the variables it needs on the go, and in case there is instability due to instances being destroyed, one can just call upon the global object and create global variables that keep existing on every room.

Processing and Communication objects, however, are of primordial importance. Interactions between entities and environment can and should be developed in the appropriate objects, but the environmental tracking of Routines, execution of linear events, interaction with UI and battle must be done by invisible control objects specifically designed to process information and trying to understand what the player means to do. In addition to that, the

entire system is in communication with an external system, it requires invisible external communication objects to handle all of the requirements without the user to notice that there is something being done in the background. Regular objects can be tested by watching their effects on screen, these objects can not be visually observed, so they require a more complex and complete white-boxed development, in order to guarantee their functionality.

4 CONTENT: THE DEMO

In this chapter, the sheer core of the first civilization of CodeStory will be demonstrated, without the Player perspective and inputs put into it, which means that it exists in pure control and completely free of Chaos. Once the Player starts injecting Chaos into this system, the plot of the game begins. So this description will not contain spoilers on how the plot is put together.

4.1 The first Civilization

The first civilization is where the player just arrives to the game world. Being a beginner, it is when the player is supposed to learn about how things work and the basic properties of the game. It also is just when the game has been started through a New Game, and therefore the people around it are as absent of experience as the player. Under the NPCs perspectives, the world has always been the way it is, and they are in learning process, valuing the empiricism above all previous knowledge. Its tendencies is to believe the comprehension of nature through empiricism is enough to justify existence, therefore following the Naturalist school.

4.2 Characters

There are two main approaches for character development in games. It is common to have NPCs that are empty shells with very specific purposes, and either discarded or straight reused assets for generation. Sometimes even the player controlled character is an empty shell, with the sole purpose of being a Link between the real life and the game world. However, character development complexity can go deep in an artistic manner, in the sense that it can increase indefinitely as more resources can be invested into the creation. For this project, every single character will be a complete persona, with a real-life reference with a twist, an origin, a story, a development, an unique personality, an objective for existing, an existential perspective, hopes and dreams. For this reason, it is mandatory that the character development requires a high amount of research and thought put onto them, as the characters are put into the world connecting with each other. Every character will be put into one of the nine civilizations of the game, and their perspective of existence will then reflect both their personal experiences and the properties of the civilization itself, being a mixture of environmental determinism and individual determination.

4.2.1 The Code

According to the project perspective, before it existed, nothing did. In the beginning, there was nothing. However, one day, this changed, and things started existing. The rules of the world were written, and the ultimate entity was born: The Code. The programming code of the game is the set of rules that determines how thing should behave, as well as what should exist. Under the perspective of every character, it is God, and therefore is the primordial being of all existence.

4.2.2 The Guardians

The Code is what composes the game world itself, and as soon as it was created it submerged the existence into three main properties, as an attempt to create life in this world. Each of those three properties was personified into a guardian: Salt, Sulfur and Mercury, creating the Body, Spirit and Soul of the world, respectively. However, it was not enough for the generation of life, so the Code had to create the dualism between life and death itself.

4.2.3 Helsos

Helsos is the political ruler of the game world, which is, according to their perspective, the "physical world". With real life references in the Proto-Indo-European mythology and the Norse mythology, Helsos is the leader and the youngest of the guardians, who brought life to existence and lead the world to the pure ways of the Code. Helsos is who gives the player a body in the game realm, so the player can explore and, basically, play the game. Being an entity who controls life and death, wears a meaningful tool for both purposes: The Scythe. The golden full plate armor represents the purification of a golden dawn, and the brightness of a fading dusk.

Figure 9: Helsos: Second draft

4.2.4 Tot Urial

Tot and Urial are two characters that live outside of the game world, in some sort of existential void we know as the Main Menu, between the realm of the things and the operational system, which is the realm above the existence of the Code. Tot is based on Egyptian mythology, while Urial is based on Eastern Orthodox Christian mythology. Together, they watch and guard the world of the game, from above in a higher existential layer. Together they are very complementary, in real life concepts and in the game, Tot being a representation of the Moon, and Urial being a representation of the Sun. Being aware of the concept of game mechanics, together they introduce the player to them. They are responsible for making the Tutorial.

