OAMK
OULU UNIVERSITY OF
APPLIED SCIENCES

Florian Brandsma

EXPEDITION GAME EDITOR

EXPEDITION GAME EDITOR

Florian Brandsma
Bachelor’s Thesis
Spring 2020
Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology, Option of Internet Services

---

Author: Florian Brandsma
Title of the bachelor's thesis: Expedition Game Editor
Supervisor: Kari Laitinen
Term and year of completion: Spring 2020 Number of pages: 36

This thesis handles a personal project: The Expedition Game Editor. It describes the resource problems faced when developing adventure games while offering the editor as a solution.

The aim of the project was to provide a solution for affordable adventure game development. The editor allows developers to create adventure games without the assistance of specialists using built-in assets and game logic.

The production of the editor is still in progress, but it has reached a state where the initial phase of the game logic has been proven to work. The editor alleviates some of the problems in adventure game development, but it is not without its own limitations. Developers are restricted to only the built-in assets and logic, potentially limiting their creative vision.

In the future, the editor can be used to create simple and complex adventure games.

Keywords: game editor, game design, game development, software development, unity

PREFACE

I graduated as a Game Designer from Deltion College, the Netherlands, in 2014. Since then I have attempted to put my mark on the gaming landscape by creating an adventure game, inspired by the games I used to play when I was younger.

I had spent two years working on my game before deciding not to continue with it. Recreating the vast amount of content of an adventure game is outside the scope of what a single person can achieve within a reasonable amount of time. The plan was to salvage parts of what I had made and create a more contained adventure game.

A faction in my game world was called *The Expedition*, a group of adventurers who trekked through a vast desert. I chose to create a game about them, as a desert setting would require a limited amount of assets to make it believable and would therefore be possible to create alone or with a small team.

This project too became more than it was meant to be due to the potential I saw in it. But this time I knew that if I were to create a game of this scope, I would need the right tool: a game editor.

The Expedition Game Editor is an ongoing personal project which started in the summer of 2016 and entered production in the spring of 2018. The project files can be found in GitHub by searching the name of this author.

I would like to thank Erwin van Dieren for providing the textures for the character models found in the current version of the editor and Lukasz Holubowicz for his continued support and testing throughout the production phase of the project.
Secondly, I would like to thank all the lecturers at Oulu UAS, for without their lectures realizing this project would not be possible. Lastly, I would like to thank my supervisor Kari Laitinen and language teacher Kaija Posio for their guidance in writing this thesis.

Oulu, 19.03.2020
Florian Brandsma

CONTENTS

ABSTRACT 3
PREFACE 4
CONTENTS 5
VOCABULARY 6
1 INTRODUCTION 7
2 ENGINE VERSUS EDITOR 8
- 2.1 Unity 9
- 2.2 Expedition Game Editor 10

3 FEATURES 12
- 3.1 Assets 13
  - 3.1.1 Items 13
  - 3.1.2 Interactables 15
  - 3.1.3 Regions 18
  - 3.1.4 Scenes 21
- 3.2 Stages 22
  - 3.2.1 Chapter 22
  - 3.2.2 Phase 24
  - 3.2.3 Quest 25
  - 3.2.4 Objective 26
  - 3.2.5 Interaction 27
  - 3.2.6 Time 31

4 LIMITATIONS 32
5 CONCLUSION 35
REFERENCES 36

VOCABULARY

AAA	Triple-A, a classification used to describe high budget games
Asset	A local resource used in game development
NPC	Non-playable character
UI	User interface

1 INTRODUCTION

Making a game is hard. Making an adventure game is even harder. Making an adventure game that competes with AAA giants, such as Blizzard's World of Warcraft or CD Projekt Red's Witcher games, alone or with a small team and with limited resources, is impossible.

Adventure games are almost by definition large. There is little adventure to be had in the confinement of a single room, compared to an entire world. Making such a world takes vast amounts of time and resources. The Witcher 3 cost $81 million to make over the course of 3.5 years by a team of 240 people, with 1,500 people being involved in total (Chalk 2015).

The solution to this problem is using the right tools. Developers have been using proprietary editors for decades, as Blizzard has for World of Warcraft with WowEdit (Staats 2019, 224). When these developers are unwilling to share their editors or if the licence fees are too high, one must make their own editor.

This thesis handles the Expedition Game Editor, a work-in-progress adventure game creating and editing tool.

The editor allows developers to create adventure games without the assistance of specialists. This will in turn alleviate the obstacles described above and greatly reduce the time and resources normally required to make a game of similar scope.

This thesis will cover the logic, existing and planned features of the editor and how they can together produce adventure games of any size.

2 ENGINE VERSUS EDITOR

Knowing the difference between a game engine and an editor can be difficult. Both are used to make video games, but one is quite unlike the other. A game engine is a framework for game development, but not all frameworks are engines.

An engine consists of several sub-engines that handle physics, rendering, object collision, audio, lighting, animation and AI. They are capable of importing files from external sources for visuals and audio and they allow developers to create scripts to create game logic. A debugging tool is provided by the engine to aid troubleshooting during the development process. Some game engines, such as Unity, come with built-in scripts to control other parts of the engine (Unity 2020a).

A game editor is a graphical UI that can be used on top of a game engine to help streamline the development of a game. Editors are tailored to create specific types of games. Like game engines, an editor comes with a scene editor where developers can place assets. The editor then communicates with the engine to construct the game.

The difference between the two is that the editor is more restrictive, following a set of rules specific for the type of game it is designed for whereas an engine is more liberal in its capabilities. An editor typically does not allow the creation of scripts and while this adds more restrictions, it enables the creation of games without programmers.

Developing a game editor can be a costly and time consuming endeavour and it is not worth doing for many game projects. For the massively-multiplayer online roleplaying game World of Warcraft, Blizzard developed their own editor, WowEdit. WowEdit allowed game designers to craft the World of Warcraft almost entirely independently from programmers, creating up to 99 percent of the original game’s abilities (Staats 2019, 224).

Blizzard has demonstrated the value of using a game editor for large-scale, long term game projects. The editor that was used to build World of Warcraft over fifteen years ago is still used to maintain and create new content for the game.

2.1 Unity

Unity is a game engine that is developed by Unity Technologies. While it is largely used to make games, it is also used in other industries such as automotive, film, architecture and gambling (Unity 2020b).

In 2016 it was enjoying a 45 percent share of the global game engine market, ahead of its closest competitor at 17 percent (TNW DEALS 2016). This means that there are vast resources available on the Internet for developers, ranging from documentation and tutorials to frequently asked questions and answers. Many solutions to problems that users will run into are readily available online.

Unity obtained its popularity by its initial accessibility. Ever since the engine was released in 2005 it had always had the option to be used for free, whereas its competitors, e.g. Epic Games' Unreal Engine previously charged users upfront (Sweeney 2015). Only when products made with Unity result in revenue earnings of more than $100k for individuals and $200k for businesses, will users have to start paying fees (Unity 2020c).

Paired with its accessibility is also the ability to deploy products on up to twenty-five different platforms with a single click (Unity 2020d). As of today, 50 percent of games across these platforms are made in Unity (Unity 2020e).

The reasons above have contributed to the decision to develop the Expedition Game Editor in Unity.

2.2 Expedition Game Editor

The Expedition Game Editor is an application that is being developed in Unity. The idea originated out of the desire to be able to create and maintain large-scale adventure games with limited resources.

The editor application is one half of a broader Expedition system. It is used to write information to a database. That information is read by the other half: a game application. The game is constructed using information from the database that was entered by the editor.

Entire games can be made in the editor, using only built-in assets. Developers can create characters, enemies, abilities, items, interactions, quests, scenarios and entire worlds, all with relative ease in a simple UI and without programming or the assistance of specialists.

When the editor is ready, it can be used on both desktop computers and mobile devices. Games made with the editor can be deployed on all platforms supported by Unity as described earlier in Section 2.1.

Logic

The editor can be used by developers to create games without programming. Programmers are responsible for creating a game's logic, but the use of the editor does not require any. The editor has been designed with a certain logic in mind, which has been pre-programmed into the system. This logic can be observed in Figure 1. The way in which the logic operates is best explained in the context of the editor's features.

