Tampere University of Applied Sciences

Game Development Using the Visionaire Studio Game Engine

Janina Korpela

BACHELOR'S THESIS
May 2024

Degree Programme in Media and Arts
Interactive Media

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Media and Arts
Interactive Media

KORPELA, JANINA:
Game Development Using the Visionaire Studio Game Engine

Bachelor's thesis 46 pages, appendices 1 page
May 2024

The purpose of this thesis was to research a video game development tool called Visionaire Studio, which specializes in the point-and-click adventure game genre. The goal was to examine Visionaire Studio's merits as a game development tool and research why a developer would or would not choose Visionaire Studio as their game engine.

The theoretical section first explores the concept of game engines and the point-and-click adventure game genre to give background for the thesis. Then, Visionaire Studio is studied in multiple ways. The design and community of the engine are examined, and the various features provided by Visionaire Studio are evaluated in detail. Alternative options to Visionaire Studio are also introduced.

The thesis also includes a practical game project created using Visionaire Studio. The goal of the practical project was to develop a small-scale 2D adventure game while simultaneously examining the features and workflows presented in Visionaire Studio. The project focused on showcasing what the common game production pipeline in Visionaire Studio could look like.

It was concluded that Visionaire Studio's straightforward design, extensive documentation, and active community make it a great development tool choice, especially for beginners or anyone otherwise unfamiliar with coding. Additionally, the game project and the process of creating it were reflected upon.

Key words: visionaire studio, game engine, point-and-click adventure, game development, software

CONTENTS

ABBREVIATIONS AND TERMS

2.5D	Two-and-a-half dimensional. A game mixing 2D and 3D graphics, perspective, or gameplay.
AAA	High-budget games produced or distributed by major companies.
Box2D	Open-source two-dimensional physics simulator engine for games.
CSS, Lua, Ilios	Programming languages.
Indie	Independent developer.
MS-DOS	Microsoft Disk Operating System.
NPC	Non-Player-Character.
UI	User interface.

1 INTRODUCTION

For several years now, the gaming industry alone has dwarfed what were once considered the biggest entertainment industries – music and film. This growth has been accompanied by increased demand for various game development tools, in particular game engines. Since their creation, game engines have come a long way in enabling developers to bring their visions to life. From their humble beginnings, game engines have grown into versatile powerhouses that offer high-fidelity graphics, realistic physics simulations, and support for a variety of platforms. The proliferation of modern game engines such as Unity and Unreal Engine has also democratized game development, facilitated by an abundance of different development tools, active developer communities, and a wealth of tutorials and resources. However, this abundance of choice can halt the process at the crucial first step: which software should you choose to create your game?

This thesis aims to examine one of these tools, the Visionaire Studio game engine. The goal was to explore in detail the design, community, and features of Visionaire Studio, and provide a good overview of the capabilities of the engine. This thesis also includes a practical project that aimed to create a simple game in Visionaire to demonstrate the possible workflows and pipelines of the engine on a more practical level. Additionally, the topics of game engines and the point-and-click adventure genre in which Visionaire Studio specializes were introduced.

This thesis is motivated by a combination of factors: the abundance of various game development tools available online today and a personal interest in game development. As an artist with little programming knowledge, the aim was to determine whether this tool could enable someone with a similar background to create a game independently from scratch to finish. The author hopes for this thesis to serve as a well-rounded look into Visionaire and answer the question of why and for whom Visionaire could be the game engine of their choice.

2 GAME ENGINES

A game engine is a software or framework that enables the creation of video games. The main goal of a game engine is to make the life of a developer easier, allowing them to create and manage several aspects of a game efficiently. Some common functionalities that a game engine manages include but are not limited to

- 2D or 3D rendering
- user input
- physics simulation
- sound
- animation
- memory management
- graphical user interfaces
- debugging
- compiling and exporting projects.

Today, a wide variety of different game engines and frameworks are used and are widely available for the average user. As the games industry and the demand for more powerful and flexible development tools grows, new engines are released while existing ones are expanded. (Andrade 2015; Rădulescu 2020.)

Game engines can be roughly divided into two categories: generalists and specialists. General engines offer a variety of tools and features that make them very flexible and powerful, which is why they can be used to develop almost any kind of game. They are often used by both large studios as well as indie developers. Unity and Unreal Engine by Epic Games are arguably the best-known examples, both of which are the current industry standard, but other similar examples include Godot Engine and CryEngine. Some major studios also prefer to use exclusively their own engines to develop their titles, such as Anvil, used by Ubisoft for its Assassin's Creed titles, and IW Engine which powers the Call of Duty game franchise. (Andrade 2015; Rădulescu 2020.)

Visionaire Studio is an example of an engine optimized for a specific type of gameplay mechanic and user group: the point-and-click adventure genre. These

specialist engines are often smaller in scope and have limited features. Some don't even require any previous programming knowledge and are often targeted towards newcomers or other non-programmers. While the tools they offer might not be as exhaustive compared to the likes of Unity and Unreal, they are a good gateway to the realm of game development and a solution for someone looking for the most efficient way to build a specific type of game. (Andrade 2015; Rădulescu 2020.) Similar examples of specialist engines include GameMaker, RPG Maker, Adventure Game Studio, and Construct 3.

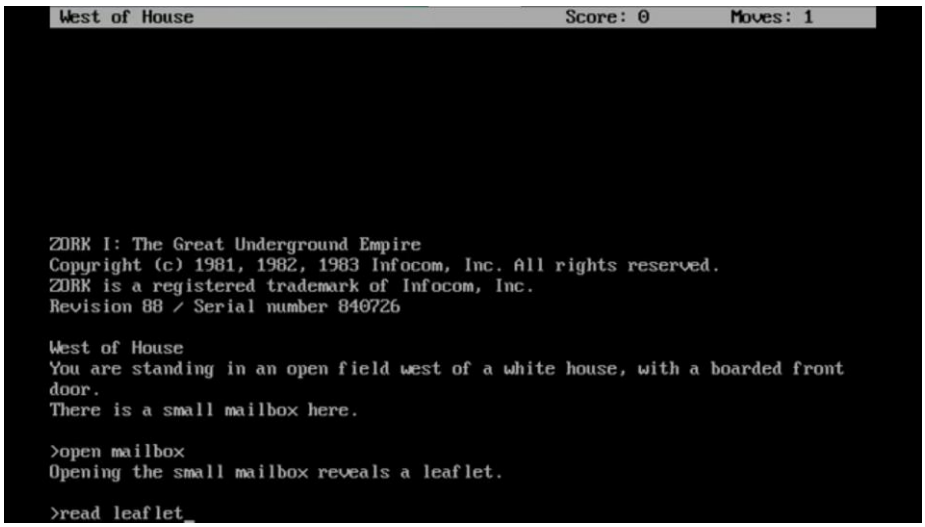
When it comes to the history of game engines, it wasn't until the mid-1990's that the concept of a game engine was born. Although some game development kits existed, in the past, every game had to be essentially coded from the ground up, and very little of the code could be reused for future games. With time, the concept of a piece of software that provided common functionality for multiple games became common practice, especially with the release of Doom and Quake during the 90s. This revolution in how games were made, along with the realization that licensing those engines proved to be an additional revenue stream for game companies, allowed game development tools to become more accessible than ever before. Furthermore, new online distribution methods further empowered indie developers to create innovative and captivating games with limited resources that could compete with AAA titles. (Andrade 2015; Balasubramanian 2022.)

Nowadays, the game engine industry is growing each year. Increasingly powerful tools and features are continuously added and refined, as the demand for games that push the limitations of graphics and gameplay grows. New technologies also redefine how games are created and played. Bilas (2024) argues that AI-driven tools, increasingly powerful and expansive cross-platform experiences, and subscription-based models are some of the biggest trends that are likely to shake the gaming and game engine industries in the following years.

3 THE POINT-AND-CLICK ADVENTURE GENRE

The point-and-click adventure genre is a video game genre in which typically the player can only navigate the environment by controlling their character with the mouse. Themes of exploration, puzzle-solving, mystery, and rich narratives are common. At the same time, action and strategic elements are often excluded completely, and heavy emphasis is instead based on interacting with non-player-characters and objects, combining items, and finding clues to progress in the story. A typical point-and-click adventure game includes one playable character, which the player controls to move in the environment and progress in the story by talking to other characters, finding items, and solving various puzzles. (Vranešević 2014.)

The origins of the point-and-click adventure genre can be traced back to the early text-based adventures, in which players could read descriptions of areas and features and progress the story by writing out text commands such as “Go North” or “Check Inventory” (Picture 1). This was followed by a fusion of text with illustrations, with titles like Mystery House, The Pawn, and eventually 1984’s King’s Quest which allowed the player to move in a graphical environment using directional keys on the keyboard. The final leap - a mouse-driven interface, now known as a genre staple - came in 1984 when Apple Macintosh introduced the concept of a computer desktop controlled by a mouse and keyboard. Over the following decade or so, several classic and hugely successful adventure titles were released: Maniac Mansion (1987), the Broken Sword series (1996-2013), Myst (1993), Grim Fandango (1998), and the Monkey Island series (1990-2022) (Picture 2), to name just a few. (Bevan, Hill, McFerran, Jarratt 2018; Gailloreto 2021.)



