公立はこだて未来大学 2019 年度 システム情報科学実習

グループ報告書

Future University Hakodate 2019 System Information Science Practice

Group Report

プロジェクト名

クリエイティブ A.I.

Project Name

Creative A.I.

グループ名

物語分析班

システム班

視聴覚班

音響班

Group Name

Story analysis Group

System Group

Visual Group

Audio Group

プロジェクト番号/Project No.

9

プロジェクトリーダ/Project Leader

1017009 白石智誠 Tomonari Shiraishi

グループリーダ/Group Leader

1017046 宇田朗子 Akiko Uda

1017003 太田和宏 Kazuhiro Ohta

1017059 友広純々野 Suzuno Tomohiro

1017122 山内拓真 Takuma Yamauchi

グループメンバ/Group Member

1017044 石川一稀 Kazuki Ishikawa

1017189 稲垣武 Takeru Inagaki

1017046 宇田朗子 Akiko Uda

1017003 太田和宏 Kazuhiro Ohta

1017048 小川卓也 Takuya Ogawa

1017138 斉藤勇璃 Yuuri Saito

1017154 宍戸建元 Takeru Shishido

1017009 白石智誠 Tomonari Shiraishi

1017059 友広純々野 SuzunoTomohiro

1017097 中村祥吾 Nakamura Shogo

1017156 長野恭介 Kyosuke Nagano

1017116 西川和真 Kazuma Nishikawa

1017015 根本さくら Sakura Nemoto

1017122 山内拓真 Takuma Yamauchi

1017186 蓬畑旺周 Akinori Yomogihata

指導教員

村井源 平田圭二 迎山和司 田柳恵美子 角薫 松原仁

Advisor

Hajime Murai　Kazushi Mukaiyama　Emiko Tayanagi　Keiji Hirata　Kaoru Sumi Hitoshi Matsubara

提出日

2020 年 1 月 22 日

Date of Submission

January. 22, 2020

概要

物語自動生成研究の一例として，「気まぐれ人工知能プロジェクト『作家ですのよ』」[1]が挙げられる. 2012 年にスタートしたこのプロジェクトは，星新一らしい小説を自動生成することを目標としており, 2016 年には星新一賞の一次審査を通過したことで話題を集めた．このように人工知能分野では，より人間の感性に基づく創造物を自動生成する取り組みが試みられている．しかし，文法や意味において「正しい」文章を生成するということや，繋がりのある文章を生成するという部分においてまだまだ課題が残されている．

また，ゲーム AI に関して三宅(2012)は，ゲーム AI に関する 2 つの国際的な産業カンファレンス(Game Developers Conference AI Summit，Game AI Conference)と 2 つの国際学会（IEEE Computational Intelligence and Games, AIIDE）が設立され，国際的にデジタルゲームにおける人工知能を研究する気運は高まりつつあるにもかかわらず，全体的な進歩のスピードに対して情報環境が整備されておらず，依然としてデジタルゲームの AI に関する情報は集めにくく研究者もエンジニアも参入しにくい状況にあると述べている．

本プロジェクトの目標は，物語自動生成の技術，およびデジタルゲームにおける人工知能技術などを考案したうえで，ロールプレイングゲーム(RPG)内の要素を自動生成する人工知能システムを開発し，評価を行うことである．具体的には，ゲームシナリオの自動生成，ダンジョンマップの自動生成, BGM の自動選択などが挙げられる．また，自動生成要素に加え，ゲームをより魅力的に魅せるための視覚的要素の制作も目標としている．プロジェクトのメンバーは，物語分析班，システム班，視覚班，音響班の 4 班に分かれて作業を分担し，開発を行った．

キーワード 人工知能，プログラミング, RPG, 物語分析，ゲーム AI

（＊文責：根本さくら）

Abstract

One example of research on automatic story generation is an attempt to automatically create short stories, such as Shinichi Hoshi [1]. The project, which started in 2012, aims to automatically generate novels that seem to be Shinichi Hoshi, and in 2016 attracted attention after passing the first screening of the Shinichi Hoshi Award. Therefore, in the field of artificial intelligence, attempts are being made to automatically create creatures based on human sensitivity. However, there are still issues to be solved in order to generate "correct" grammar and semantics and generate connected sentences.