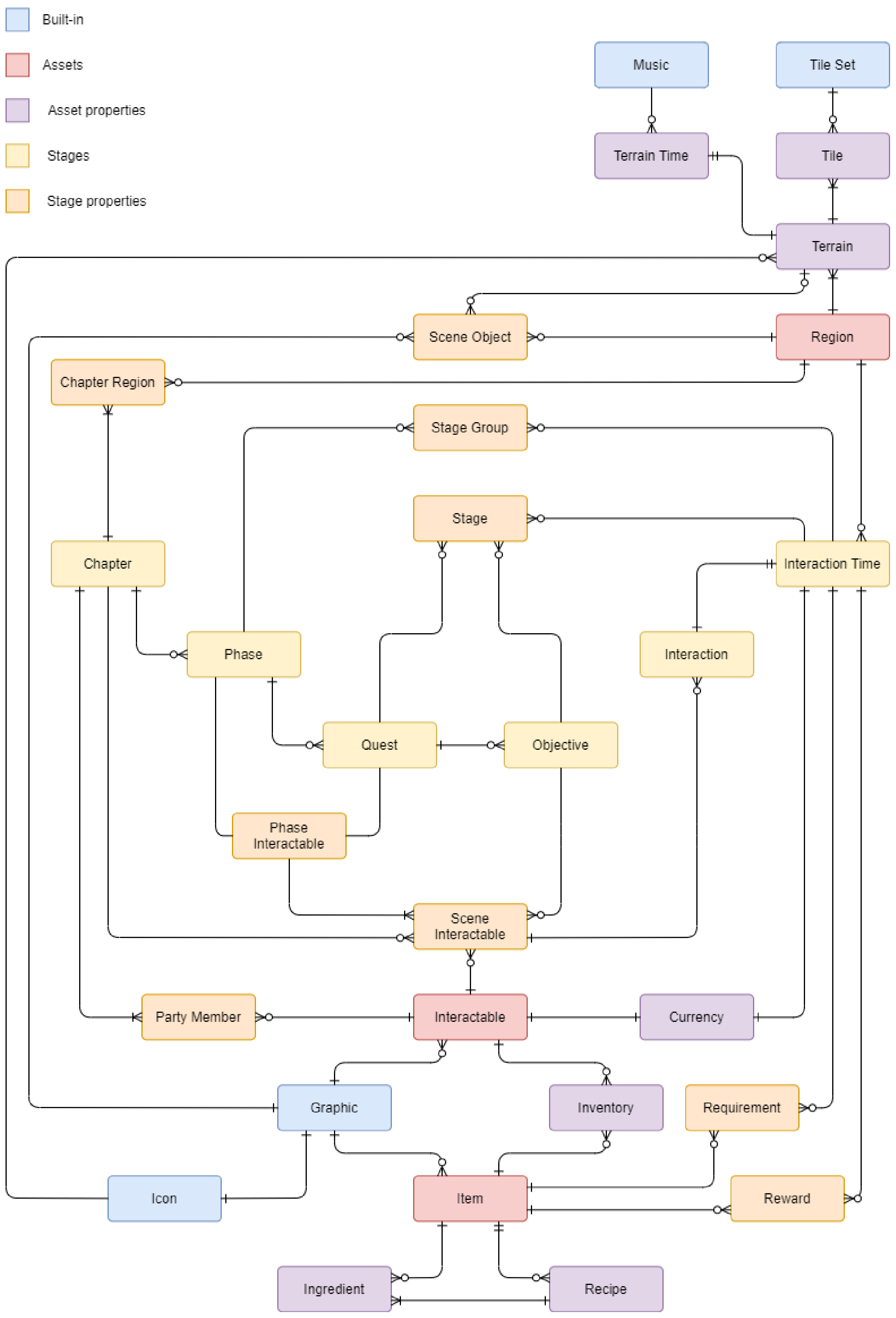


FIGURE 1. A diagram of the Expedition system logic

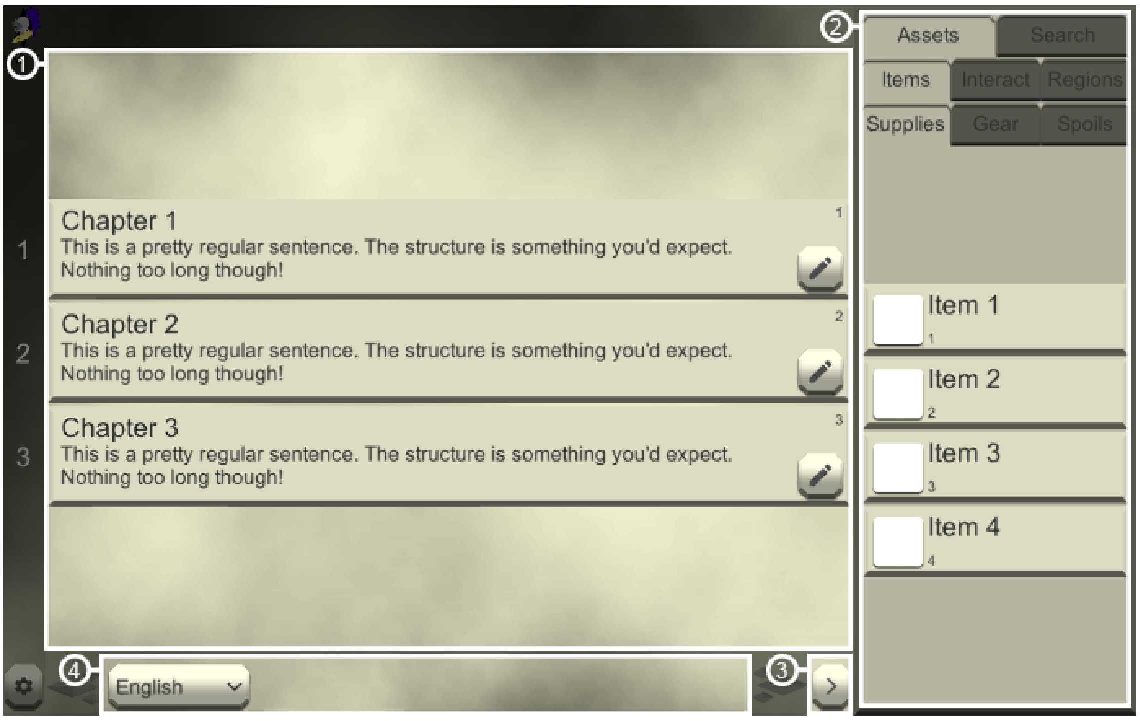
3 FEATURES

General

The Expedition Game Editor’s features are centred around the logic introduced in Section 2.2. While the logic is complicated, the editor’s features are designed to be minimalistic so that it can even be used on mobile devices.

The general UI layout can be seen in Figure 2. It is divided into four sections:

1. The main view displays all the relevant information of the current selection and is also the place where all the editing happens.
2. The asset window displays a list of assets depending on the selected tab.
3. The arrow button minimizes the window it is paired with. In this case, the asset window.
4. The action bar is dedicated to commands and shortcuts that are different for every sub-editor.



1 Chapter 1
This is a pretty regular sentence. The structure is something you'd expect. Nothing too long though!

2 Chapter 2
This is a pretty regular sentence. The structure is something you'd expect. Nothing too long though!

3 Chapter 3
This is a pretty regular sentence. The structure is something you'd expect. Nothing too long though!

FIGURE 2. The home page of the Expedition Game Editor

3.1 Assets

A game produced by the editor is comprised in large part of assets. The assets in the editor are pieces that are created by a developer that can then be used to give context to the game. Many parts of the editor require assets for input, and so they are the cornerstone of the system.

There are three types of assets: items, interactables and regions.

3.1.1 Items

Items are goods that can be collected in the game. They can be used to upgrade characters, to restore health points, to craft other items or to progress in the game by passing certain requirements.

They are divided into three categories: supplies, gear and spoils. Categories are made to help clarify an item's purpose in the game, as well as being able to sort items by category in an inventory.

Some features are shared by all item categories. All items have a name and a graphic to help identify it, as well as an index. A graphic provides both an icon and a 3D model, which can be seen in the main view of Figure 3. The index is also used for sorting the items in both the editor and inventory. Furthermore, all items come with some general properties, such as a monetary value, for which they could be purchased from or sold to a shop for ingame currency. They also have a weight value, which is used during movement calculations, and a short description, which gives further context to the item itself.