PICTURE 1. Zork I: The Great Underground Empire, one of the earliest text-based adventures (Infocom 1980).

PICTURE 2. The Secret of Monkey Island (Special Edition) (LucasArts 2009).

Although the genre was considered by many to be dead after the reputed golden age of the 90s, several successful point-and-click adventure games have been released in recent years. YouTube creator WelfareWalrus (2020) argues that the genre never died in the first place, and it has only been "overshadowed by how massively popular other genres have become". When looking at games released on Steam that have the tag "Point & Click", the number has grown rapidly, as can be seen in Table 1. One reason for this popularity may be the often-simple mechanics of point-and-click adventure games, which makes them relatively easy to create compared to other genres and therefore a popular choice for many developers. Additionally, the COVID-19 pandemic, which essentially confined people

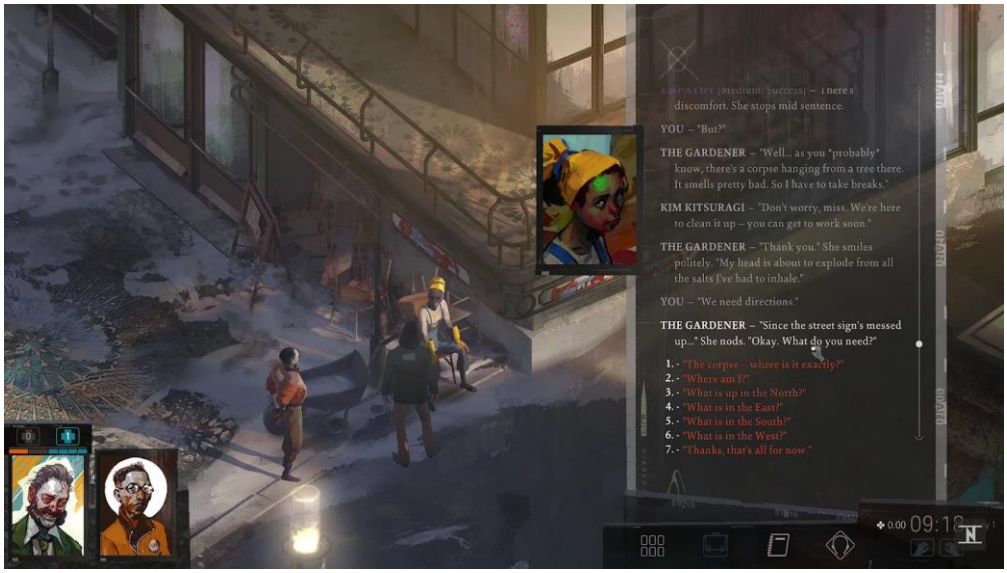
to their homes, might have contributed to this trend, as people had excess time to engage in gaming and game development. Of course, the overall rise in the popularity of gaming and the Steam platform is also a factor to take into consideration.

TABLE 1. Point-and-click games released on Steam by year. (SteamDB 2024a)

Year	**Point-and-click Games**
2023	992
2021	799
2019	285
2017	266
2015	227
2013	39
2011	25

Examples of modern point-and-click adventure games include notable releases like Telltale's The Walking Dead, Disco Elysium, Broken Age, Night in The Woods, NORCO, Life is Strange, and more. Many modern point-and-click adventure games lack some classic features of the genre like puzzle solving or even the simple one-click interface but have instead found a way to modernize and innovate the genre while retaining the core elements of exploration and storytelling (Picture 3) (WelfareWalrus 2020). A shift can also be observed in who is making point-and-click adventure games today. While the classic adventure games were known for big developers like Sierra and LucasArts, the modern point-and-click adventure game genre has been overtaken by indie developers, who find new ways to innovate and build upon the foundations of their predecessors. In Picture 4, a modern point-and-click adventure can be seen.

PICTURE 3. There Is No Game: A Wrong Dimension, a whimsical comedy adventure that requires outside-of-the-box thinking (Draw Me A Pixel 2020).



PICTURE 4. Disco Elysium, a modern point-and-click adventure utilizing RPG elements (ZA/UM 2019).

While the themes of exploration and narrative are often remembered with nostalgia and warmth, the genre is also somewhat infamous for its overtly difficult and frustrating puzzle logic. This phenomenon, known as “moon logic puzzles”, meaning “wildly convoluted or counterintuitive thinking required to understand a seemingly impenetrable puzzle or plotline” (Jones 2021), is a common occurrence in point-and-click adventure games. In Picture 5, a famous example of a moon logic puzzle from Monkey Island 2 can be seen, where a monkey’s tail had to be used as a wrench for a broken pump. Contrarily, point-and-click adventure

games can also suffer from the exact opposite problem; puzzles that are too simplistic or repetitive. As the simple controls and gameplay mechanics only offer so much variation, developers need to be creative in designing their puzzles, while simultaneously maintaining the coherence of the overall story.

PICTURE 5. Monkey Island 2: LeChuck's Revenge, showing the “monkey wrench” puzzle (LucasArts 1991).

Designing puzzles for adventure games can be hard due to their non-linear, sprawling nature, and their inherently simple mechanics. Luckily, the process can be streamlined by using what is called a Puzzle Dependency Chart. Ron Gilbert (2014), who is known as one of the most influential figures in the point-and-click adventure scene and widely recognized for his work on several LucasArts adventure games like the first two Monkey Island titles, as well as coming up with the concept of Puzzle Dependency Charts, describes them as “a list of all the puzzles and steps for solving a puzzle in an adventure game”. According to Noah Falstein (2013), the Puzzle Dependency Chart should be focused solely on the dependencies between the puzzles, not the story. In Figure 1, an example of a simple Puzzle Dependency Chart created for the practical project of this thesis can be seen.

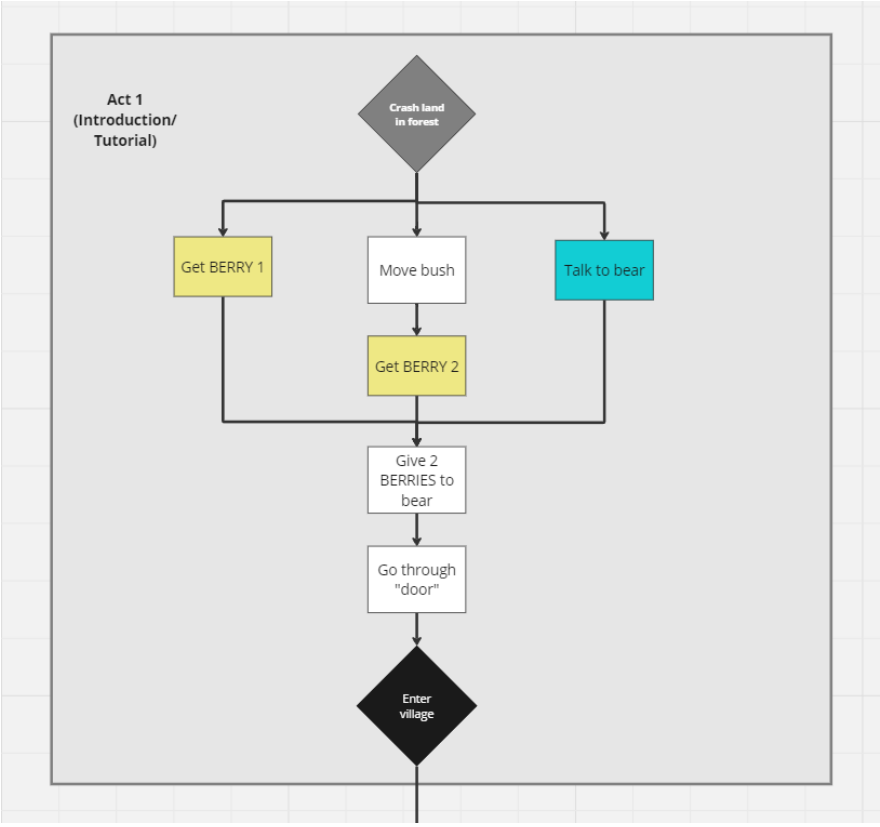


FIGURE 1. A Puzzle Dependency Chart created for the practical project of this thesis.

4 VISIONAIRE STUDIO

Visionaire Studio is a multi-platform game engine primarily designed for the development of 2D and 2.5D point-and-click adventure games. It was developed by German Visionaire Studio UG and founded by Thomas Dibke (Visionaire Studio Wiki 2017). (Visionaire Studio n.d.b.) Compared to more well-known game engine alternatives like Unity, Unreal Engine, and Godot Engine, Visionaire Studio remains a relatively unpopular choice for game developers. However, despite its unknown status in the game development world, it has an active community and receives regular updates.