Regarding game AI, Miyake (2012) acknowledged that despite the growing international interest in artificial intelligence research, two international industrial conferences on game AI (Game Developers Conference AI Summit and Game AI Conference) and two With the establishment of an international academic society (IEEE Computational Intelligence and Games, AIIDE), the information environment of the overall speed of progress has not been improved in digital games,

and it is still difficult to collect information on digital game AI. Both engineers and engineers said it was difficult to enter the market.

The goal of this project is to develop and evaluate an artificial intelligence system that automatically generates the elements of a role-playing game (RPG) after devising techniques for automatic story generation and artificial intelligence technology in digital games. Examples include automatic generation of game scenarios, automatic generation of dungeon maps, and automatic selection of background music. Project members are divided into four groups: story analysis group, system group, visual group, and audio group. Work has been split and development has taken place.

Keywords　artificial intelligence, programming, RPG, story analysis, gameAI

(*Responsibility for wording: Sakura Nemoto)

目次

4.3 音響班

4.3.1 分析

4.3.1.1 分析の再検討

4.3.1.2 分析内容の再定義

4.3.1.3 後期における場面の分析

4.3.1.4 音響特徴量の抽出

4.3.2　ニューラルネットワークの学習

4.3.2.1 データの調整

4.3.2.2 データセット作成プログラム

4.3.2.3 ニューラルネットワークの実装

4.3.3　選曲システム

4.3.3.1 選曲システム

4.3.4 展望

4.4 視覚班

4.1.1 ロゴ

4.4.2 モデル（モーション含む）

4.4.2.1 モデリング

4.4.2.1.1 素体の作成

4.4.2.1.2 キャラクターのラフモデリング

4.4.2.2 キャラクターモデル

4.4.2.2.1 主人公モデル

4.4.2.2.2 ヒロインモデル

4.4.2.2.3 用心棒モデル

4.4.2.2.4 シスターモデル

4.4.2.2.5 モブモデル

4.4.2.3 建築モデル

4.4.2.4 エネミーモデル

4.4.2.4.1 ナメクジ

4.4.2.5.2 カエル

4.4.3 ステージの制作

4.4.3.1 ステージの詳細について（後期）

4.4.3.1.1 アルトリーヨ

4.4.3.1.2 トフー

4.4.3.1.3 ダンジョン

第1章 はじめに

1.1 背景

三宅(2015)は，デジタルゲームにおける人工知能は，1980年～90年代に様々な応用が試みられたが，90年代後半からその大規模化・構造化に伴い次第に役割が明晰化され，大きく三つの人工知能「キャラクタAI」，「メタAI」，「ナビゲーションAI」に分化することとなったと述べている．人工知能を用いたゲーム開発の成功例としては，1980年に発売された「ローグ(Rogue)」が挙げられる．ローグとは，ダンジョン探索型のRPGである．マップが毎回自動生成されるため，飽きがこないというのが特徴である.このローグに似た特徴を持つゲームは，ローグライクゲーム(Rogue-like Games)と呼ばれ，多くのソフトウェアが開発されていくこととなった．また，ゲームシナリオの自動生成において川野ら(2018)は，シナリオライターの負担軽減と物語多様性の担保という観点から，ゲームシナリオ自動生成システムの開発が必要であると述べている．このゲームシナリオ自動生成システムにおいては，TRPG方式に基づく物語自動生成ゲームの開発[5]など挑戦例は挙げられるが，未だ成功例はない．