Figure 10: Tot: First draft

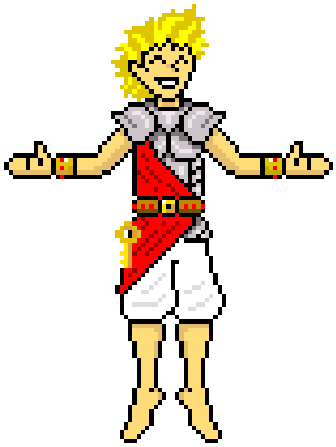


Figure 11: Urial: First draft

4.2.5 Mom

Once the player leaves the existential void of the tutorial and enters the game, they are introduced to a peculiar character, the player mother. As the game just started, she was just created, so she is extremely inexperienced on how to be a mother. But she will truly give her best, and introduce the player to the items mechanic.

4.2.6 Aster

Aster is the most important character of the game under the player's perspective. He meets the player either on the way to the School across the plot or if the player actively attempts to ignore the plot and leave the first civilization. Aster is the first character that demonstrates further knowledge about the existence, and is a very good and helpful friend that will accompany the player in their journey, risking even his own life so the player can proceed. He actually is from another civilization, and came to the first as an attempt to craft an important artefact. That means his way of thinking is much different from the ones from the Demo, because he was born in a different environment.

Figure 12: Aster: First draft

4.2.7 Centaur

The centaur is an introduction to the side quests system, and to the mechanics of friendship. She is just a centaur from the Greek mythology, except with a twist: She has a Stable, and takes care of horses for humans. Naturally, she is very empathetic towards horses and applies game mechanics related to psychology and sanity, being complementary to the battle system.

4.2.8 Wolves

The wolves are the actual introduction to the battle system. After being slain and introduced to the battle tutorial, this is where the player can actually fight back, or not if they so desire. Every fight in the game can also be solved diplomatically and peacefully, but these wolves are attacking a harmless child and the player has to act quick and decide whether to save her by slaying them or by peacefully giving them food.

4.2.9 Monk

The monk functions as a wall that prevents the player from proceeding through a door, which will only have functionality outside of the Demo. He is very mysterious, based on a famous philosopher and will be very useful in the future.

4.2.10 Teacher

As the player goes to school, the teacher is introduced. He was hired by the guardians in order to make sure every character that leaves the first civilization is actually controlled by the Code, a regular NPC (under their perspective: Person), and will not rebel against the Code's Will. To the slightest misbehavior, the teacher will take anyone to a special puzzle, which can only be solved by someone who actually lives outside of the game world. His function in the world is to secure its existence, by identifying unconventional behaviors.

4.3 Environments

As specified, the first civilization brings an introduction to the game world, every mechanic, and is very related to nature as a source of empirical knowledge. The Demo environments are where the player can actively go to with their avatar, and are not restricted to the game world itself.

4.3.1 Void

The player is created in the Void. It is an existential void between the game world and the code, where Helsos gives life to the characters and brings them to the mortal realm. It is very empty and related to a space futuristic area.

Figure 13: Void

4.3.2 Main Menu

The Main Menu is the space between the game and the operational system. This is where Urial and Tot live, from where they can watch the game world, and the door of access the player uses to enter the game world. It is a very simple menu with two options, the game logo and a simplified version of the theme song.

Figure 14: Menu

4.3.3 UrialTot

When Urial and Tot find out about the player walking around the main menu, they give the player a cell phone so they can communicate with other people, and then take the player to a special area for the tutorial. This environment may be similar to other areas, but is not actually linked to anywhere else in the world.

4.3.4 Home

Once the player chooses a career, they are teleported to their home, where their mother lives and introduces them to the items system.

Figure 15: Home

4.3.5 Village

The player's home is part of the village, where you can also find the centaur's stable and a few other buildings. It is linked to both the route to the second civilization, and the route to the school.