Based on a predecessor developed between 1996 and 1998 on MS-DOS, the first version of Visionaire Studio was developed as part of a student project and released back in 2001. Over the years, the developer team has constantly increased in size and new features and technical functions have been added and modernized. Despite being mostly popular with indie developers, several commercial and well-received games like The Whispered World, Deponia series, and STASIS, have been released, spearheaded by Hamburg-based developer Daedalic Entertainment. Simultaneously, a dedicated community has gradually grown around the software over the years. (Visionaire Studio 2016.)

Visionaire Studio has support for multiple platforms including Windows, macOS, Linux, iOS, Android, HTML5, PlayStation 4, Xbox One, and Nintendo Switch. The editor itself is available for Windows, Mac, and Linux. (Visionaire Studio Wiki 2023e.) There is a free version of the editor available for download, but it is limited to 10 scenes and has no option for creating builds. To acquire the full version of the program, a license must be bought from one of the five available options, each with its own set of limitations when it comes to development and distribution rights. Prices range from the least expensive single user license at 49€ to the professional license at 500€. All licenses are one-time purchases, but a new license is required for new versions of the program. (Visionaire Studio n.d.c.)

4.1 Design

Visionaire Studio's goal is to meet the genre-specific needs of the point-and-click adventure game genre, providing tools that help to create features typically found in them. At the same time, the software is also geared to be easy to use and intended for visual artists and beginners. (Visionaire Studio n.d.b.) If the user so desires, they can visually design their game without necessarily needing to write a single line of code (Visionaire Studio Wiki 2023e). This can be beneficial for people just starting their journey with game development, who wish to get familiar with basic game development principles first, or for artists who might be unfamiliar with programming.

Development in the Visionaire environment works through something called the action system. Everything that happens in-game, whether that has to do with characters or objects, is typically implemented through actions and action parts. The built-in action parts aim to cover the basic features found in point-and-click adventure games, such as starting an animation, showing dialogue text, or adding an item to the character's inventory. (Visionaire Studio Wiki 2023b.) This system is an easy way to set up a simple scene with characters and interactable objects, but it can also be limiting, as these built-in action parts are focused on such a specific genre.

Despite being geared towards beginners, Visionaire Studio allows for experimentation and personalization for those who do know programming. With the built-in text editor, custom scripts can be typed out in various programming languages, notably the Lua and Illios scripting languages, and these scripts can be utilized to build features that the built-in action parts don't cover. For those who don't know how to program and still wish to have access to more extensive features, many community-made scripts and plugins are available both through the Visionaire Studio Wiki and Discord server, often free of cost. (Visionaire Studio Wiki 2023e.)

While the primary focus of Visionaire is the point-and-click adventure genre, many genres utilizing similar game mechanics can be explored, like interactive visual novels or first-person adventure games similar to Myst. The developer team has

also been working on adding new features that allow for more experimentation, aiming for tools that open the possibility for entirely new genres of games to be developed, like platformers or other physics-based games. (Dionous Games 2024a.) However, as these implementations are still relatively new, there are few examples of these features in use.

4.2 Community

Visionaire Studio has a small but active community. Many indie developers and hobbyists use the engine, but it is not particularly well known outside the point-and-click adventure game community. To compare, at the time of writing almost 42000 games uploaded to the digital video game distribution platform Steam are detected using Unity for development according to the website SteamDB (2024b), while the figure for Visionaire Studio is just below 10. While it is worth bearing in mind that the tracker is not entirely accurate and can only make educated guesses, it can still give an idea of the relatively small size of Visionaire Studio - and its community - compared to its popular alternatives.

Visionaire Studio has a main website, which contains basic information about the software like licensing options, main features of the engine, and a forum, as well as a Wiki, which has an extensive manual with various tutorials intended for introducing new users to Visionaire Studio and its features, created with input from the community members as well as the Visionaire Studio developer team. The Wiki is partly unfinished, with some pages lacking text altogether, but these are primarily pages about advanced features like the Box2D integration introduced in the version 5 update (Visionaire Studio 2020). The Wiki also contains the Released Game Index, a compiled list of released games made using Visionaire Studio. The list, however, works on a self-report basis, so it is not a completely accurate representation of all games made with the engine. (Visionaire Studio Wiki 2023e.)

The biggest community-based activity around Visionaire Studio is focused on its official Discord server, which houses a little under 900 members and has been running actively since 2017 (Visionaire Studio 2022a). While Visionaire Studio

has other official social media accounts, namely Twitter and Facebook, most community activity happens in the Discord server, where people share their projects or plugins, chat, and help other community members with any Visionaire-related problems.

There are a handful of tutorials and guides about Visionaire Studio online. There are a few tutorials on YouTube, notably Hitchhiker's videos in German and William Kruger's in English (Visionaire Studio Wiki 2023g). There is also an unofficial manual made by Dionous Games, an indie game development studio, that is a combination of their own experience working with Visionaire Studio and extensively gathered material from all official Visionaire platforms, including the Lua documents as well as the official Discord server (Visionaire Studio 2022b).

The author can testify that the community surrounding Visionaire Studio is welcoming and relaxed. People are eager to support fellow Visionaire developers and help new community members become familiar with the engine. The Visionaire Studio developer team is in active communication with the community, and there is no doubt the small size of the community has had its hand in helping to keep the atmosphere positive and easy-going.

4.3 Features

Most of Visionaire Studio's built-in features are specifically geared towards the development of 2D and 2.5D point-and-click adventure games. This includes support for different graphical styles, user interfaces, various media files, a particle system, and several programming languages, among others. (Visionaire Studio Wiki 2023e.) While the engine offers a variety of features and even more can be explored through the scripting possibilities as well as the community-made plugins, this section will only highlight the core features that Visionaire Studio offers for the most common adventure game experience.

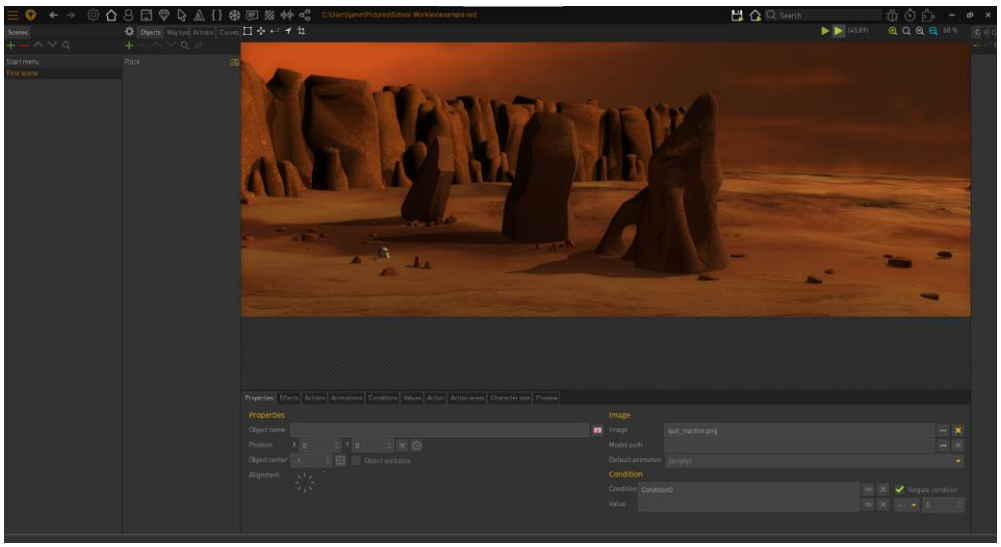
4.3.1 Visionaire Studio Editor

The Visionaire Studio editor is the main view and environment through which development happens in Visionaire. The main sections of the editor are divided into panels and include the viewport, scene list, object list, object properties, and components. At the top of the editor is the project toolbar, which contains links to the various sections of the Visionaire Studio editor including

1. Game Properties
2. Scenes
3. Characters
4. Interfaces
5. Items
6. Cursors
7. Fonts
8. Scripts
9. Particles
10. Texts
11. Shaders
12. Audio
13. Visual Scripting.

There are also buttons for the menu and home views and running the game, as well as tabs for the debug, profiler, and plugin views. The layout of the workspace changes based on the selected scenes and objects. (Visionaire Studio Wiki 2023f.)

There are various customization options to change the way the editor looks and functions. All section panels can be resized to the user's liking, and the workspace can also be completely reset if certain views go missing or the user wishes to set everything to the default view. These customization settings are saved globally, not within individual projects. In Picture 6, the default view of the editor can be seen. (Visionaire Studio Wiki 2023f.)



PICTURE 6. The Visionaire Studio editor with the official default project opened (Visionaire Studio UG 2024).