本学で一昨年に取り組まれたプロジェクトであるインタラクティブ・ストーリーテリングでは，コンピューターによるインタラクティブなシステムの開発に試みていた[6]．また，昨年度，本プロジェクトであるクリエイティブAIは，言語表現と視覚表現を用い，バトル作品とホラー作品の物語を自動生成するシステムの開発を試みていた[7]．

本プロジェクトの今年度の目標は，RPGのイベントを適切な順番で繋げ合わせるシステムを考案することで，ストーリーが破綻していないゲームシナリオを生成する人工知能システムを開発することである．また，ダンジョンの自動生成やBGMの自動選択等，デジタルゲームにおける人工知能技術の考案および開発をすることを目標にしている．それに加え，ゲームをより魅力的に魅せるための視覚的要素の制作も目標としている．最終的には完成したゲームの評価実験を行う予定である．

（＊文責：根本さくら）

1.2 目的

本プロジェクトが目指すのは，オリジナルの作品を創作できる人工知能システムを実現すると同時に，「面白さ」や「美しさ」の科学的な理解を前進させることである．この目的を実現するために，今年度はストーリーが破綻していないゲームシナリを生成し実行できるゲームシステムの開発と，そのための適切な視覚化表現，音響効果の構築を対象として取り上げた．ゲームはドラマなどのように単純にストーリーが進行するのではなく，プレイヤーの意思で，プレイヤーが選択した

選択肢によってストーリーが分岐する．また，三宅(2015)は，ゲーム技術は，楽しい・怖い・感動する・興奮するなどの言葉を超えたユーザ体験を生み出すものであると述べている．加えてゲームは，美しい映像，盛り上がる音楽など，さまざまなエンターテインメント要素を含んだコンテンツである．そのため，人の心を動かす「面白い」オリジナル作品を創作できる人工知能システムを実現するうえで，ストーリー，映像，音楽のすべてに「面白い」要素が含まれている，また，さまざまなユーザ体験を生み出すことのできるゲームを取り上げるのが最適であると考えた．最終的には，人が遊べるレベルに「面白い」ゲームを完成させる予定である．

（＊文責：根本さくら）

1.3 課題

目的を達成するために解決すべき課題は以下の通りである．

● 分析対象とするゲームのジャンルと作品の決定
● 分析方法の決定
●RPG のストーリー構造やイベントの分析
● 分析データからイベントを自動生成するプログラムの制作
●イベント再生システムの制作
●自動生成されたイベントをイベント再生システムで読み取れるようテキスト化
●ゲーム操作システムの制作
●ゲーム内 UI の制作
● 戦闘システムの制作
●コンセプトアートの制作
●キャラクターデザインの制作
●ステージ・キャラクターのモデリング
●使用曲の決定
●曲の自動決定システムの制作
●プログラムの統合

本プロジェクトでは，分析対象とするゲームのジャンルと作品の決定，分析方法の決，RPG のストーリー構造やイベントの分析，分析データからイベントを自動生成するプログラムの制作，イベント再生システムの制作，自動生成されたイベントをイベント再生システムで読み取れるようテキスト化，ゲーム操作システムの制作，ゲーム内 UI の制作，アイテムの使用設定，戦闘システムの制作，コンセプトアートの制作，キャラクターデザインの制作，ステージ・キャラクターのモデリング，使用曲の決定，曲の自動選択システムの制作，プログラムの統合を課題とした．また，今後これらを評価する方法としてゲームの完成度に関するアンケート調査を実施し，そのアンケートの結果を分析する予定である．