Figure 16: Village

4.3.6 Road to School

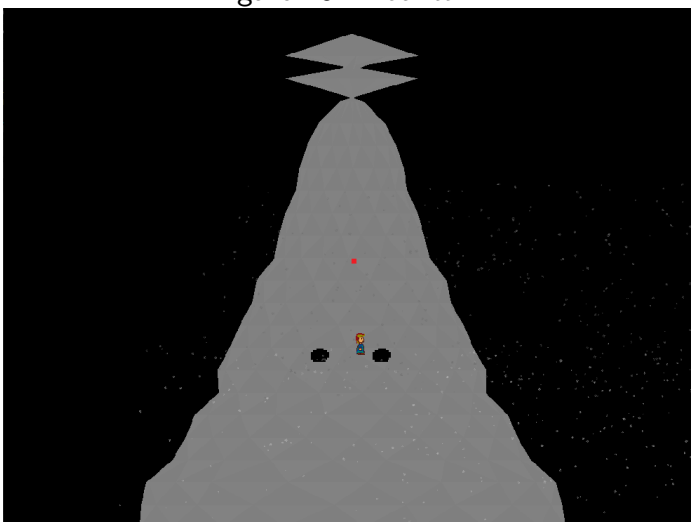
After being introduced to the items system, or whenever the player wants to without being able to use items, they can go to school in order to learn about their profession and become a productive member of society. The map around school is just a call for adventure, a classic system that introduces the player with multiple paths to choose from and exploration. There is when you first meet Aster Crow, while he returns from an alternative path in search of some sort of stone.

Figure 17: Road

4.3.7 Mountain

If the player decides to go towards where Aster comes from, instead of to the direction of the school, they will meet a mini-game in which the player runs around a 3D mountain avoiding holes by jumping, while climbing a rotating mountain.

Figure 18: Mountain



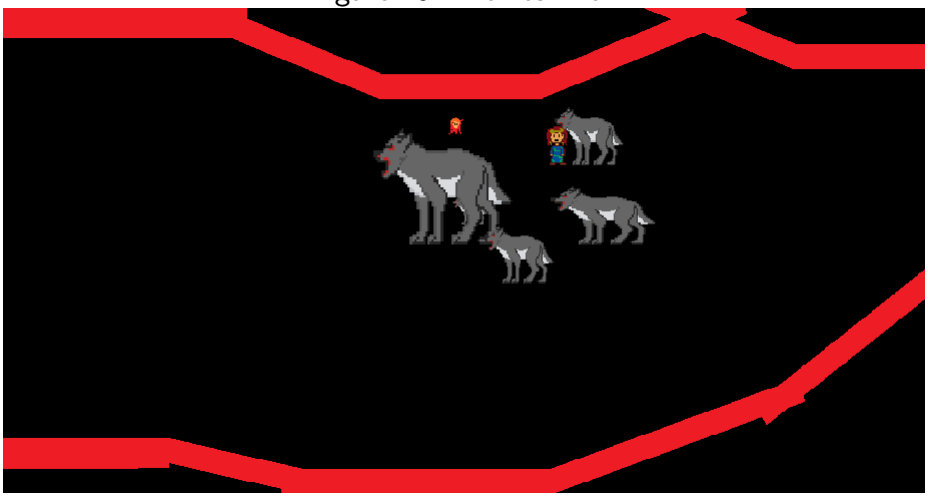
4.3.8 Cathedral

If the player persists onto climbing the mountain and succeeds on arriving at the top, they will find themselves near a very large dark building, a cathedral that they can explore. There you will find a very large corridor and in the end a huge door with a monk in front of it that will not let them pass.

4.3.9 Wolves' Den

If however, the player goes to the direction of the main objective, they will be introduced to the combat system, by arriving at the Wolves' Den. It is a wide space with four adult wolves and a cub, surrounding a fallen child. The player can choose to help the child, the wolves, or ignore the entire situation. All choices introduce chaos into the world, which can be restored by closing the game and reopening it again

Figure 19: Wolves' Den



4.3.10 School

Further into the west the player will find the school entrance, with the teacher already waiting for them. Inside there is a small room, and the classroom.