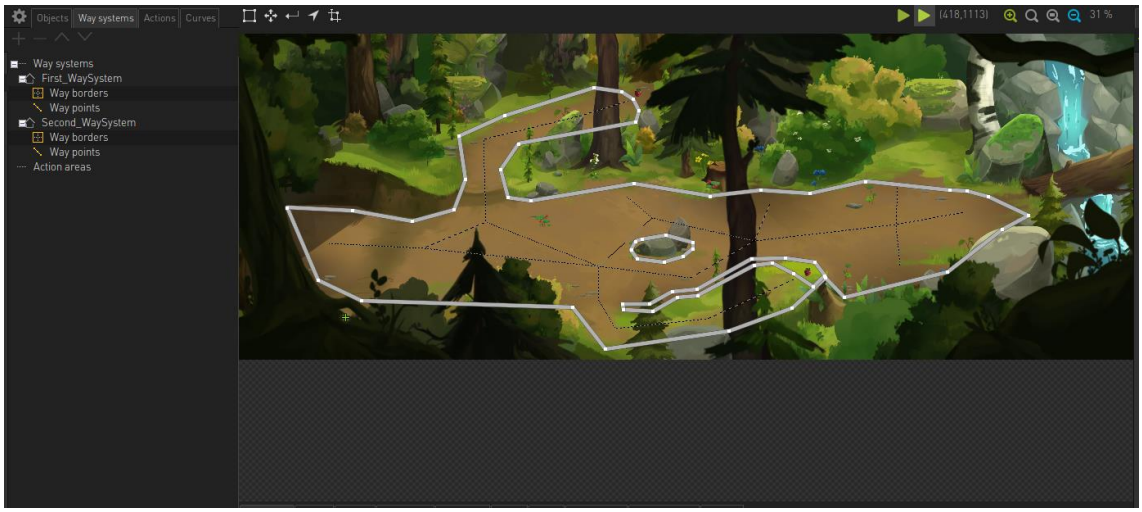
A Visionaire project can be saved either as a VED or VEB file, the latter being notably smaller in size but written in binary data format. All resources related to the project, whether that's graphics, audio files, 3D models, fonts, and so on, must be located in the same folder as the project file to be used in the project. (Visionaire Studio Wiki 2023f.) It is good game development practice to keep up with logical naming conventions and folder structure to make organizing and managing the project easier.

4.3.2 Scenes and objects

Games made in Visionaire are made up of scenes, which serve as the so-called "stages" of the game. Scenes are divided into two different types: scenes and menus, with the latter having no characters. Several scene-related settings can be adjusted through the scene properties window, in which the background image and music among other properties are selected. (Visionaire Studio Wiki 2024h.)

Each scene has a Way System, which Visionaire Studio uses to help determine where in the scene the character can walk. Since Visionaire is a 2D engine and backgrounds are typically made of static 2D images, the engine doesn't know where the so-called "floor" of the scene is. Therefore, a way system is set up to create the feeling of a 3D space. Each way system is made up of a way border

polygon and a way point network. It is possible to switch between different way systems through action parts, which can be used to block or unblock certain areas of the scene. Picture 7 showcases an example of a way system. (Visionaire Studio Wiki 2024h.)



PICTURE 7. A way system in Visionaire Studio (Visionaire Studio UG 2024).

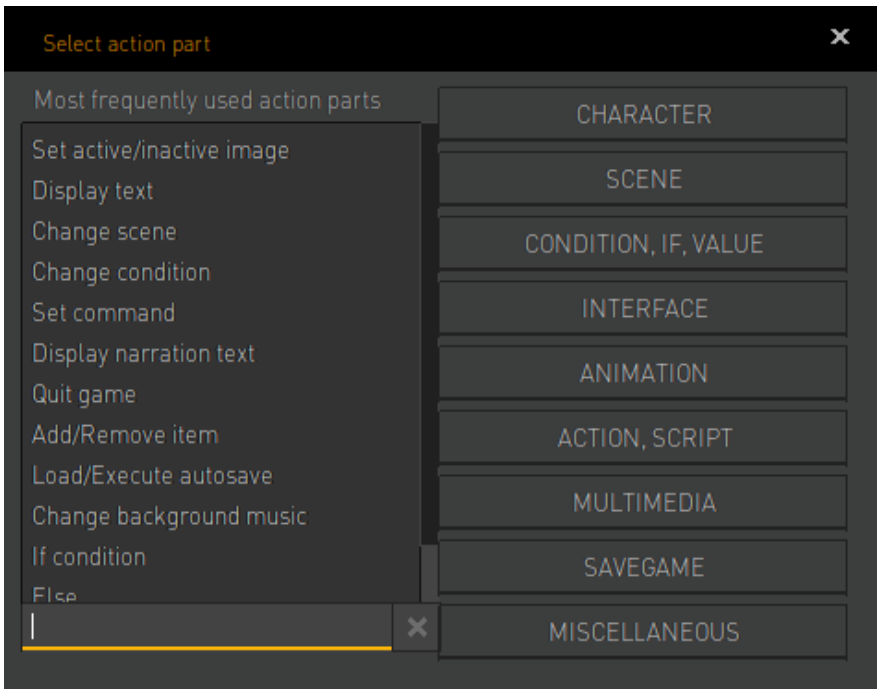
Everything in the scene apart from the background, characters, and interface elements are called objects or scene objects. Objects can have images and animations attached to them, but they don't necessarily have to. They can be interacted with if a hotspot, called the object area, is added to them. It is common to have objects in your scene that have no image, actions, or object area attached to them, but only serve as position information for characters to be placed. Scene objects can be sorted in order of importance, allowing the possibility to layer the playable character behind some objects and in front of others. If a way system is set accordingly, the character can walk both in front of the object as well as behind it. (Visionaire Studio Wiki 2024h.)

It is possible to add additional features to selected objects with components. This can be done via the components tab of the editor. Components are divided into four categories: Internal Components, Shaders, Particle Containers, and Behaviours. The biggest category out of these is Internal Components, which includes features like transform, which lets you scale, rotate, and displace the respective object, text, Box2D, and movie. (Visionaire Studio Wiki 2024h.)

4.3.3 Action system

Development in the Visionaire environment works through something called the action system, which is made up of actions and action parts. These action parts are the basic building blocks of the engine, designed to make adding common features to the game beginner-friendly. Almost everything that happens in the game, be that picking up an item, going through a door, or starting a dialogue, is done through actions. (Visionaire Studio Wiki 2023b.)

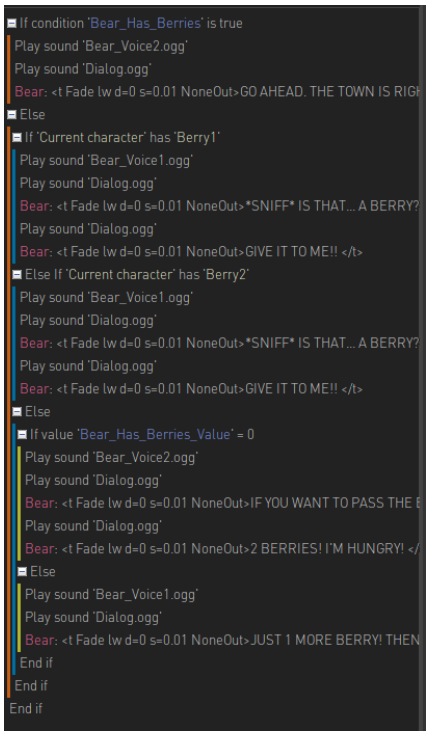
Actions are typically tied to an object that they relate to, and one action can include multiple action parts. Once an action is executed, like clicking a specific object area with the left mouse button, Visionaire goes through the list of action parts inside that action from top to bottom, executing them in order. There is a variety of different built-in action parts to choose from in the action part list (Picture 8). Since actions are tied to their specific objects, there is no one location where all actions can be managed. (Visionaire Studio Wiki 2023b.)



PICTURE 8. The “Action part” pop-up window containing all built-in action parts available to the user (Visionaire Studio UG 2024).

When creating an action, the execution type can also be defined. This is the event that will trigger the action. In most cases, it is a previously created game command like “Look” or “Use”, typically set to the left mouse button, but there are also options for double click, button hold, mouse wheel up, and more. (Visionaire Studio Wiki 2023b.) There are also two special execution types that can only be used with actions tied directly to scenes. These are “At beginning of scene” and “At end of scene”, and they will execute the action immediately as the scene is started or quit, respectively. (Visionaire Studio Wiki 2024h.)

Taking a step further, there are also conditions and values. Condition is a variable that can contain the value of “true” or “false”, and value is an integer-based variable that is a fixed or random number. They can be used to build if statements and structures that determine if an event should be executed based on whether certain conditions are met. These are an essential part of game development both in Visionaire Studio and in general and help to build more complex conditional situations. In Picture 9, an action utilizing conditions and values has been set up. (Visionaire Studio Wiki 2024e.)



PICTURE 9. An action that utilizes multiple different conditions and values to determine what a character will say (Visionaire Studio UG 2024).

Conditions and values can be written in multiple locations in the editor. For organization and management purposes, it is recommended to create conditions in the locations they are related to. However, they can be accessed from anywhere regardless of their location. (Visionaire Studio Wiki 2024e.)

4.3.4 Graphics

Visionaire Studio's primary focus is providing features that support 2D and 2.5D graphics in various ways, from high-resolution graphics to pixel art (Visionaire Studio n.d.a). Many built-in features also help to liven up static 2D images and make environments feel more alive and dynamic: animations, particle effects, shaders, and parallax scrolling, to name a few. Visionaire also supports 3D characters with animations, meaning that the user can combine a 2D background with a 3D character (Visionaire Studio Wiki 2024c).