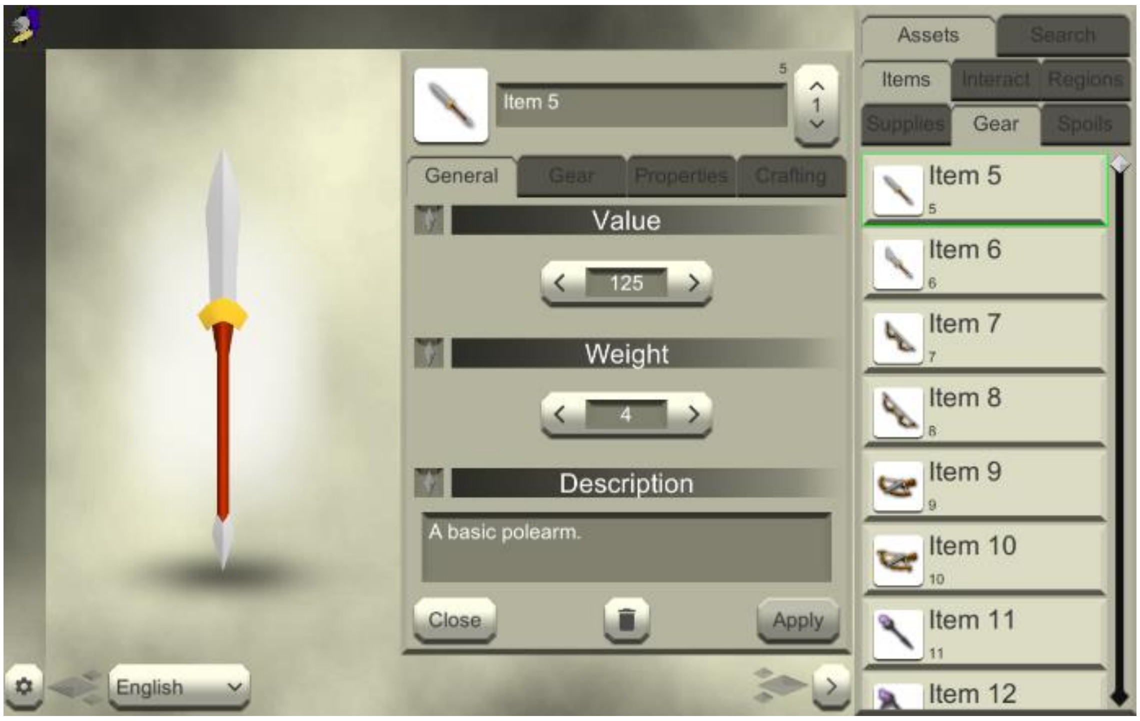


FIGURE 3. A selected gear-type item being edited in the item sub-editor

Supplies

Supplies are items that can trigger an effect when used. The effect is created by chaining several commands. Commands answer what effect does, to whom and how much.

A supply item could be used to restore or decrease a target's health, offensive or defensive capabilities. The target can be either the player character itself or an NPC. It can also be specified if the item will be consumed when it is used, as would the case for edibles.

A description can be written for the effect which can be read in the game. In this way, players can easily tell what the item does when it is used.

Gear

Gear is a type of item that can be equipped by an interactable, with the purpose of manipulating that interactable's attributes. Like supplies, gear can also have an effect. The effect is triggered when the interactable strikes or is struck by a target, depending on the command chain.

A piece of gear can be equipped in one of several gear slots and every piece must be assigned a slot it could occupy. These slots are the main-hand, off-hand, head, body, legs, hands and feet. When gear is equipped, it can provide basic offensive and defensive attributes to the interactable. Offensive and defensive attributes have a primary value that provide offensive and defensive power. The secondary values are used for elemental bonuses, such as frost (cold) and fire (heat) bonus damage and resistance. Enemies that are encountered in the game might also benefit from these bonuses and should be considered when engaged in game for a strategical advantage. Simultaneously, heat resisting equipment might be beneficial outside of combat as well in the journey across the desert.

There are no restrictions to the graphic that is used to represent the item or the slot it can be equipped to. This means that it is possible to equip a sword onto your head slot, for those extra offensive attributes.

Spoils

Spoils are basic items whose main purpose is to be used as materials to craft other items or to sell to a shop for in-game currency. They are intended to be obtained from defeating enemies, gathered in the world, bought from in-game shops or from scrapping gear.

Recipes

A recipe is a collection of items required to craft another item or obtained from scrapping one. Supply and gear items can be crafted by combining other items according to their recipe. Scrapping is the opposite of crafting, where an item is consumed in return for other items according to the recipe. For every item in the recipe it is possible to allocate the amount required by it or obtained from it.

3.1.2 Interactables

Interactables are entities in the game that can be interacted with. They are used as a base for playable characters (also known as party members) and scene interactables. The latter are the kinds of interactables the player can interact with in the game world.

They are the most widely used asset in the editor and as such need to be easily identifiable. Like items, it is possible to give an interactable a name and a graphic. Interactables have also a description that helps define them outside of the player's interactions with them. One difference between items and interactables is that while items are categorized by default, an interactable's category must be selected manually.

Interactables are not only limited to characters but may also include objects, such as chests, which award the player with treasure upon interaction, or an apple ready to be plucked from a tree.

What sets an interactable apart from a regular object and makes them unique are their attributes. Not all interactables will be happy to see the player character and may engage in combat. All interactables have survival attributes which will come into play in these situations.

Survival attributes include health points and when these points drop to zero, the interactable loses. Offensive and defensive survival attributes further contribute to this outcome. These attributes are the same as those on gear. Attributes can also help define an interactable's character. Some may be more resistant to low temperatures and therefore have a higher frost resistance value. The sub-editor showing off some of the attributes can be seen in Figure 4.

All interactables can equip gear to enhance their base attributes. It is unlikely, but not impossible for enemies, such as wild animals, to wear equipment. The base values play a more vital role in that regard.

Other survival attributes also provide some realism to a desert-based adventure aspect of the game, such as hydration values and closely-knit movement speed and weight values. The heavier a character is, the slower and thirstier it becomes. Running out of hydration points may have serious consequences. Carrying many items could further bog the character down, further decreasing its movement speed. When either health or hydration goes down, supplies could be used to restore them as demonstrated earlier in Section 3.1.1.

Inventory

Every interactable has an inventory in which items can be stored and used. When an interactable loses, it is possible to loot them for the items they held in their inventory. The quantity of each item can be set by the developer, as well as the chance of it appearing when looting. It is common in games to have enemies award the player with a specific item upon defeat. All these same enemies have a small chance to award the player with a rare item. While all have the potential to award the item, the chance of them doing so must be set by the developer.

Not only can interactables carry items, but they can also equip gear type items from their inventory. Doing so will increase the interactable’s base attributes with that of the equipped gear.

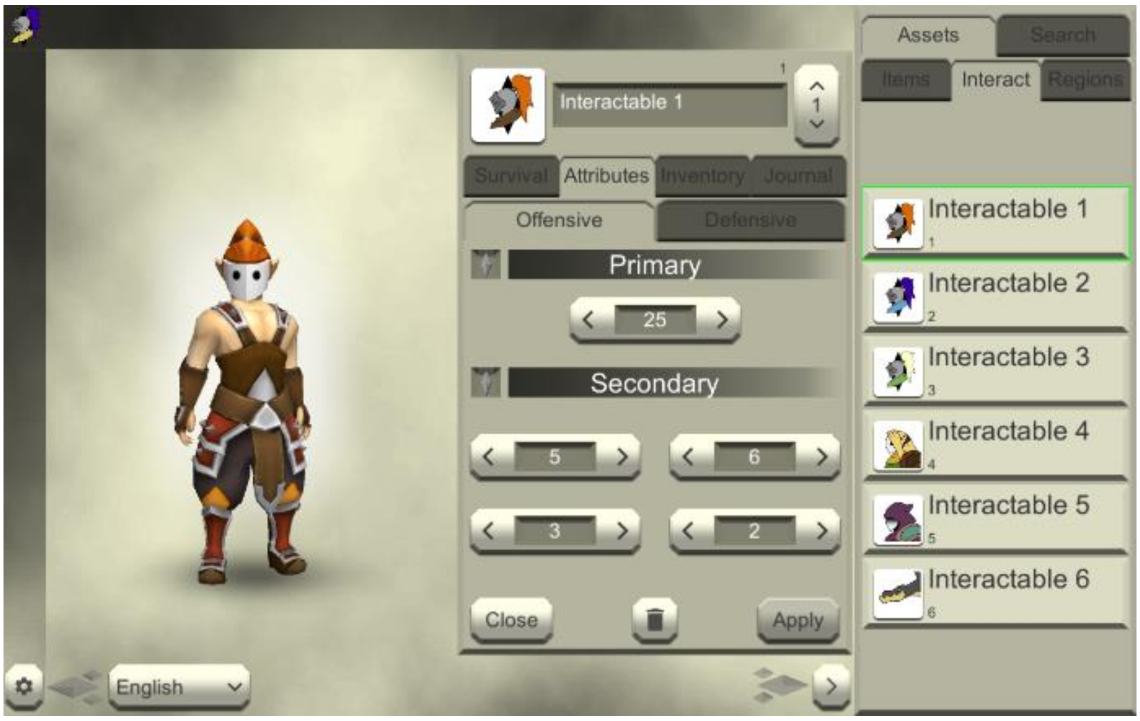


FIGURE 4. A selected interactable being edited in the interactable sub-editor

3.1.3 Regions

Regions are collections of terrains on which the game is played on. Every region has a name to help identify it in both the editor and game.