（＊文責：根本さくら）

第 2 章 プロジェクト学習の概要

2.1 課題の設定

本プロジェクトは，人工知能による創造的な物語自動生成システムの開発を通して物語構造への理解を前進させることを目標としている．前年度の同プロジェクトは物語プロットを自動生成し，そのプロットに基づく視覚表現を 3DCG で行うことで統合的な物語の生成システムを開発していた．
今年度はゲームのジャンルとしてロールプレイングゲームを選択した．その理由は，ロールプレイングゲームは日本で長く続くジャンルであり，シリーズとして多くの作品が作られていることから，物語を分析するのにあたって適切であると考えたからである．
本プロジェクトでは課題解決のために，既存の物語構造を分析したデータを収集物語を生成する物語分析班，生成されたプロットに合わせた音響選定システムを制作する音響班，プロットに合わせた視覚表現を行う，それらを統合した可変なロールプレイングゲームを実行可能なシステムを作成するシステム班の４つの班に分かれて活動した．

（＊文責：白石智誠）

2.2 到達目標

本プロジェクトでは，物語プロットを自動生成し，そのプロットに基づいた視覚表現や音響などを統合したゲームシステムを開発することを到達目標に設定した．ここで，物語自動生成システムの完成版とは，分析したデータから，視覚表現と音響表現を含むロールプレイングゲームとしてプレイ可能な矛盾のない物語プロットを自動生成できるシステムを指す．また，物語自動生成システムによって生成された物語が面白いかどうか，矛盾がないかどうかなどをアンケートで回答してもらい，その結果を分析する予定である．

（＊文責：白石智誠）

2.3 課題の割り当て

2.3.1 物語分析班

物語分析班は PPG のシナリオを自動生成することを課題としている．シナリオを自動生成するため，シナリオの構造分析・シナリオを自動生成するためのアルゴリズムの作成・
シナリオを自動生成するために必要なデータベースの作成を一年間を通して行う．今年度の活動では「面白い」RPG シナリオの自動生成を行えるような人工知能システムを作成することを目標としてい

る．このような人工知能システムを作成するにはどのような作品のシナリオの構造分析を行うべきかを考え，選んだ作品を分析することで「面白い」RPG シナリオを生成できると仮定した．この仮定をもとに物語構造の分析を行っていく．

（＊文責：宇田朗子）

2.3.2 システム班

システム班は，物語分析班の作成した物語データと，音響班が作成した音楽データを読み込み，視覚班の作成した 3D モデルなどの素材を用いて，Unity 上で RPG の一連の流れ（イベントシーンの生成，フラグの管理，マップ上の座標変更，戦闘の実行，音楽，効果音の変更など）を構築できるシステムの開発を目標としている．この目標を達成するために，イベント生成の分野では，物語データと音楽データを読み込み，それをタグ（セリフや音楽の情報）ごとに分け，イベントが発生したときにその情報を出力できるシステムを作成している．ゲーム的な要素では，自由に探索ができるマップ画面や，タグから敵を生成できる戦闘システムを作成している．これらの RPG の基礎的なシステムを開発し，ダンジョンの自動生成や，難易度の調整を行い，自動生成 RPG の面白さを追求していく．

（＊文責：西川和真）

2.3.3 音響班

音響班は，ゲーム中の各シーンに対し自動的に適切な音楽を設定することを人工知能を用いて実現することを目標とする．適切なシーンに適切な音楽を設定することができればシーンを引き立たせることが可能である．今回はゲーム中の音楽がシーン中のどの要素に影響されているのかを解明することができれば，適切な音楽の設定が可能であると考えた．それを音楽に高い評価を受けている市販のゲームを分析し，その結果を機械学習に生かすことによって実現する．

（＊文責：山内拓真）

2.3.4 視覚班

視覚班は RPG に必要なビジュアル部分の制作を受け持つグループである．1 年間を通しての目的は，プレイヤーに対して違和感のないグラフィックや見せ方を実現させることである．例えば，グラフィックのクオリティの低さや粗さといった視覚的な部分があると，それら気がとられてしまい，プレイヤーがシナリオに集中できなくなるとういう問題が起きる．シナリオに集中できずに，プレイヤーの感情が阻害されてしまうと，本プロジェクトの「面白い」ゲーム制作という目的とは反する問題になりうる．またこのほかにも，面白さを支えるようなグラフィックや見せ方を実現させる目的もある．高いグラフィックの実現や，魅力的なキャラクターを制作することで，プレイヤーの興味を高め，面白さを支えることができると考える．これらの点から制作するゲームにたいして自由度の高い表現をするという課題をを立てることで目的を達成させる．ここにおける自由度の高さとは，生成されるシナリオに柔軟