Visionaire Studio has no lighting system, but scenes can be made to look more dynamic by using lightmaps. Lightmaps are image files that act as a kind of overlay that adds a color tint. They can be set to affect either characters or objects or both. For example, a lightmap can be created so that when a character steps into a visibly darker area of the background, it also receives a darker tint, which helps to sell the illusion of dynamic lighting. However, the flexibility of these lightmaps is somewhat limited, as the whole character or object gets tinted in one color only based on the pixel corresponding to the current position of the object's animation center. (Visionaire Studio Wiki 2024h.)

Visionaire Studio supports resolutions from 320x200 pixels up to 4K. Custom resolutions can also be defined. (Visionaire Studio Wiki 2023e.) In addition, if the selected background image is larger than the game's default resolution, the scene will automatically become scrollable, allowing the user to create large scenes not limited to the boundaries of the screen. This scrolling feature also opens the possibility of adding parallax scrolling to different objects in the scene, allowing them to scroll at a different speed compared to the scene, which can help add a sense of depth. (Visionaire Studio Wiki 2024h.)

4.3.5 Animations

In Visionaire Studio, most visual elements in the game can easily be animated. This includes characters, scene objects, interface buttons, cursors, and inventory items. Animations are typically implemented as image sequences in PNG or WebP file format, but some animations can also be done with object components or the particle system. (Visionaire Studio Wiki 2023c.) Visionaire also supports spine animations for 2D characters (Visionaire Studio Wiki 2024c). The recommended skeletal animation programs for spine animations are Spine and Dragonbones (Dionous Games 2024a).

Animations are adjusted and controlled through the animation editor, which works like a typical animation timeline. In the editor, individual animation frames can be added, removed, and adjusted. Different properties of the animation can be determined as well through the properties panel, like the speed of the animation or the number of times the animation will loop. (Visionaire Studio Wiki 2023c.) Through scripting, it is also possible to access even more advanced animation workflows, like Bézier curves and dynamic frame duration (Visionaire Studio Wiki 2023d).

4.3.6 Scripting

Although Visionaire Studio's most essential features can be accessed without ever needing to write a line of code, the engine has support for scripting to allow users to experiment and go beyond the limits of fundamental features. The main programming language that Visionaire utilizes is a combination of their own Visionaire object model and the Lua scripting language, which is a lightweight, cross-platform programming language. Visionaire object models are used to access the data structure of Visionaire, which can then be manipulated through the scripting language to customize several aspects of the game. Typically, scripts are added in the "Scripts" section of the editor and then accessed via using the "Call script" or "Execute a script" action parts. (Visionaire Studio Wiki 2024i.)

The version 5 update introduced the Visionaire exclusive scripting language called Ilios, which also includes a visual node-based scripting mode (Visionaire Studio 2023). Everything in Ilios is organized around classes that can be attached to objects, characters, interfaces, buttons, scenes, and the game itself through components. At present, the language is not as well documented as Lua and is still under development (Visionaire Studio API n.d). However, Ilios can offer the possibility to develop games beyond the point-and-click adventure genre through the physics-based Box2D integration that was introduced alongside it (Dianous Games 2024b).

Scripting in Visionaire is extensively documented both in the Wiki as well as the LuaDocs, a reference documentation for developers gathering information about the scripting languages of Visionaire (Visionaire Studio Wiki 2024i). In addition, free-to-use Lua scripts made by community members and the Visionaire development team can be found on the Wiki and the Discord server (Visionaire Studio Wiki 2024d).

4.3.7 Particle effects

Visionaire features a fairly straightforward particle system that allows the user to create simple real-time simulations. These particle systems can be used to liven up often static 2D backgrounds and “animate” the scene without having to do frame-by-frame or spine animations for individual sprites. Although certain adjustment parameters are not clearly explained in the engine, the documentation is extensive and covers not only specific adjustments in detail but also explains the basic concept of particle systems. (Visionaire Studio Wiki 2024g).

Each particle system in Visionaire contains an emitter and particles. By adjusting various parameters for both the emitter and the particles, various types of phenomena can be mimicked, like fire, rain, dust, and smoke. While editing, the particle system can be previewed simultaneously, so any adjustments to the system will instantly affect the preview. As with any aspect of game development, performance issues should be taken into consideration, as heavy particle systems can

affect players with slow devices and disturb user experience. (Visionaire Studio Wiki 2024g).

4.3.8 File formats

Visionaire Studio supports PNG or WebP format for image files. Since most Visionaire games rely heavily on 2D graphics for backgrounds, characters, interfaces, and animations, WebP is recommended out of these two due to better optimization and compression possibilities. (Visionaire Studio Wiki 2024f.)

For audio, Visionaire Studio recommends the use of OGG Vorbis and OPUS formats, due to them both being open-source formats with good sound quality and good compression capabilities (Visionaire Studio Wiki 2024b), but MP3 and WAV files are supported as well. There are several places in Visionaire where audio files can be implemented and adjusted according to whether they are attached to specific scenes, objects, or characters. There are also action parts that allow you to play, change or stop sounds, and to adjust their volume and settings (Visionaire Studio Wiki 2024a).

For video files, Visionaire Studio supports the MKV format (Visionaire Studio Wiki 2016). As with audio, video files can be played primarily through an action called “Play video”, in which case the video will proceed to cover the whole screen. Videos can also be started through object components, in which case they act like regular animations and allow for the character to walk in front of and behind them. (Visionaire Studio Wiki 2024h.)

Visionaire Studio also provides support for 3D workflows, although the documentation and support for 3D models are somewhat rudimentary, and models and their animations can be difficult to get working. Several 3D file formats can be imported into Visionaire. To confirm a 3D model’s compatibility with Visionaire, the model can be imported into Assimp Viewer, which Visionaire uses to load its 3D models. (Visionaire Studio Wiki 2023a.)

4.4 Alternatives to Visionaire Studio

This section highlights some alternative options to Visionaire Studio which also have various features specifically supporting the creation of point-and-click adventure games.

Adventure Game Studio, created by Chris Jones and released in 1997, is a free, open-source alternative to Visionaire that similarly specializes in the point-and-click adventure genre. It has support for various resolutions, multimedia formats, and plugins. Although it supports various platforms, the editor itself is solely Windows-based. Compared to Visionaire, the documentation and tutorials available for Adventure Game Studio are more outdated. (Adventure Game Studio 2024.)

Although not specifically developed for the creation of point-and-click adventure games, another option would be GameMaker by YoYo Games. GameMaker has a drag-and-drop visual programming system that is, similarly to Visionaire, catered towards people unfamiliar with coding. It specializes in the creation of 2D games and has support for multiple platforms. For commercial use and Console export, a one-time purchase of $99,99 is required, otherwise the engine is free. (GameMaker 2024.)

There are also various add-ons, plugins, and toolkits available for other engines that are exclusively crafted for the creation of adventure games. For Unity, there is the Adventure Creator plugin made by ICEBOX Studios. It is priced at $80 for a one-time purchase and uses a system called ActionList similarly to Visionaire which makes it possible to create a game without writing any code. (Unity Asset Store 2024.) For Godot Engine, there is the Escoria Framework which can be downloaded for free. Previous knowledge of GDScript, the Godot Engine scripting language, is not mandatory, although it is recommended. (Escoria Docs 2021.)

5 PRACTICAL PROJECT

This chapter describes the process of working on the practical project that was done as a part of this thesis. The goal of the practical project was to develop a simple point-and-click adventure game in Visionaire Studio to showcase in practice how game development in Visionaire Studio works, especially from the perspective of someone new to the software. Advanced features of Visionaire Studio are not mentioned in this chapter as the scope of the game project was kept small and the goal was to examine only the most basic and beginner-friendly features of Visionaire.

The project is an adventure game called Astronaut Crash that has many common point-and-click features: items, puzzles, NPC characters, and a mouse-driven interface. The game was developed primarily using the 5.3 version of Visionaire Studio. Plenty of tutorials, free resources, and help from the community were utilized to tackle problems faced during the development process. The game can be downloaded for free on the itch.io game hosting website and played on PC (Appendix 1).

5.1 Starting the project

The project was started in September of 2023 with initial ideas for the gameplay, story, and visual look of the game, although most of the development happened within a two-month timeframe. Many games in the point-and-click adventure genre were used as points of reference and inspiration, like tiny & Tall: Gleipnir by TT Studios (2018) and Frog's Adventure by Sokpop Collective (2023). Eventually, a concept was settled upon involving an astronaut crash landing on a planet and trying to find their ship by solving item puzzles and talking to several characters. In Picture 10 and Picture 11, concept art of the main character of Astronaut Crash can be seen.

PICTURE 10. Initial concept art for the main character of Astronaut Crash.

PICTURE 11. Color variations for the main character of Astronaut Crash.