Figure 20: School

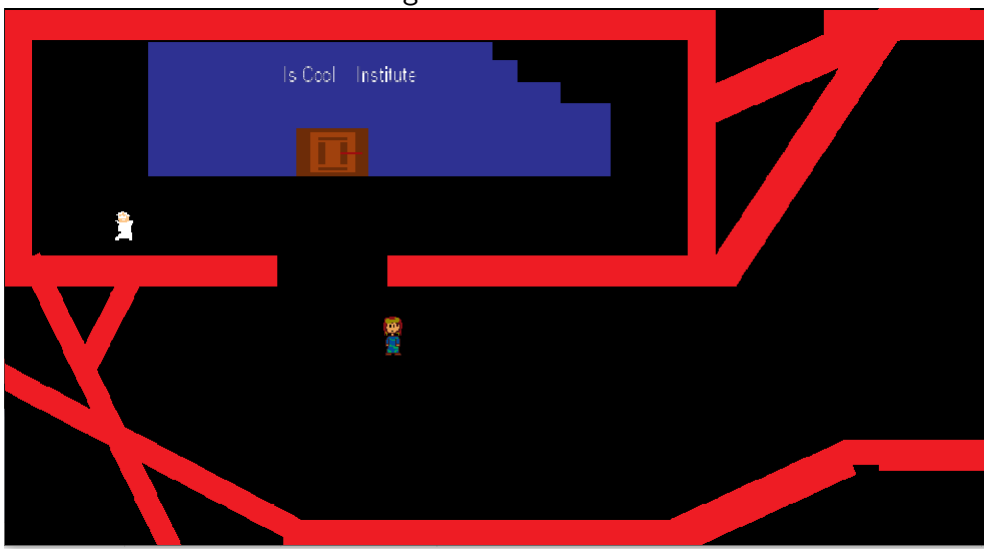
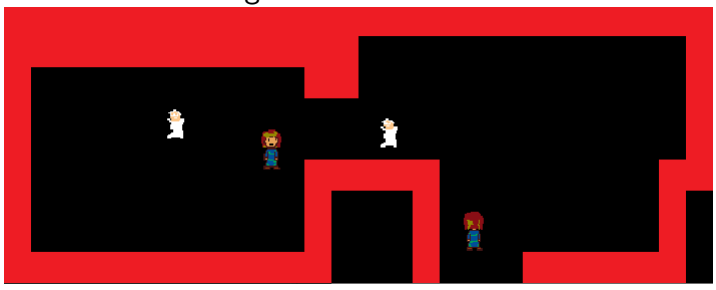