に対応できるような視覚的部分を指す．この表現を実現するために，３DCG でゲーム内の視覚部分を制作することに決定した．３D モデルは一度作ることで，キャラクターに様々なポーズを付与することや，そのポーズを付与する際にモーションキャプチャーを用いることが可能である点があげられる．また，一つのモデルを別のモデルに派生することが可能である点もあげられる．これらの点から生成されるシナリオに対して，一定の量のモデルから複数の表現を実現することでき，制作するゲームの幅を広くすることができると考える．

また，中間発表までの目的として各メンバー間の作業を円滑にする目的がある．この目的のために，メンバーの認識を統一する課題を立てた．この課題に向けて，プロトタイプの制作に伴い，最終的に開発するゲームイメージの可視化をおこなった．中間発表までの時点では，ゲームイメージ可視化のために，イベント上で使用するイラストや，キャラクターやステージのデザイン，デザインを基にモデルの制作をおこなった．ゲーム以外でも使用できるゲームロゴの制作をおこなった．中間発表までの開発の項目では，開発までの流れ，成果物，成果物の詳細な制作方法について記述していく．

（＊文責：友広純々野）

第 3 章 中間発表までの開発

3.1 物語分析班

3.1.1 作品分析

3.1.1.1 分析作品の検討

物語を自動生成するアルゴリズムを作成するにあたって，必要なデータを用意する必要がある．本項では，物語分析を行い，分析データを収集する作業に取り組むまでに行った作業工程について記述する．

まず，本プロジェクトでは，ゲームの中で「面白い」の主軸を担っているものはシナリオであり，「面白い」ゲームのシナリオを分析することで「面白い」シナリオの構造がわかると仮説を立てた．その仮説を検証するため，世間一般に「面白い」と評価されている RPG の構造分析結果から自動生成したゲームシナリオを使用したゲームの開発が目標となっていた．その為，最初にどのようなゲームを制作するかを決定する必要があった．選定方法にはブレインストーミングを用い，各自希望するゲームジャンルやゲームの要素を自由に提案してもらった．ゲームジャンルでは，『ロールプレイングゲーム（RPG）』や『恋愛シュミレーション』，『アクションゲーム』などが挙げられた．また，ゲームの要素としては，『王道』，『異世界』，『タイムトラベル』，『ロボット』，『推理』，『脱出』，『仮面ライダー』，『プリキュア』，『日常系』，『クトゥルフ神話』などが挙げられた．意見が出切ったタイミングで似たようなアイディア同士を結合させたり，多数決を取ったり，討論を交わしたりなどして，アイディアを絞り込んだ．結果，候補に残ったのは，【ドラゴンクエスト】のような『王道 RPG』と【ときめきメモリアル】のような『恋愛シュミレーション』の２つだった．