After planning the character and the general look of the game, a Puzzle Dependency Chart was created to outline the story, which can be seen in Figure 2. This helped to better understand the number of backgrounds, characters, and items the final game would require. The story follows an astronaut who crash-lands on a planet and goes on a mission to find his missing ship. By interacting with the various creatures living in the forest, finding items, and combining them, the player progresses in the story. Finally, a new default project was started in Visionaire Studio.

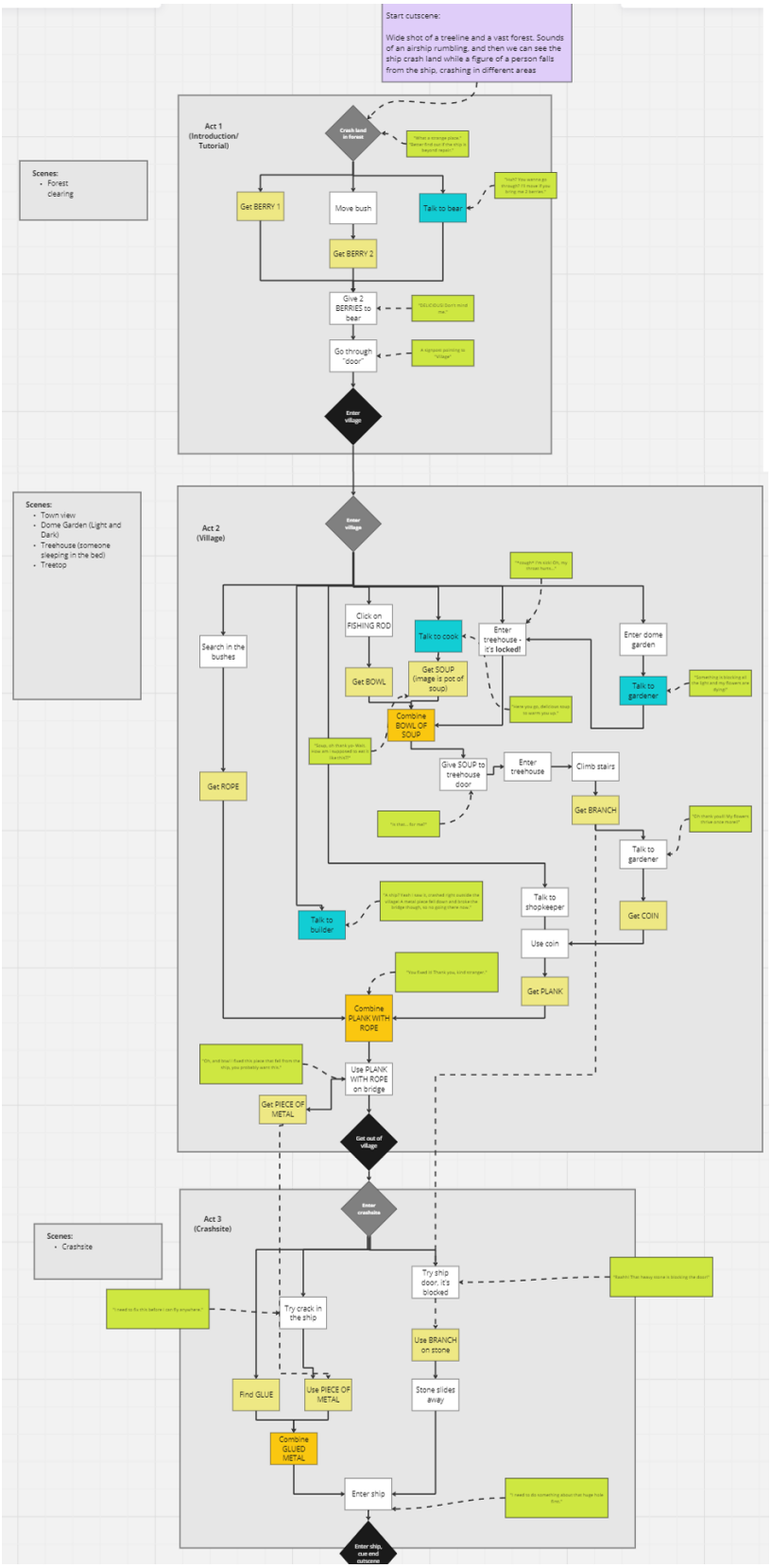


FIGURE 2. A Puzzle Dependency Chart that outlines the puzzles and the main actions of Astronaut Crash.

5.2 Scenes and menus

The first step of development was creating the required scenes and menus in Visionaire Studio. Astronaut Crash consists of one menu type of scene and six so-called regular game scenes – each game scene is a different “place” in the game with its own background. Additionally, there are also two menus that serve as the opening and ending credits, respectively. The development started with working on the game scenes. The only menu scene, which is the main menu of the game (Picture 12), was added later in development.

PICTURE 12. Main menu of Astronaut Crash.

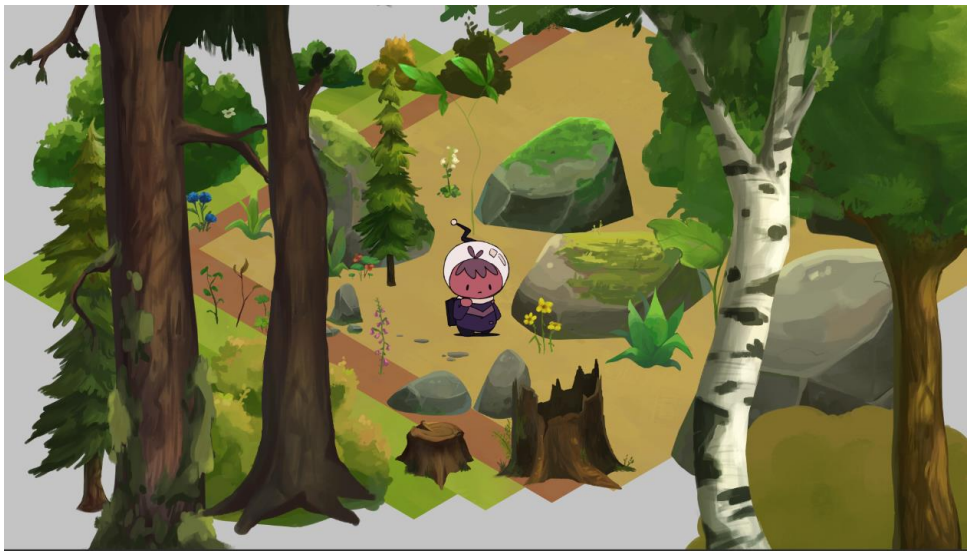
All backgrounds in Astronaut Crash are hand-painted and generally consist of three layers that are

1. the main background image that includes everything the character can walk in front of
2. foreground elements that are always in front of all the other elements
3. objects that the character can walk both in front of and behind, depending on the character’s position.

By dividing background elements like this, a feeling of three-dimensionality was achieved.

To streamline the process of creating multiple backgrounds, several environmental assets were painted separately and then used to create the different backgrounds (Picture 13). This way, one rock or tree could be used multiple times in

a scene without having to paint every single background element separately. This saved a lot of time and helped to create a unified look for all the backgrounds. Still, this painting process took considerably more amount of time compared to setting the backgrounds up in Visionaire. Picture 14 showcases the process from a sketch to a finished background. All the graphics were painted in Clip Studio Paint.



PICTURE 13. Environmental assets painted for the backgrounds of Astronaut Crash.

PICTURE 14. Initial sketch of the town scene and the final background.

Every game scene also includes a way system that determines where the character can walk. In general, the way systems were easy to set up but the way

points that control pathfinding required some fiddling. Additionally, parallax scrolling was added to some foreground elements, but it was quickly determined that the effect didn't work that well with the isometric perspective of the game.

The initial plan was to also create several animations for the backgrounds to help liven them up, especially all the plants and water elements, but unfortunately, this fell out of the scope quite quickly. However, one background animation was made for the first game scene – a simple waterfall animation consisting of three frames. Setting this animation up was straightforward and followed the same process that character animations use, which is detailed in the next section.

5.3 Characters and animations

Astronaut Crash has one playable character and six non-playable characters, also known as NPCs. The playable character is the one the player controls throughout the game, and the NPCs are side characters that the player can talk to, give items to, and receive items from. All the NPCs are static and therefore have no walking animations. In Picture 15, the playable main character and two NPCs can be seen.

PICTURE 15. Screenshot from Astronaut Crash showing the main character and two NPCs.

Creating the characters in Visionaire was very straightforward. Since Visionaire has a separate place where characters are created, it's just a matter of giving them an image or animation, a place in the scene where they are positioned at, and assigning actions to them so that the player can interact with them. As with the backgrounds, the process of drawing and animating the characters took notably more time than implementing them.

All character animations in Visionaire are imported through different categories that make the implementation process relatively easy. These animation categories are walk, standing (idle), talk, random, character, and turn animations. For this game project, only two categories were used: walk and standing (idle). All the characters in Astronaut Crash have an idle animation and the main character also has a walking animation.