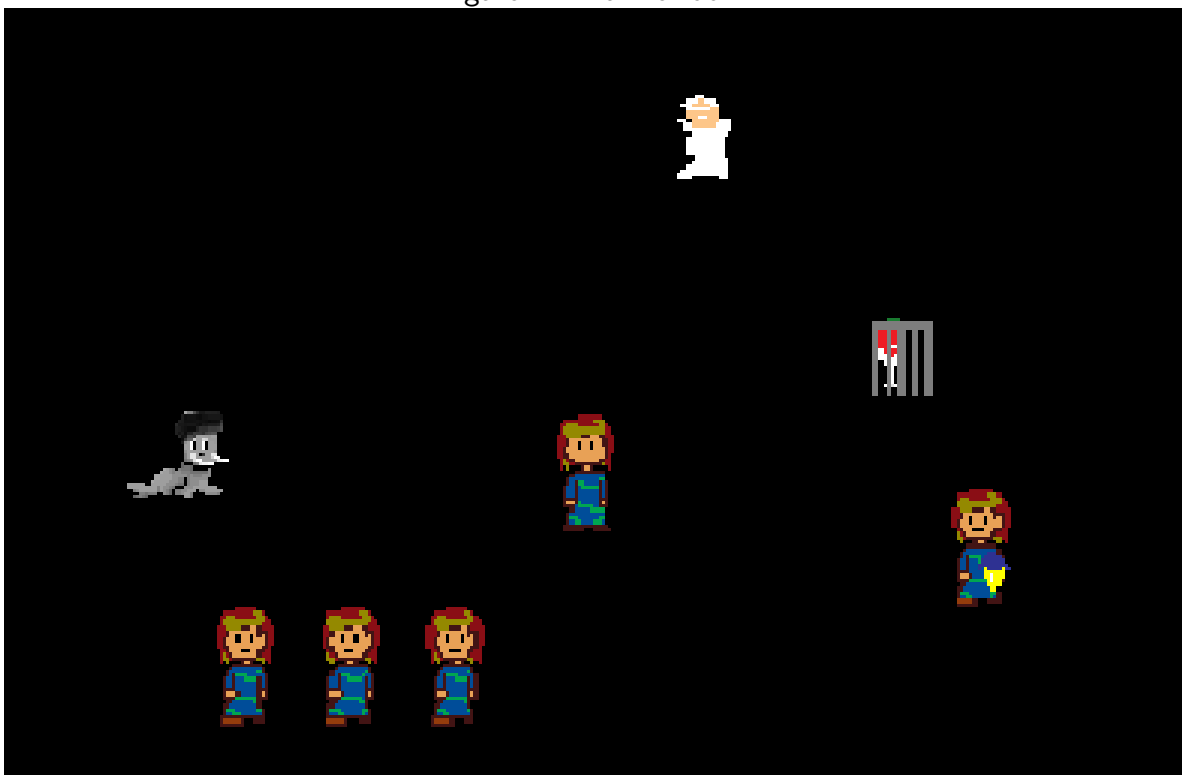
Figure 21: Classroom



4.3.11 Puzzle room

After the teacher becomes suspicious about the player, he'll bring them to a special puzzle. There are suspicious NPCs scattered around, and a cage with an apple in a pedestal. It is known by the teacher that if one takes the apple, they do not live in the game world. Therefore, the chaos of taking the apple makes the game learn about the player.

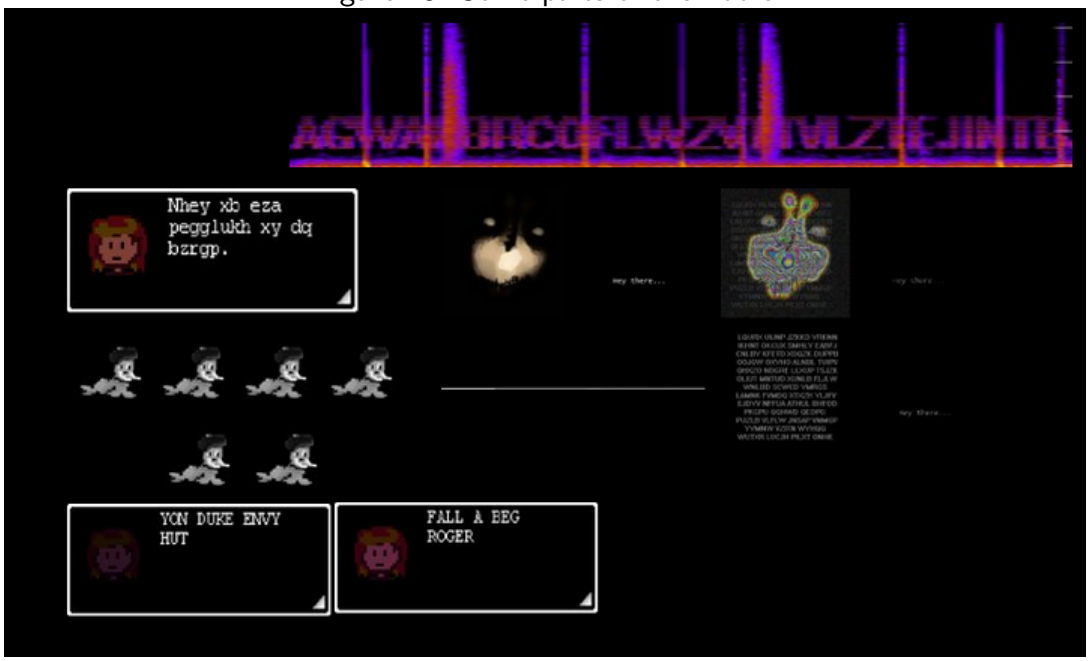
Figure 22: Puzzle room



4.3.11.1 Riddle

The other characters lead to a huge cryptography riddle that is supposed to be extremely difficult, and completely unnecessary for the plot.