Every region's environment is determined by the tile set. Where regions are made up of terrains, a terrain is made up of tiles, for both the performance of the game and ease of development. The tile set, as the name implies, is a set of tiles that can be used by each terrain throughout the region. Tiles can be thought of as pieces to a puzzle, where some pieces fit, and others do not. They fit when their borders share the same colour. An example of a tile set and how the individual tiles are pieced together can be seen in Figure 5.

Terrains

Terrains are collections of tiles. They have their own properties to influence their atmosphere, making the region as an entirety more interesting. Every terrain can be identified with a name and an icon. The name is to help identify it in both the editor and game, whereas the icon only helps identify to it in the editor. Various properties further define the terrain. One such key property is the music that plays in the background when the player finds itself on that terrain within the game. Some properties are not only atmospheric like but also impact gameplay.

Weather conditions play a crucial role in any expedition, especially when the greatest obstacle is the environment itself, which is why it is possible to change the general weather conditions and wind per terrain. The weather conditions mainly describe the temperature, cloudiness and precipitation. Wind is described in kilometres per hour and direction, which can act as a great boon or barrier to the player. As detailed in earlier Section 3.1.1, equipped gear can have a great impact on these conditions. If the player enters a frozen cavern, it is wise to equip some gear with a higher frost resistance.

Weather conditions can vary greatly depending on the time of day and as such, these values can be changed for both day and night-time.

Tiles

Tiles belong to the tile set that is chosen for the region. Developers can place tiles based on the dimensions of the terrains and region. All tiles have built-in properties that describe the tile's main material and hardness, which affect an interactable's movement speed over it. A walking interactable carrying many kilos of goods will have a harder time walking through sand than a bird would fly over it. Hardness works together with the weather to create environmental hazards. If the hardness of the tile is low and the main material is sand, a strong wind will create a sandstorm.

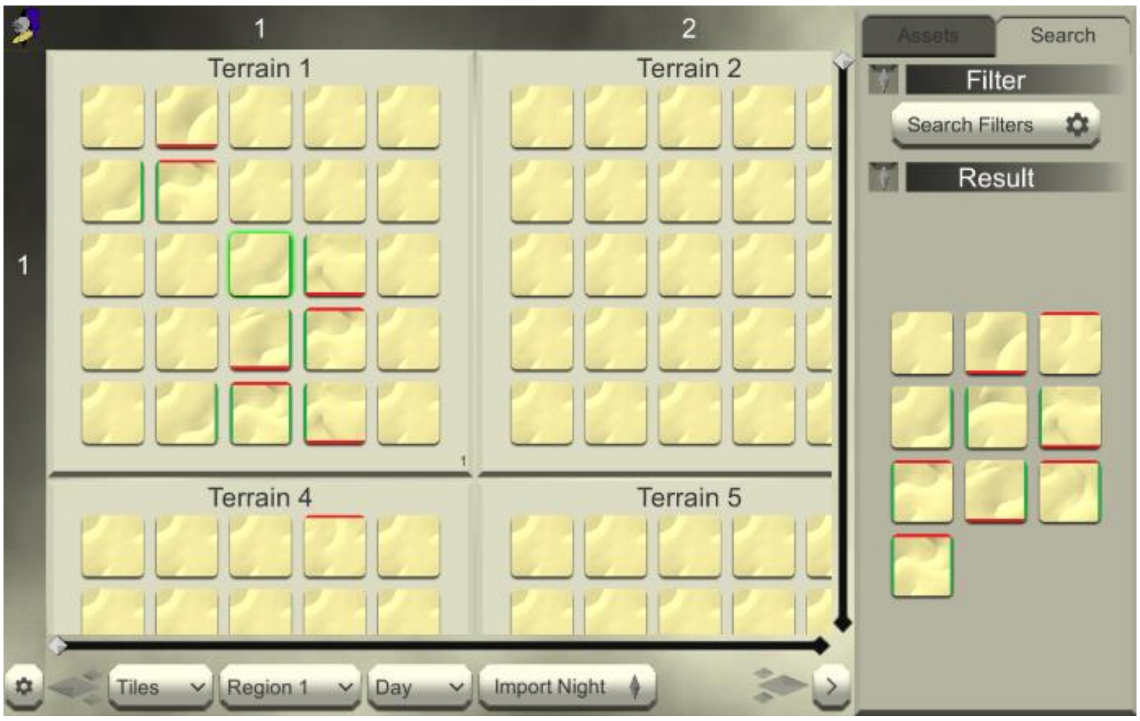


FIGURE 5. A tile being picked from a tile set in the tile sub-editor

Expansion

The game that the editor is initially designed for will be about the Expedition's journey through the desert. The journey through the desert will provide them plenty of challenges. One of the greatest challenges is returning to familiar grounds after getting lost in a seemingly endless desert. The reason for regions and terrains to ultimately be square, as well as the use of tiles, is to support the region expansion feature.

When a player approaches the border of a region, there would normally be a barrier to prevent them from continuing. In the region sub-editor, it is possible to enable region expansion in all directions: north, east, south and west. When the player approaches the edge of any of the enabled directions, a new region will be loaded on that location. What region will be loaded is decided by the repetition properties, which can be seen in Figure 6. These properties serve to provide an intentionally confusing and yet believable experience for the player, making them feel lost in a hostile environment.

The use of tiles is beneficial when creating randomized regions. A special tile is required in this situation to ensure that all pieces will fit together, no matter their placement.

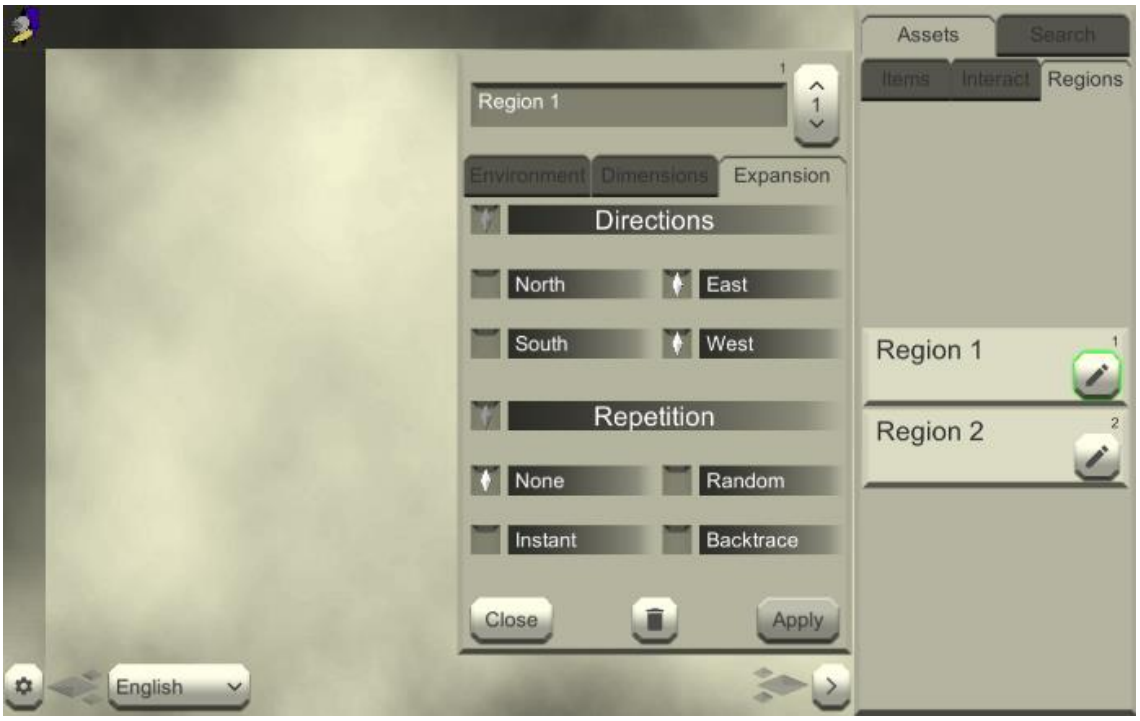


FIGURE 6. Expansion settings of a selected region in the region sub-editor

3.1.4 Scenes

A scene is a collection of everything bound to a region. All the tiles, terrains, scene interactables and scene objects belonging to that come together in the scene. The result of this can be seen in Figure 7.

Scene objects serve as decoration while the scene interactables give players something to do, separate from any progression they have made within the game. An example of such an interaction is to be able to pluck an apple from a tree, as mentioned in Section 3.1.2. If the player sees an apple in a tree, they can pluck it regardless of what stage in the game they are in.