Each animation is also divided into four different "directions" called back, front, left, and right. These automatically match the degree of said direction, so when the player clicks to the right side of the screen, the animation assigned with "right" and 0 degrees angle will be played. This, however, can be manually adjusted. Since the main character in Astronaut Crash is facing the camera at a 45-degree angle rather than pointing directly towards the camera, as this can often feel awkward in games, an angle of 45 degrees was defined for the "right" facing animation. All NPCs in Astronaut Crash only have an animation facing one direction, as they do not move from their designated place, but the main character has back and front views for both the walking and the idle animations (Picture 16).

PICTURE 16. Frames from the main character's animations: idle (front), idle (back), walk (front), walk (back).

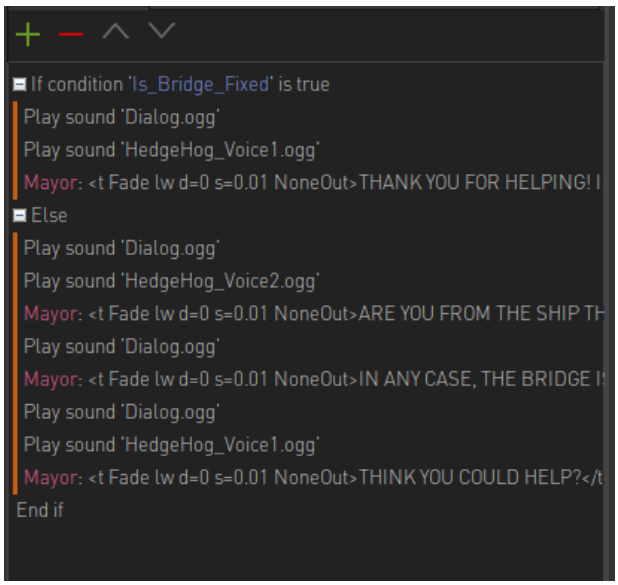
Several other animation settings can also be defined in the animation editor, like the speed of the animation. Animations can also be mirrored with the “Mirror following action” setting, which automatically flips the whole animation sequence and copies the speed of the animation from the original animation sequence. Another good thing is that if any changes are made to a file in the directory, the updates are automatically reflected in the engine without having to re-implement anything.

5.4 Programming

Although no programming was required for the development of Astronaut Crash, this section covers the so-called technical parts of setting up the overall gameplay and flow of the project, mainly by using actions and action parts.

As mentioned previously, action parts are the basic building blocks of Visionaire. These were used to set up any kind of action or change happening in the game. In Astronaut Crash they are mostly tied to characters or certain objects in a scene. In general, the process of setting up action parts was easy. Built-in action parts like “if character has item…” and “if autosave exists…” helped a lot. The process was, however, somewhat tedious. For example, every time a character says something, a dialogue sound plays. This had to be manually added as an action part called “Play sound” for every action. A script would have surely made it possible for the sound to be played any time a character spoke and helped to streamline the process.

Conditions, if statements, and values were also utilized a lot in actions. These were mostly used to determine what a character says based on different conditions. These conditions could be for example whether they already have an item, whether the player is talking to them for the first time, whether a certain puzzle has already been solved, and so on. In Picture 17, a simple if statement can be seen, where an NPC will speak different lines depending on whether the player has already fixed a bridge.



PICTURE 17. An action containing the “If condition” action part.

Perhaps the most challenging part of the project was setting up the Save system. This involved researching how saving in Visionaire works, which was not documented in large detail on the Visionaire Wiki website. After going through the forum and asking for help in the community Discord server, it became apparent that Visionaire has two built-in systems for saving – quick saves and save slots. These two systems have different action parts, and the confusion came from using the wrong action parts for the saving method being replicated. Thankfully, the community is very active, and an answer was obtained in a matter of minutes.

Although no coding was required for Astronaut Crash, some free-to-use scripts created by the community were utilized to polish aspects of the game. These were implemented easily by copying the script and pasting it into the corresponding section in the editor. Two different scripts were used – a script that allowed the camera to continuously follow the main character and a script for creating speech bubbles called Argo Bubbles. For the speech bubbles, manual tweaking to the code was done following instructions created by the script’s creator, which involved adjusting some CSS-based values.

5.5 Interfaces

Many point-and-click games have several what are called “commands” that the player can switch between to interact with the world in various ways. Some typical commands include “Walk to”, “Use”, and “Look at”, and switching between them changes how the character reacts to other characters and objects in the world. This typically also changes the cursor’s visual look.

For Astronaut Crash, a simple interface with a one-command cursor was decided upon. With the same cursor, the player in Astronaut Crash can walk around the scene, talk to characters, use items, click on menu buttons, and so on. It was important that everything in the game could be done using the mouse, so no keyboard commands were required for any part of the game – although the escape key can be used to go back to the main menu.

The game also has other interfaces that get activated depending on certain conditions. For example, the inventory interface that includes slots for items is always visible during game scenes but gets deactivated whenever the player goes to the main menu. All pop-up windows also had to be done as separate interfaces whose activation is controlled through the action part “Show/Hide interface”. These pop-up interfaces include the “Confirmation” interface that is used when the player overwrites a previous save (Picture 18) and the “Credits” interface that shows when the player presses the credits button.

PICTURE 18. The “Confirmation” interface which gets activated when the player is trying to start a new save.

5.6 Polishing

Although there was not enough time to polish the game to the extent that would have been desirable, some things managed to be implemented that enhanced the overall game experience. One of these things was audio. Astronaut Crash uses one music track which was implemented through the settings of the first game scene. For the rest of the game scenes, it was possible to check “Continue music from the previous scene” in the scene settings. This was very convenient as it allowed the music to continue seamlessly from one scene to another without having to implement it again.

For sounds, most were implemented through the action part “Play sound”. This included character speaking sounds, UI sound effects, doors opening, receiving an item, and more. Some were implemented elsewhere. For example, the main character’s walking sounds can be implemented through the character settings, and there is an action part called “Change walking sound” that can be used to switch the sound when the character enters a new scene. Implementing sounds was mostly straightforward but the process was somewhat tedious and slow. The volume of each sound also needs to be adjusted separately, although Visionaire

Studio has built-in audio buses – output channels that can combine several different audio sources to one – that could have solved this problem, but they were not explored for this project.

Since the scope of the game was quite small, there was not enough time to dive deep into optimization. For example, although the WebP format is recommended for all images, this project only uses PNG images. This was mostly done because converting all images to WebP was time-consuming and resulted in a loss of quality, and since the game is quite small in size, it runs smoothly nevertheless. Another optimization option that was not explored in the project was various settings for resolution, audio, and so on, all of which would have certainly enhanced the player experience.

6 CONCLUSION

At the beginning of this thesis, the goal was set to research Visionaire Studio as a game development tool and examine its design, community, and features. Game engines and the point-and-click adventure game genre were briefly introduced to understand what Visionaire Studio is designed for and what the practical game project created for this thesis aims to explore. The goal was to explore the potential of Visionaire Studio and provide a detailed examination of what the general pipeline of game creation with Visionaire could resemble, particularly from the perspective of someone who is an artist or new to game development.

Through various theoretical and practical methods, the thesis gave a detailed picture of Visionaire Studio as a game engine. At times, it was challenging to find a variety of sources and discuss in-depth about a software tool that is not as well-known as many of its contenders. However, in a way, this also became one of the goals of this thesis along the way – could attention be brought to this software and maybe introduce someone to a tool they have never heard of? In the end, the author feels that Visionaire was examined through many different lenses and both its theoretical and practical aspects were demonstrated.

The main takeaway from Visionaire is that it is a very beginner- and artist-friendly engine. For someone just starting their journey with game development, Visionaire's action parts are an excellent way to learn about game logic and an easy way to immediately set up interactions within your game. For an artist, the graphic pipeline is convenient and straightforward, allowing them to focus on visuals. The extensive documentation, although in some places lacking, along with the active community makes the whole learning process easier. From personal experience, it was surprising how little time it took to set up a working game, and as previously mentioned, throughout the whole project notably more time was spent on painting the various game assets than developing the game in Visionaire.

Visionaire Studio is a specialized engine, which makes it very good in its own area of expertise but somewhat limiting in others. For someone interested in creating point-and-click adventures, interactive visual novels, or any other similar

type of game, Visionaire is a fantastic choice. While it is theoretically possible to create other types of games in Visionaire with scripting, the author wouldn't consider it an optimal way to go about developing such a game. It is also good to note that Visionaire Studio is not a free piece of software. For an independent developer or a hobbyist, any kind of cost can be a deciding factor on whether to turn to other options.

As with any software or tool that deals with a subject as broad as game development, Visionaire has numerous features that this thesis didn't get to touch upon. Some of these are 3D models and their implementation, localization, dialogue trees, shaders, particle effects, scripting, and many more. It goes without saying that by exploring these advanced features and community-made plugins, it is possible to create games that differ from the typical and simpler point-and-click experience. Similarly, more problems with optimization would certainly arise were the game bigger and more complicated, which could allow for testing the limits of the engine.