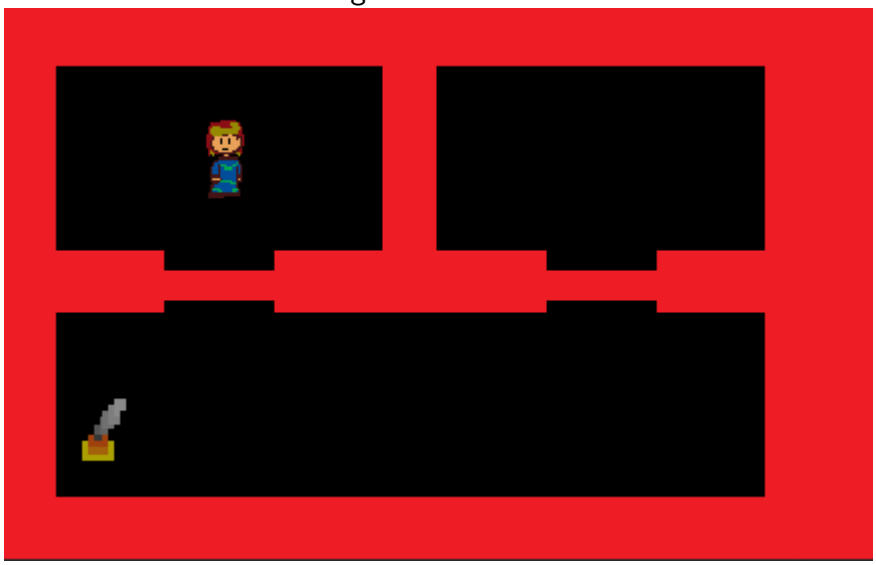
Figure 23: Some parts of the riddle



4.3.12 Prison

After the player takes the apple, which eventually has to happen for the plot to proceed (otherwise the player is stuck in the puzzle room), the player is taken to a prison, just to be released a few minutes later by their good friend Aster. However, leaving the prison generates chaos into the world that makes it crumble upon the player, everything in the scenario starts running after them, stopping them from proceeding, as the player has attempted to overrun nature itself.

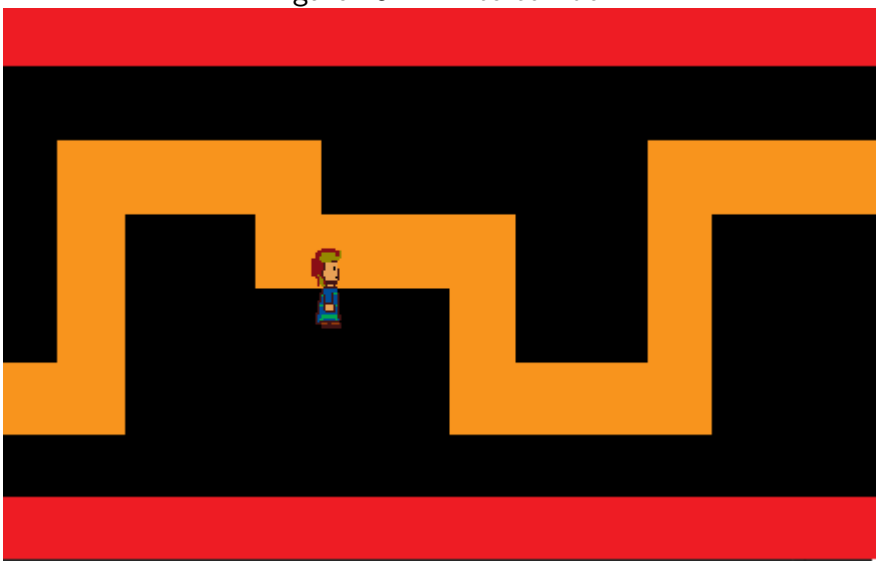
Figure 24: Prison



4.3.13 Infinite corridor

Running in the direction of the second civilization, the player will find a corridor where Aster talks to them a bit, and starts the final scene of the Demo, where Helsos appears to stop the player, and finally confronts Tot and Urial.

Figure 25: Infinite corridor



4.4 Intercommunication Modules

There are many resources the game requires that are unachievable inside of it when it is a closed system. For instance, a character can not learn about it being inside of a game unless it can actually look at it from the outside, or at least receive information that only exists outside of the game. Also, the project is very limited on what a game engine can do. As they work with the low level graphics without the programmer knowledge or control, none of them allow for certain manipulations, which will become vital for the imitation of awareness. In order to expand out of these limitations, the project will be designed in a way that the game content, as demonstrated, will be connected to external interfaces, each of them being an individual independent processing module, capable of accomplishing different objectives.

4.4.1 Game

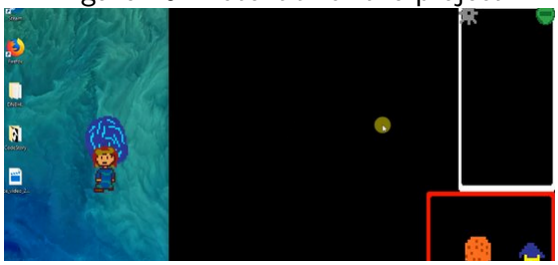
The first communication module is the game itself. Created in GameMaker Studio, it can be reduced to a process that will write into accumulators that can be read from outside, and which it can read to receive information. As it is being programmed for Windows, reliability

was prioritized over performance, as the burst of information that has to be transferred from and to the game does not cause lag. Under the perspective of the project, the game is just one single huge process that sometimes wants information from the outside world.