Objects and interactions can be modified in the scene editor, depending on the method by which the scene editor was opened. Their Transform values include a position, rotation and an angle in 3D space. Also included is the object's scale, which is multiplied by its linked graphic's built-in base size. Objects can also be bound to a tile, meaning that they will only be activated when the tile is also active.

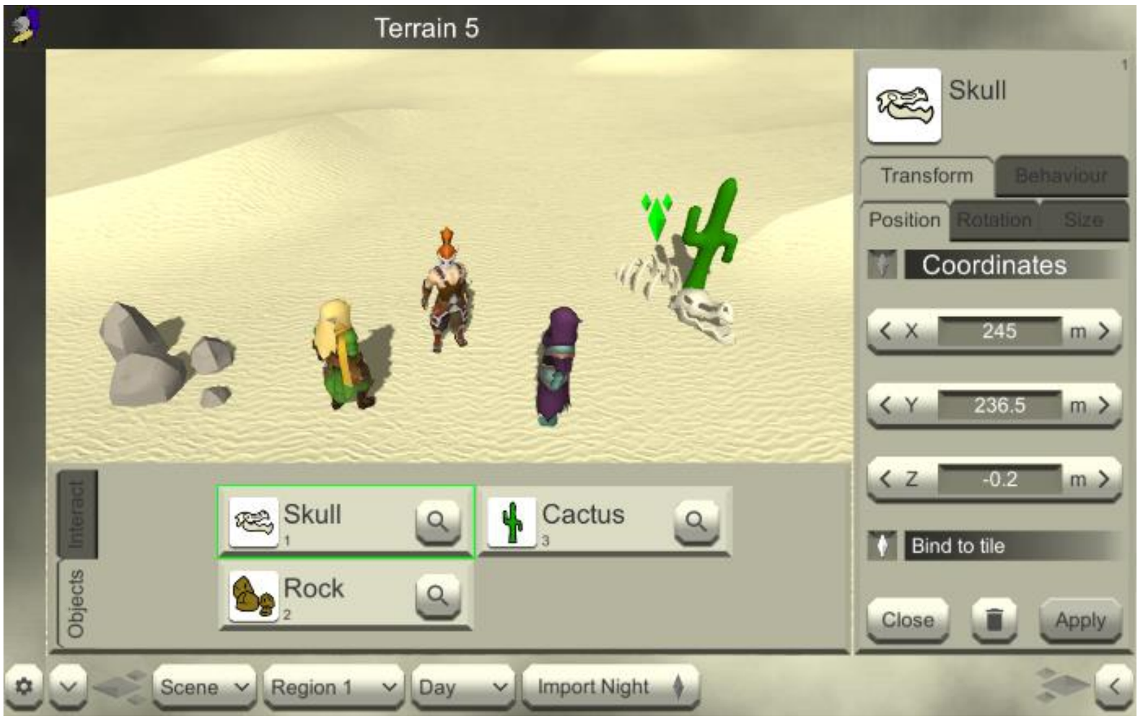


FIGURE 7. A selected scene object being edited in the scene sub-editor

3.2 Stages

Many adventure games are static. The game world never changes, resulting in shopkeepers standing on their feet until the end of time, waiting for the player to interact with them. Time does not affect this world, and most of its inhabitants are often unfazed by the player's actions.

The editor allows developers to create adventure games of various complexities. While it is entirely possible to create the simple games mentioned above, it is also possible to create more intricate and believable scenarios. This is achieved by dividing the game's structure in multiple branching stages, each representing a moment in time. Stages are made up of chapters, phases, quests, objectives down to time-based interactions. In the editor, stages are listed in the main view and can be edited by pressing the edit button as seen in Figure 8.

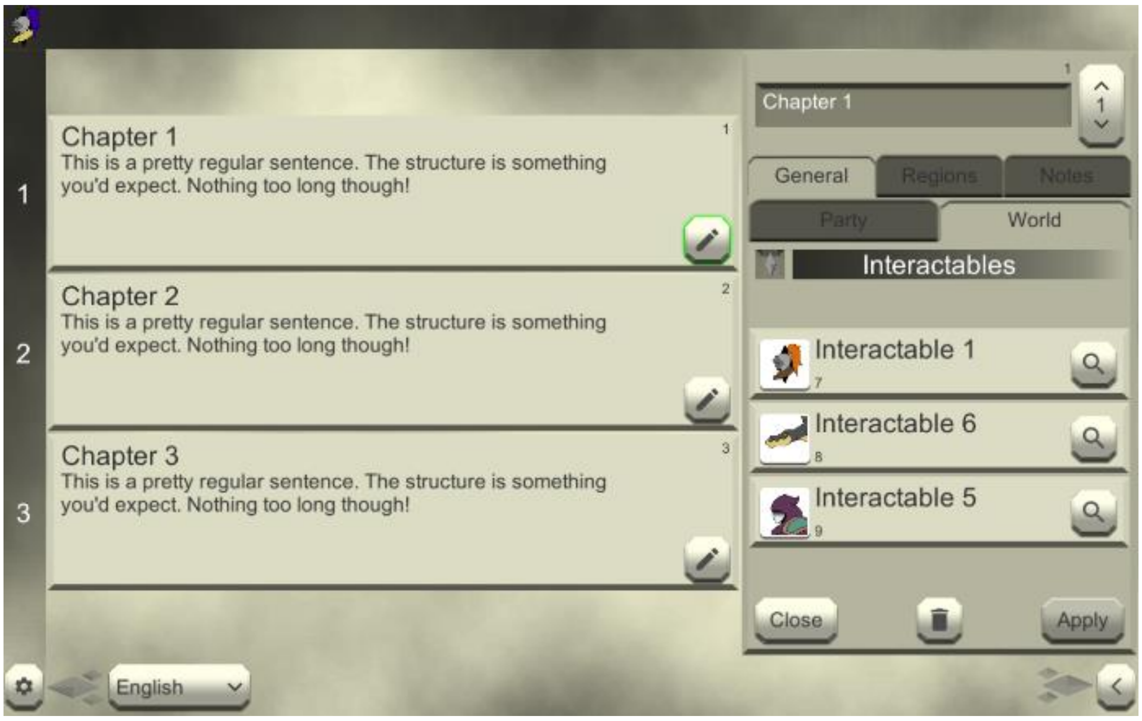
3.2.1 Chapter

A chapter is the first stage of the overall structure and is identified by its name. All the stages within the chapter are supported by including a set of reoccurring main characters, including both playable characters (party members) and NPC (scene interactables). Explorable regions are also defined per chapter and their purpose here will be further detailed in Section 3.2.2.

Party members are the characters that are directly controlled by the player. The game progresses by their actions and is observed through their lens. For each chapter it is possible to have multiple party members. One of them is controlled directly while the others follow. Scene interactables are main characters that can be interacted with throughout all stages belonging to the chapter.

Both party members and scene interactables are linked to an interactable from the assets. As an interactable serves as a whole character, more than one of each cannot exist at any given time and therefore these characters are defined at this stage.

Chapters are linear, meaning that one must be completed before the next can be started. When the last chapter is complete, the game itself is complete. In order to complete a chapter, the last phase stage belonging to the chapter must be completed.


Chapter 1
This is a pretty regular sentence. The structure is something you'd expect. Nothing too long though!

Chapter 2
This is a pretty regular sentence. The structure is something you'd expect. Nothing too long though!