Overall, the game project is deemed satisfactory. There was a lot of room for polishing that could have been done – talking animations for characters, additional NPCs to populate the scenes, background animations, particle effects, UI animations – many things that could have added more detail to the game but there was no time for in the end. Nevertheless, the game works and can be played from start to finish with several items, NPCs, and scenes to explore, and as it stands, it is a good project that allowed the author to get familiar with Visionaire and create a game independently from scratch to finish. The game was also published on itch.io, which hopefully, along with this thesis, inspires others to try their hands at making games, perhaps with Visionaire Studio.

When it comes to the future of Visionaire Studio, it is safe to say that the engine will continue receiving updates that expand its current features and introduce new ones. Hopefully, fans of the point-and-click adventure genre can look forward to Visionaire Studio providing an even more robust and accessible approach to game development and being at the forefront of keeping the genre alive.

To conclude, the author hopes that by studying Visionaire Studio in theory and practice, a thorough coverage of the engine, and a detailed examination of why someone should or should not choose Visionaire as their game development tool was achieved. However, at the end of the day, whatever tool one chooses for their project, it is up to their creativity and problem-solving mindset to get the most out of it.

REFERENCES

Adventure Game Studio. 2024. Home page. Read on 22.5.2024. https://www.adventuregamestudio.co.uk/

Andrade, A. 2015. Game engines: a survey. EAI Endorsed Transactions on Serious Games, 2(6), 150615–150616. https://doi.org/10.4108/eai.5-11-2015.150615

Balasubramanian, K. 2022. Game Engines: All You Need to Know. Gameopedia. Released on 15.9.2022. Read on 24.2.2024. https://www.gameopedia.com/game-engines-all-you-need-to-know-about/

Bevan, M., Hill, J., McFerran, D., Jarratt, S. 2018. The Art of Point-and-Click Adventure Games. Bitmap Books.

Bilas, A. 2024. Game industry trends to expect in 2024. GameAnalytics. Released on 25.1.2024. Read on 24.2.2024. https://gameanalytics.com/blog/gaming-industry-trends-2024/

Dionous Games. 2024a. Visionaire Studio Guide. Read on 13.9.2023. https://www.dionous.com/visionaire-guide/

Dionous Games. 2024b. Ilios Guide. Read on 13.9.2023. https://www.dionous.com/visionaire-studio-guide-ilios/

Escoria Docs. 2021. Welcome to Escoria's documentation! Read on 22.5.2024. https://docs.escoria-framework.org/en/devel/index.html

Falstein, N. 2013. The Arcane Art of Puzzle Dependency Diagrams. The Game Developers Conference. Presentation. Watched on 15.2.2024. https://www.gdcvault.com/play/1017978/The-Arcane-Art-of-Puzzle

Gailloreto, C. 2021. The Rise And Fall (& Rise) Of Point-and-Click Adventure Games. Screen Rant. Released on 14.9.2021. Read on 16.2.2024. https://screenrant.com/point-click-adventure-games-history-lucasarts-myst-remake/

GameMaker. 2024. Home page. Read on 22.5.2024. https://gamemaker.io/en

Gilbert, R. 2014. Puzzle Dependency Charts. Grumpy Gamer. Released on 10.8.2014. Read on 15.2.2024. https://grumpygamer.com/puzzle_dependency_charts

Jones, P. 2021. Moon logic. Haggard Hawks. Released on 28.4.2024. Read on 2.3.2024. https://www.haggardhawks.com/post/moon-logic

Rădulescu, R. 2020. Game Engine Basics. GDQuest. Released on 15.8.2020. Read on 24.2.2024. https://www.gdquest.com/tutorial/getting-started/learn-to/game-engine-basics/

SteamDB. 2024a. Steam Point & Click Game Releases by Year. Read on 15.2.2024. https://steamdb.info/stats/releases/?tagid=1698

SteamDB. 2024b. What are games built with and what technologies do they use? Read on 25.11.2023. https://steamdb.info/tech/

Unity Asset Store. 2024. Adventure Creator. Read on 22.5.2024. https://assetstore.unity.com/packages/tools/game-toolkits/adventure-creator-11896

Visionaire Studio API. N.d. Ilios. Read on 20.1.2024. https://www.visionaire-studio.com/luadocs/ilios.html

Visionaire Studio. 2016. Einführung. Handbuch. Read on 11.10.2023. https://www.visionaire-studio.net/book/visionaire-studio-handbuch

Visionaire Studio. 2020. 5.1 Bugfix 2 uploaded. Forum. Read on 15.11.2023. https://www.visionaire-studio.net/forum/thread/5-1-bugfix-2-uploaded/

Visionaire Studio. 2022a. Official Discord Server. Blog. Read on 16.11.2023. https://www.visionaire-studio.net/news/article/official-discord-server/

Visionaire Studio. 2022b. Visionaire Studio Guide / Manual. Blog. Read on 15.11.2023. https://www.visionaire-studio.net/news/article/visionaire-studio-guide-manual/

Visionaire Studio. 2023. 5.2 1233 Update released. Forum. Read on 15.11.2023. https://www.visionaire-studio.net/forum/thread/5-2-1233-update-released

Visionaire Studio. N.d.a. Engine. Read on 13.9.2023. https://www.visionaire-studio.net/cms/visionaire-functions-english.html

Visionaire Studio. N.d.b. Home page. Read on 6.9.2023. https://www.visionaire-studio.net/

Visionaire Studio. N.d.c. Licenses. Read on 13.9.2023. https://www.visionaire-studio.net/shop/

Visionaire Studio Wiki. 2016. Video Encoding. Read on 2.2.2024. https://wiki.visionaire-tracker.net/wiki/Video_Encoding

Visionaire Studio Wiki. 2017. Meet The Team. Read on 11.10.2023. https://wiki.visionaire-tracker.net/wiki/Meet_The_Team

Visionaire Studio Wiki. 2023a. 3D Models. Read on 7.12.2023 https://wiki.visionaire-tracker.net/wiki/3D_Models

Visionaire Studio Wiki. 2023b. Action System. Read on 13.1.2024. https://wiki.visionaire-tracker.net/wiki/Action_System

Visionaire Studio Wiki. 2023c. Animations. Read on 7.12.2023. https://wiki.visionaire-tracker.net/wiki/Animations

Visionaire Studio Wiki. 2023d. Animation Tips and Tricks. Read on 12.1.2024. https://wiki.visionaire-tracker.net/wiki/Animation_Tips_and_Tricks

Visionaire Studio Wiki. 2023e. An Introduction to Visionaire Studio: Adventure Game Engine. Read on 12.10.2023. https://wiki.visionaire-tracker.net/wiki/An_Introduction_to_Visionaire_Studio:_Adventure_Game_Engine

Visionaire Studio Wiki. 2023f. Getting Started with the Visionaire Studio Editor. Read on 5.1.2024. https://wiki.visionaire-tracker.net/wiki/Getting_Started_with_the_Visionaire_Studio_Editor

Visionaire Studio Wiki. 2023g. Tutorials. Read on 12.10.2023. https://wiki.visionaire-tracker.net/wiki/Tutorials

Visionaire Studio Wiki. 2024a. Action Parts. Read on 2.2.2024. https://wiki.visionaire-tracker.net/wiki/Action_Parts

Visionaire Studio Wiki. 2024b. Audio Encoding. Read on 14.2.2024. https://wiki.visionaire-tracker.net/wiki/Audio_Encoding

Visionaire Studio Wiki. 2024c. Characters. Read on 23.1.2024. https://wiki.visionaire-tracker.net/wiki/Characters

Visionaire Studio Wiki. 2024d. Compiled Index of Lua Scripts for Visionaire Studio. Read on 2.2.2024. https://wiki.visionaire-tracker.net/wiki/Compiled_Index_of_Lua_Scripts_for_Visionaire_Studio

Visionaire Studio Wiki. 2024e. Conditions and Values. Read on 2.2.2024. https://wiki.visionaire-tracker.net/wiki/Conditions_and_Values

Visionaire Studio Wiki. 2024f. Image Encoding. Read on 5.2.2024. https://wiki.visionaire-tracker.net/wiki/Image_Encoding

Visionaire Studio Wiki. 2024g. Particle System. Read on 23.1.2024. https://wiki.visionaire-tracker.net/wiki/Particle_System

Visionaire Studio Wiki. 2024h. Scenes and Objects. Read on 23.1.2024. https://wiki.visionaire-tracker.net/wiki/Scenes_and_Objects

Visionaire Studio Wiki. 2024i. Scripting. Read on 23.1.2024. https://wiki.visionaire-tracker.net/wiki/Scripting

Vranešević, G. 2014. The secret of the point and click adventures: Psychoanalytic point of pointing in a bygone genre. Teorija in Praksa, 51(1), 22–42. Read on 16.2.2024. https://www.proquest.com/scholarly-journals/secret-point-click-adventures-psychoanalytic/docview/1518934369/se-2

WelfareWalrus. 2020. Why The Point And Click Adventure Lives On. Watched on 15.2.2024. https://www.youtube.com/watch?v=a0VRyGVcXng&t=4s

APPENDICES

Appendix 1. Game file

The game can be downloaded for free on itch.io:
https://zaaga.itch.io/astronaut-crash