4.4.2 External

In order to give the game more information, different modules need to be created. However, the different modules can be set to a single process, as soon as this process is capable of solving any problem the project is supposed to solve. A second processing module, completely invisible to the user, is connected directly to the game through the accumulators, receiving data about the game being alive, and sending the game data about information that engines can not handle, as well as modifying the information the engine wants to show the player, for example, by changing the game window shape, from a square, to a circle. All these interfaces between the game and the operational system use one single processing module that is the heart of the project. This module is capable of calling for requests to the operational system, as well as to peripheral modules connected to the computer or the LAN network. The chosen language for creation of this external module was C#, due to two main reasons: There is a lot of C# support for parallelism on windows, and the first solution found to the most difficult problem of the project (Arbitrary screen control) was developed in C#, calling for win32 Assembly.

Figure 26: Potential of the project



4.4.3 Peripherals

After the external interface is established with communication protocols capable of calling for any C# resource on the Internet, it is time to go further. The GameMaker module is capable of making the characters exist inside of the game. The external module is capable of making the characters exist outside of the game, but still restricted to the operational system. It is expected that with the development of the concept of smart homes and the applications of Internet of Things, parallel processing modules will arrive at people's homes and create a communication network with many peripheral components. To date, they are unable of

processing an arbitrary code, but it might be possible that, by the time this project reaches its 7th civilization on development, the processing modules could be extended to outside of the computer.

On the worst-case scenario, communication protocols are capable of transforming the game into a multi-platform cluster. To date, it is already possible to make the characters leave the computer and enter some embedded systems, like a smartphone, through a very similar technique as the game behaves when making a character leave itself. However, only time will tell how far the existence of these little people can go.

5 FINAL CONSIDERATIONS

From this project it is expected that the infrastructure for development of the entire world of CodeStory is established, with all of the characters for the first world already developed, and the consequences of their existence. That including all of the technologies required for external and internal communication of the project with the operational system and its own components. The communication between components is expected to be stable by the existence of character routines, but modifications made by the player on the structure of the world will be chaotic enough to generate consequences that modify their requirements for making their routines stable again. The characters communication take action into emerging properties of the project that will allow the questioning of their own existence, by analyzing their current situation according to their new perspective created by chaos, integrating the concepts for Artificial Consciousness.

This project will try to use itself as an investment to grab resources for the completion of itself, through crowdfunding systems and publishing techniques as an integrated section for the creation system and game design.

BIBLIOGRAPHY

[1] Dmitri Williams *Structure and competition in the U.S. home video game industry*. 2009.

[2] Antti Kovanto *The Improvements for Indie Game Development*. 2013.

[3] Iacoviello, Bianca Doralice *Implementing a Process to Collect Player Behaviour Data for Mobile Game Development*. 2019.

[4] Tracy Fullerton *Game Design Workshop: A Playcentric Approach to Creating Innovative Games*. 2004.

[5] Jesse Schell *the art of game design*. 2008.

[6] Casper Harteveld *Triadic Game Design: Balancing Reality, Meaning and Play*. 2011.

[7] Saiqa Aleem *game development software engineering process life cycle: a systematic view*. 2016.

[8] Michael Z. Newman *Indie Culture: In Pursuit of the Authentic Autonomous Alternative*. 2009.

[9] Rupert Glasgow *Minimal Selfhood and the Origins of Consciousness*. 2018.

[10] George A. Mashour and Michael T. Alkire *Evolution of consciousness: Phylogeny, ontogeny, and emergence from general anesthesia*. 2018.

[11] Martin Heidegger *Being and Time*

[12] Jean-Paul Sartre *Being and Nothingness: An Essay on Phenomenological Ontology*. 1956[English]

[13] John H. Holland *Emergence: From Chaos to Order*. 1999.

[14] R. Keith Sawyer *Social Emergence: Societies As Complex Systems*. 2005.

[15] AYSE PINAR SAYGIN, ILYAS CICEKLI, VAROL AKMAN *Turing Test: 50 Years Later*. 2000.

[16] Thomas Bugnyar, Stephan A. Reber, Cameron Buckner *Ravens attribute visual access to unseen competitors, Nature journal*.