候補を 2 つに絞った後，どちらにするかは個人個人の好みではなく，客観的な視点で判断できるように話し合った．方針としては 2 つの観点から対象を評価することにした．1 つは，「分析対象として，必要な作品数が存在するのか」．分析する作品数としては，ひと作品あたり 2 桁以上あることが望ましいとされていた．『王道 RPG』では，【ドラゴンクエスト】シリーズや【FINAL FANTASY】シリーズ，【ファイアーエムブレム】シリーズや【テイルズ オブ】シリーズなどが分析するに足る作品と挙げられた．作品数は【ドラゴンクエスト】シリーズが【ドラゴンクエスト】から【ドラゴンクエスト XI 過ぎ去りし時を求めて】までの 11 本，【FINAL FANTASY】シリーズが【FINAL FANTASY】から【FINAL FANTASY XV】までの 15 本，と充分な作品数が発売されていると判断された．また，『恋愛シュミレーション』では，【ときめきメモリアル】シリーズが分析するに足る作品と挙げられ，作品数は【ときめきメモリアル】から【ときめきメモリアル 4】までの 4 本に加えて外伝や派生作品などがあり，こちらも充分な作品数が発売されていると判断された．もう 1 つは，「自動生成する物語を活かせるのはどちらか」．ゲームの自動生成という分野において，ダンジョンやお使いのような簡単なサブクエスト，敵キャラクターの行動パターンの自動生成は既に実現されている．今回のプロジェクトでは，新規性のある分野での自動生成を目指すこととなっていたので，自動生成する候補はキャラクターやスキル，技や武器，選択肢やストーリーの分岐の自動生成，曲やエフェクト，難易度調整や各種パラメーター調整などが挙げられた．その中でも選択肢やストーリーの分岐の自動生成を今回は行うこととなった．ユーザーが選んだ選択肢によってストーリーが変化するゲームは，『王道 RPG』『恋愛シュミレーション』どちらのゲームジャンルでもそのような作品は存在していた．最終的に，『恋愛シュミレーション』を希望していたプロジェクトメンバーから出た「恋愛物を作りたいのであって，恋愛シュミレーションゲームを作りたいわけではない」という意見と，「『王道 RPG』というゲームジャンルは，アクションや推理，恋愛など他の様々なジャンルを含んでいる」という意見から，恋愛要素のある『王道 RPG』を制作することに決定した．

次に，決定したゲームジャンルである『王道 RPG』の中から分析対象となる作品を選定した．分析対象となる条件は仮説から分析する価値のある，世間一般に「面白い」と評価されている RPG 作品であることとした．「面白い」RPG 作品というのは，個人の趣向によって千差万別であり，定義も曖昧で主観的なものである．そのため物語分析班では今回，「売り上げが多い」，「シリーズが続いている」「多くの人が知っている」の 3 つに当てはまるものを「面白い」RPG 作品であると定義して，選定を行った．この 3 つの定義は物語分析班で協議して決定した．「売り上げが多い」に決定したのは，ゲームを購入するのはそのゲームが面白そうと思ったからであり，売り上げの数がそう思った人の数に比例すると考えたからである．「シリーズが続いている」に決定したのは上記の通り，分析対象として必要な作品数が存在しているということもあるが，それだけ「面白い」と感じて次回作を希望する人が多かったということであると考えたからである．「多くの人が知っている」に決定したのは「多くの人が知っている」ということは人気が高いということと同義であり，人気が高いゲーム作品にはそれを裏付けるだけの面白さがあると考えたからである．また，ゲームジャンルを決定した際の意見から，恋愛要素が含まれているということや，実際にそのゲーム作品が入手できるということも条件とした．以上

の条件から，最終的に物語分析班は，【FINAL FANTASY Ⅳ】，【FINAL FANTASY Ⅵ】，【FINAL FANTASY Ⅶ】，【FINAL FANTASY Ⅷ】，【FINAL FANTASY Ⅸ】，【ドラゴンクエスト】，【ドラゴンクエストⅤ 天空の花嫁】，【テイルズ オブ エクシリア】，【シャイニング・ハーツ】，【サモンナイト】，【ルーンファクトリー4】の 9 作品を分析対象となるゲーム作品として決定した．

（＊文責：斉藤勇璃）

3.1.1.2 分析方法

物語を自動生成するアルゴリズムを作成するため，必要なデータはどういうものか．物語班では仮説にもある通り，世間一般に「面白い」と評価されている RPG の構造分析を行った結果のデータであると判断した．その為に物語の構造が明示的に分かる分析方法を取ることにした．