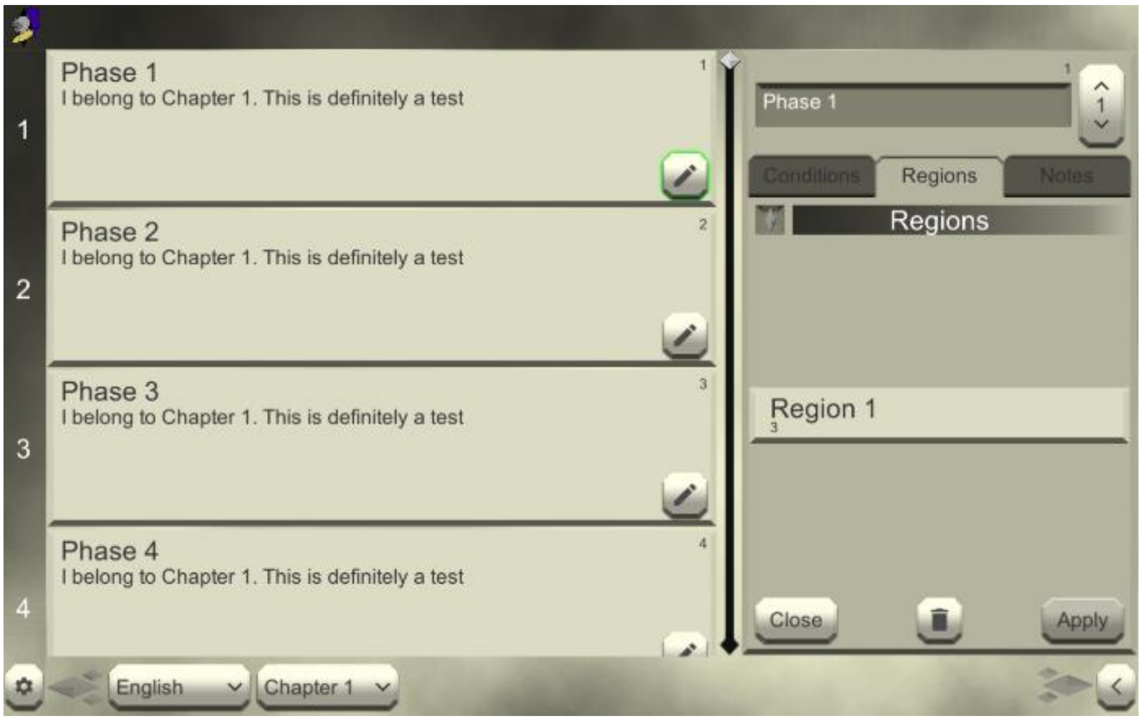
Chapter 3
This is a pretty regular sentence. The structure is something you'd expect. Nothing too long though!

FIGURE 8. A chapter being edited in the chapter sub-editor, displaying a list of main characters

3.2.2 Phase

A chapter is a collection of phases, which in itself represents a linear moment in time. The regions that were assigned in the chapter can be opened and modified for each phase as seen in Figure 9. Developers can have the region reflect the events of the game, opening up opportunities for environmental storytelling.

A phase is completed when one of its stage groups is completed. A stage group is, as the name suggests, a group of stages that can consist of multiple quests, objectives and/or interactions. Phases and stages represent a moment in time, giving the player an opportunity to complete the phase with any action. It also allows players to progress in the game as they see fit.



1
Phase 1
I belong to Chapter 1. This is definitely a test
2
Phase 2
I belong to Chapter 1. This is definitely a test
3
Phase 3
I belong to Chapter 1. This is definitely a test
4
Phase 4
I belong to Chapter 1. This is definitely a test
Phase 1
Conditions
Regions
Notes
Regions
Region 1
Close
Apply
English
Chapter 1

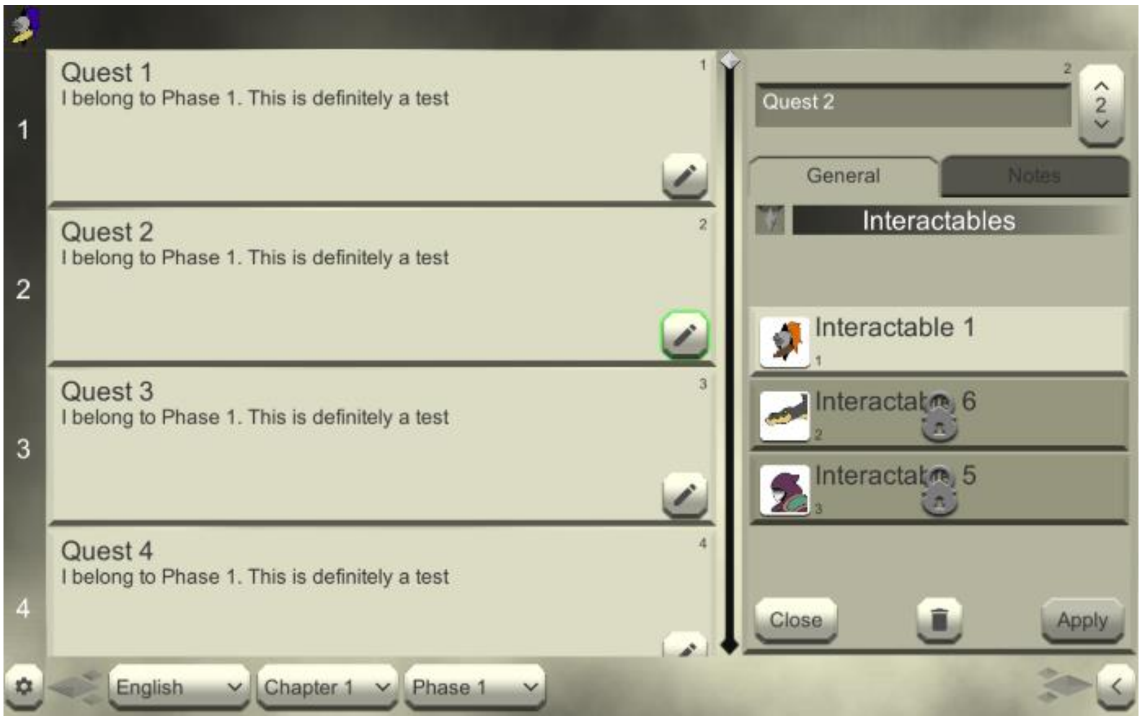
FIGURE 9. *A phase being edited in the phase sub-editor, displaying the region provided by the chapter seen in the action bar*

3.2.3 Quest

A phase is made up of quests. A quest is a self-contained adventure within the phase with a clear beginning and end. Unlike chapters and phases, quests are not linear, and they share the same moment in time. Multiple quests can be started and completed in any order.

The scene interactables that were chosen in the chapter are now allocated to a quest. A scene interactable cannot belong to multiple quests in the same phase, as the character they represent cannot be in multiple places at the same time. Developers must pick which quest each scene interactable belongs to during a phase. When an interactable has been picked for one quest, it can no longer be picked for another. An example of this can be seen in Figure 10, where all the unavailable interactables are marked with a lock icon.

The last objective of the quest must be completed in order to complete a quest.



1
Quest 1
I belong to Phase 1. This is definitely a test
2
Quest 2
I belong to Phase 1. This is definitely a test
3
Quest 3
I belong to Phase 1. This is definitely a test
4
Quest 4
I belong to Phase 1. This is definitely a test
Quest 2
General
Notes
Interactables
Interactable 1
Interactable 6
Interactable 5
Close
Apply
English
Chapter 1
Phase 1

FIGURE 10. A quest being edited in the quest sub-editor, displaying a list of both available and unavailable main characters

3.2.4 Objective

A quest is divided into objectives. An objective is a linear stage that serves to provide tangible steps, which are required to complete a quest. The progress of a quest can be measured by the objectives that are completed. Developers can write what is expected from the player during an objective as a journal entry.

Temporary scene interactables can also be assigned to objectives as can be seen in Figure 11. These scene interactables only appear in the scene while the objective is active.

The objective is completed when a specific interaction is performed by the player.

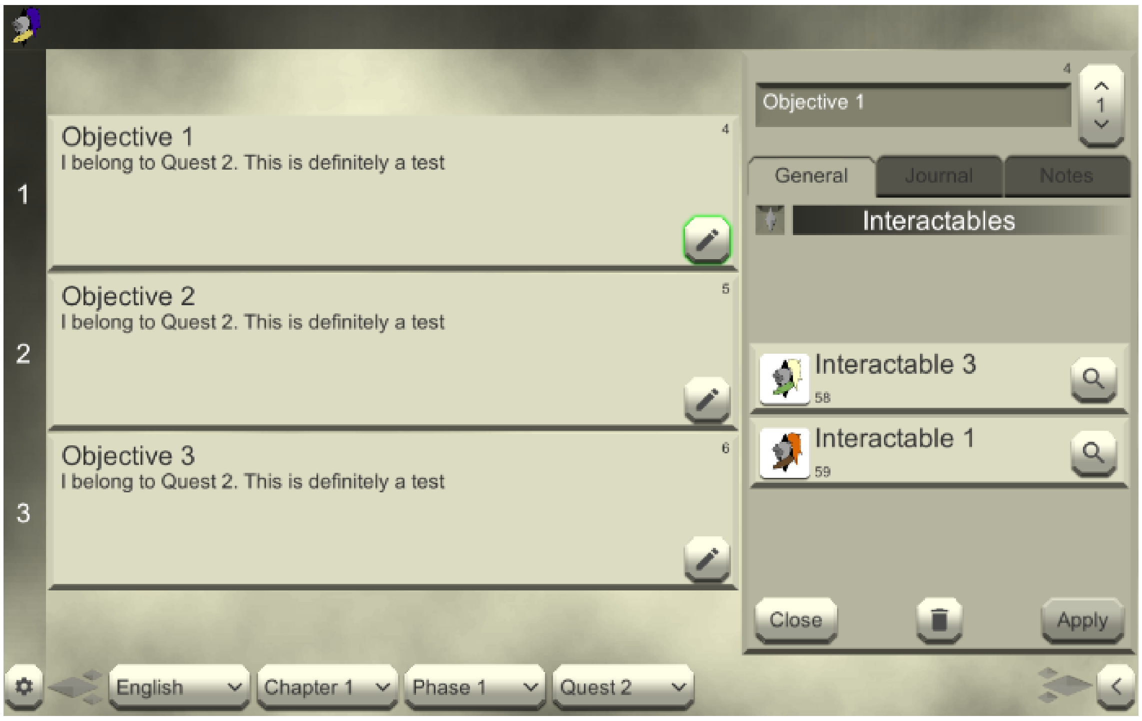


FIGURE 11. An objective being edited in the objective editor, displaying a list of temporary interactables

3.2.5 Interaction

Every scene interactable has at least one interaction. An interaction is everything that happens before, during and after the player interacts with the interactable and also specifies where the interaction may happen. Everything that happens in the game comes as a result of an interaction and so an interaction has by far the most settings of any feature in the editor.