まず，RPG 作品に含まれる要素の調査を行った．RPG シナリオが主にどのような要素で成り立っているかを調べた．次に挙げられた要素を大まかにまとめ，カテゴリーとした．広義的に意味が似ているものや内容が被っているものを一つにまとめ，新たなカテゴリーの名前をつけたりした．最後にそのカテゴリーがどのような内容で成り立っているか，またどのような手法で行われているかを考え，表にした．

この表を基に他の RPG 作品も同じようなカテゴリーで構成されているのか，分析を行った．物語全体をシーンごとに分割し，一つ一つのシナリオ内で起こったことを決定したカテゴリーに当てはめていった．カテゴリーに当てはまったシーンにはカテゴリータグを割り振った．物語分析班ではこの方法を「カテゴリー分け」とした．カテゴリー分けを行うことで，シーンに割り振られたカテゴリータグがどのように遷移しているかが分かりやすくなる．このデータを多数収集することで，物語構造の基本パターンを分析することができる．

（＊文責：斉藤勇璃）

3.1.1.3 分析作品の傾向

前期時点で実際に分析を行ったのは「ドラゴンクエストⅤ～天空の花嫁～」である．「ドラゴンクエストⅤ～天空の花嫁～」は結婚相手を選ぶシステムがあり，一般的な RPG よりも恋愛要素が多く含まれる．また，人気作ということでシナリオも長く，RPG の構造を調べるために必要な要素をしっかりと含んでいた．「ドラゴンクエストⅤ～天空の花嫁～」は少年期・青年期前期・青年期後期の 3 部に分かれており，少年期と青年期ではシナリオの流れに多少の違いある．少年期は所謂チュートリアル的な存在であり目標が簡単かつ明快なシナリオの構造になっていた．

少年期は大きく分けて「お化け退治」「妖精の手助け」「王子様の救出」の 3 つのイベントに分かれていた．これらのイベントは，イベントが始まりそのイベントが終わるまで次のイベントが発生していなかった．それに加え，どれも共通でとても簡単な道筋が示されていた．「お化け退治」と「妖精の手助け」は，まず目標が提示され，次に主人公が自ら情報を探しに行き，取得した情報をもとに目標を達

成するための戦闘を行い，目標を達成していくという流れである．これら二つのイベントは似たような流れが出てきており，物語の中での再帰性が取れる部分である．次に「王子様の救出」イベントである．このイベントはチュートリアル部分から本筋の物語に進むための物語の連続性にかかわるものであった．このイベントでは目標を達成することができず，青年期に何をするべきかなどの新しい目標を得るイベントであった．この一連の流れでこの作品における物語・ゲームの進め方を実際にゲームをプレイするユーザーに伝えることができ，また私たちが分析している物語構造の特に単純なものが見えるようになっていた．

次に青年期である．青年期はイベントの構造が入り組んでおり，一つ目の目標が達成していないにもかかわらず，次々と目標が追加されていくという構造をしていた．しかし実際に分析した結果を見てみると，入り組んだ形の中でもある程度一定の規則があるということがわかった．

青年期前期の分析について記述する．青年期前期も大まかな流れは少年期と変わりはない．しかし，少年期と比べ青年期は目標が明示されていない，目標を達成するために別の目標を達成しなければならないなど物語の構造が複雑になっていた．

（＊文責：宇田朗子）

3.1.2 データフォーマット

3.1.2.1 データフォーマットの必要性

今回，中間発表のために用意されたシナリオは，キャラクターの動きや感情，セリフなどをすでに明示的，もしくは暗示的に示している．しかし，シナリオテキストをそのまま機械に通しても，機械は処理ができない．また，ゆくゆくはシナリオの書き起こしを自動で行うため，書き起こしの規則を決めなければならない．そのため，機械でもどのような要素がこの場面に存在しているかわかるように，また誰が書いても同じような形式で書き起こせるような規則を決定する必要があった．