An interaction uses a description to identify itself. The description can be read as a goal, the desired result of the interaction. The icon of the interactable to which the interaction belongs is also displayed in the editor to help the developer identify which interactable it concerns.

General

Interactions are linear but completing the last interaction does not automatically complete the objective it belongs to. Instead, one of the interactions must be marked as the one which completes the objective.

It is possible for interactions to be repeated if this is enabled by the developer. Before this can happen, the interaction must be reset and the method by which this is done can be specified. An interaction can be repeated infinitely or for a specific amount. A practical example of this would be to interact with the apple tree from Section 3.1.2. The reset method is duration and the secondary value is time in minutes for the tree to grow a new apple.

An interaction can also trigger special effects. A filter can be placed over the camera to fade the screen in or out, to or from black or white and the duration of the transition can be specified. The screen can be shaken during an earthquake by animating the camera, and sound effects can provide a further feedback of the event. All these effects can happen before, during or after the interaction.

Behaviour

Interactions are often triggered by walking up to something and pressing a button. Some interactions may come as a surprise and a button press would take some of that surprise away. Interactions can be triggered automatically or by the player input. In both cases, the minimal distance between the party member and the interactable must be specified. If the interaction is triggered by the player input, the party member must also be facing the interactable.

A delay can be added to the interaction when it has been triggered. This can help to simulate the effect of gathering items. As such, an animation can be played during the delay, such as reaching for something on the ground. How long the delay lasts can be specified, as well as whether the delay can be interrupted by the player or not.

As interactions will often happen between characters, it is possible for them to exchange words. A separate speech editor allows editors to add a few phrases to selected interactables to simulate a conversation. If the auto-play option was selected as seen in Figure 12, such a conversation would continue even without the player input.

How the interaction happens and what happens when it does are answered, but not where it happens. In this editor, interactables are linked to interactions, not the other way around. The interaction’s position must be specified per interaction and the interactable will be placed on top of it. The position, rotation and size can be changed in the scene through the interaction editor as seen in Figure 13.

Conditions

In order to successfully complete an interaction, certain conditions must be met. These conditions come in three categories: stages, items and equipment.

Stage conditions function like stage groups for phases. Only one needs to be completed to meet the condition. To fulfil the item condition, certain items must be in the party member’s inventory. What items and how many are specified by the developer, as well as the option to retain the said items when the conditions are met.

Certain items must be equipped in the party member’s gear slots to fulfil the equipment condition. This is intended for disguises or overcoming environmental hazards.

Events

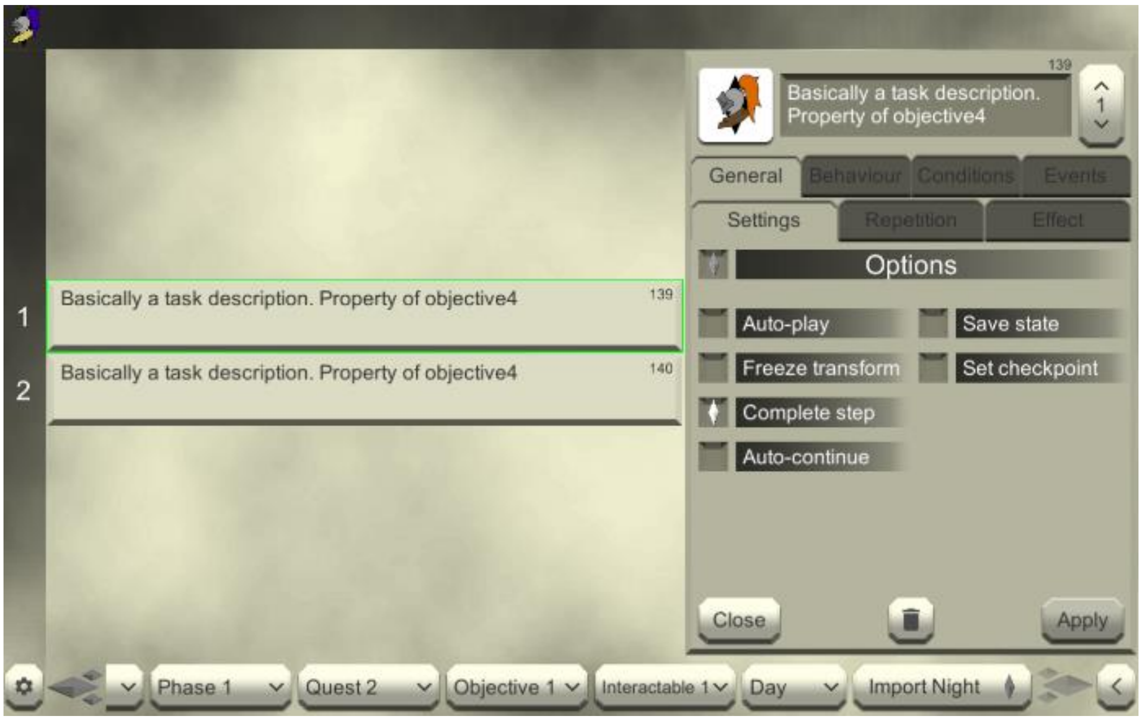
Interactions are concluded with an event. The first event is a scenario, which will temporarily take the player out of the usual game environment. A scenario can be one of three types: cutscene, encounter or shop.

A cutscene is a short cinematic experience for the player to sit back and watch, which can be used to give more context to a situation when the basic speech is not enough. Unlike the basic speech, developers can also position the camera for cinematic angles.

An encounter is a battle between the player’s party members and one or more designated enemies. An enemy is linked to an interactable, whose properties, such as health and power, are carried over to the enemy. Defeating enemies can award the player with items, as described earlier in Section 3.1.2.

The shop scenario allows the player to browse, purchase and sell items from and to a shop. Items have an inherent value, which is specified by the developer in the item editor, for which items are bought and sold. Different shops can offer different items. What items are available is decided by the developer.

When the scenario is concluded, the player is returned to the game world and can be rewarded for their deeds. Rewards can be items and in-game currency. As usual, items and item quantities can be specified by the developer. The in-game currency can then be used to purchase items from a shop.


Basically a task description. Property of objective4 139

Basically a task description. Property of objective4 140

Options

Auto-play
Save state
Freeze transform
Set checkpoint
Complete step
Auto-continue

FIGURE 12. An interaction being edited in the interaction sub-editor, displaying the general settings

3.2.6 Time

Games made with the editor will feature a day and night system. During night-time, the world goes dark (Figure 13), lighting up once more at the break of dawn. Inhabitants of the world take notice and act accordingly.

Every interaction has a separate day and night version. The interactables of the game can behave differently depending on the time of day. During the day, these interactables may be outside, doing their business. At night, they get some rest in their beds or hang out by the campfire. The shopkeeper mentioned in Section 3.2 can finally go home and get some rest.

Developers can create quests, where one objective is to meet a shady character at a specific location during night-time. That character will be elsewhere during daytime and could let the player know that now is not the right time, reminding the player of their objective and to urge them to try again later.

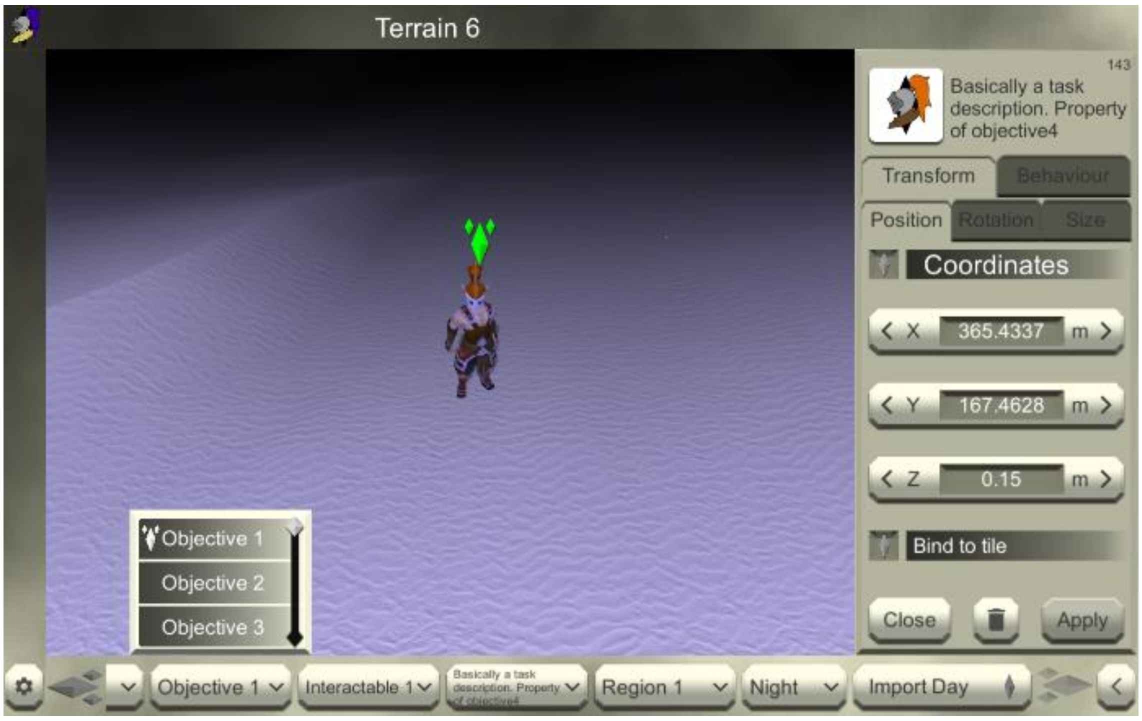


FIGURE 13. An interaction's position being edited in the scene sub-editor

4 LIMITATIONS

The editor addresses many issues concerning the development of large-scale adventure games with limited resources, but it does not come without its own limitations.

Resources

As briefly touched upon in Chapter 2, creating a game editor is a time-consuming process, and the cost of it will for many developers not outweigh the benefit. This project was conceived in 2016 and has been in production since 2018 and no game has yet been made. It might take another two years before the editor itself is even ready to do so.

Size Matters

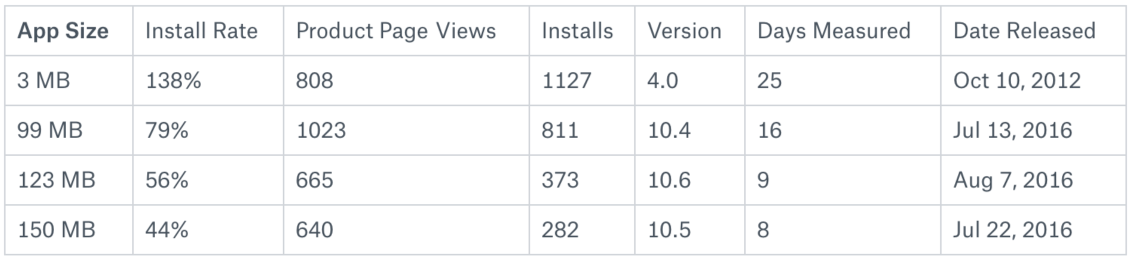
Users are limited to use assets provided by the editor and cannot upload any home-made assets. The two primary reasons for this are related to security and size.

The typical assets a user would want to upload are images in form of textures and 3D models. When a 3D model is uploaded, it would already require a material to render the texture and a skeleton to perform the animations. To detect what file is uploaded and construct it in such a way that it is usable by the editor would be very difficult. Developers could also potentially upload harmful files, which would spread to the players of their games.

If a developer decides to upload their own assets, there is also a risk that it is not well optimized and is detrimental to the game's performance. Players would also have to download all this extra content in order to play the game.

The size of a mobile app has a significant influence on its download potential. When an app reaches a size of 100MB, the download rate drops by 66 percent (Reinhardt 2016). This drop in downloads can be seen in Table 1, which shows the correlation between app sizes and install rates, as well as the product's page views. A cellular download limit prevents the downloading of files that size over 3G or 4G networks.

TABLE 1. App install rate by file size (Reinhardt 2016)



App Size	Install Rate	Product Page Views	Installs	Version	Days Measured	Date Released
3 MB	138%	808	1127	4.0	25	Oct 10, 2012
99 MB	79%	1023	811	10.4	16	Jul 13, 2016
123 MB	56%	665	373	10.6	9	Aug 7, 2016
150 MB	44%	640	282	10.5	8	Jul 22, 2016

On non-mobile platforms such, as desktop computers and game consoles, the download size does not matter quite as much, and it is not uncommon to see modern games at sizes of over 100GB.

Performance

The editor app is developed to be usable on mobile devices. The games it can produce could greatly outperform the device it is made for. It is up to developers to determine what type of game performance they deem acceptable to release on the market.

Creativity

The editor only allows users to make adventure games. There are no plans to implement features that allow users to create other types of games, such as shooter-style games.

Appearance

In character driven games, characters are often identified by their appearance. Some slots affect the appearance of the interactable, mainly the main and off-hand slots. Other slots do not provide visual feedback to the user. The reason for this is that these items must first be developed in 3D and it must be ensured that the item fits on all interactables and during all animations. The cost outweighs the benefits.

It is important to consider the value of a graphical upgrade as a reward for the player’s actions and to serve as a motivator to overcome a challenging obstacle. Items carried in the party member’s hands still provide that visual feedback.

Complexity

Developers are bound to run into logical errors if they do not possess a deep understanding of the logic provided by the editor, visible in Figure 1. The logic makes it impossible for technical bugs to occur, but it is not equipped to prevent logical errors. It is possible for developers to create requirement paradoxes, where, for example, a phase can only be completed by a quest that is locked behind a future phase. The developer may require a player to have a unique item to continue, which was possible to lose at an earlier stage. The more complex a game becomes, the more likely it is that these errors will occur, and so adequate external planning is still required.

5 CONCLUSION

Due to the resources required to create an adventure game, it is unfeasible for an individual or small team to successfully conclude a project that can match AAA giants in scope. The Expedition Game Editor addresses this problem by providing a streamlined framework to build adventure games, without writing a single line of code. This greatly reduces the time and resources required when making an adventure game.

The games made with the editor can vary in complexity, giving developers the freedom to create what they want. However, the editor is not the solution to all problems as it is met with certain limitations. Developers are limited to what the editor has to offer.

While the editor is not yet finished, it offers a glimpse into the future. The editor logic that is the foundation of the Expedition system has been proven to work and is demonstrable by navigating through stages within the editor.

REFERENCES

1. Chalk, A. 2015. The Witcher 3: Wild Hunt cost $81 million to make. Date of retrieval 12.10.2019 https://www.pcgamer.com/the-witcher-3-wild-hunt-cost-81-million-to-make/

2. Staats, J. 2019. The WoW Diary, A Journal of Computer Game Development. Las Vegas: whenitsready LLC.

3. Unity, 2020a. Game engines-how do they work? Date of retrieval 15.03.2020 https://unity3d.com/what-is-a-game-engine

4. Unity, 2020b. Real-time solutions, endless opportunities. Date of retrieval 15.03.2020 https://unity.com/solutions

5. TNW DEALS, 2016. This engine is dominating the gaming industry right now. Date of retrieval 12.10.2019 https://thenextweb.com/gaming/2016/03/24/engine-dominating-gaming-industry-right-now/

6. Sweeney, T. 2015. If You Love Something, Set It Free. Date of retrieval 15.03.2020 https://www.unrealengine.com/en-US/blog/ue4-is-free

7. Unity, 2020c. Plans and pricing. Date of retrieval 12.10.2019 https://store.unity.com/#plans-individual

8. Unity, 2020d. Multiplatform – Publishing your game to over 25 platforms. Date of retrieval 12.10.2019 https://unity3d.com/unity/features/multiplatform

9. Unity, 2020e. This is why creators choose Unity. Date of retrieval 15.03.2020 https://unity.com/our-company

10. Reinhardt, P. 2016. Effect of mobile app size on downloads. Date of retrieval 13.10.2019 https://segment.com/blog/mobile-app-size-effect-on-downloads/