まず，中間発表用に用意したシナリオを機械でも読み取れるようにシナリオデータを整形する作業の必要が出てきた．この作業を，「データフォーマット」とした．渡されたシナリオにおいて，1:場面に存在する要素をカテゴリーごとに分類，2:カテゴリーごとにタグ付け，を行なった．例えば，キャラのモーション（行動）にはａ話す，ａ考える，のように記した．これは，「a」が「キャラの行動を表すタグ」であり，a の後に続く単語が「行動の具体的な名称」である．このようなタグを前述の場面ごとの要素群に割り振っていった．タグの割り振り方，つまりはデータフォーマットの規則を示した図.3.1.2-1を下に示す．図中の番号は，あとに記述する「記述内容の各部分の説明」で扱う．

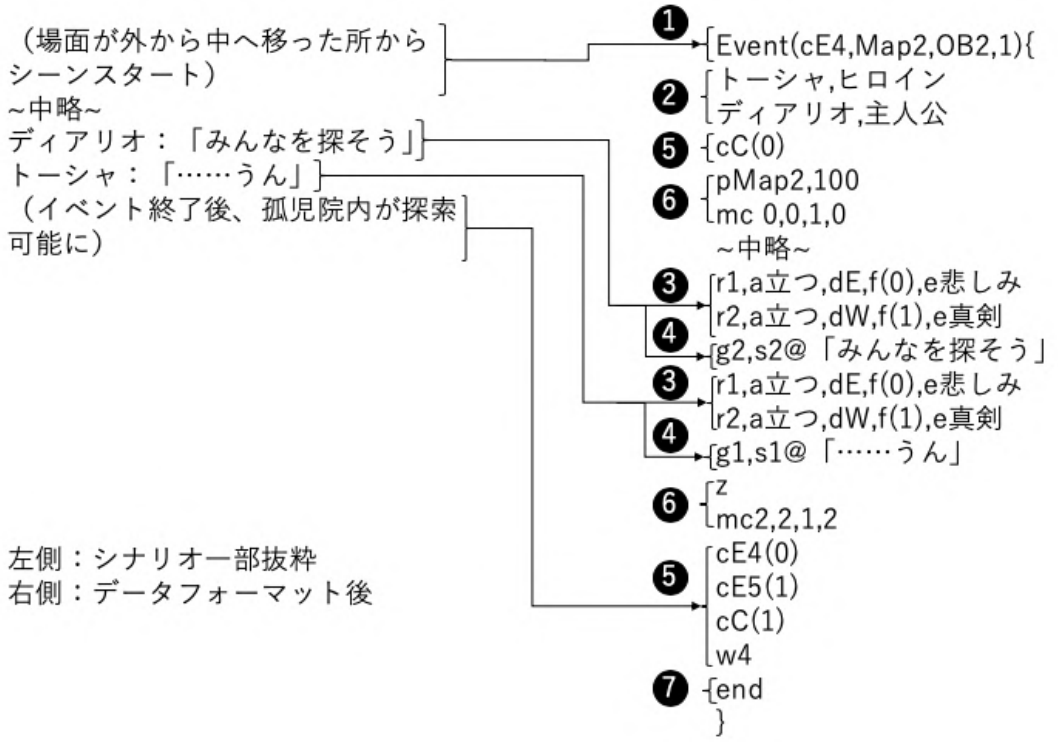


図 3.1.2-1 シナリオデータフォーマット

データフォーマットの規則については，大方は前年度の規則を踏襲しつつ，前年度にはなかった要素（様々なフラグの管理，音楽班に渡す情報など）を新たに含めるように手を加えていった．また，カテゴリーの中でも，更に種類を分けてタグ付けを行った．これによって，よりわかりやすく詳細な管理，書き起こしを可能にした．例えば，フラグ管理を行うために新たなタグ「c」を作った．しかし，フラグとは様々な種類が存在する．「戦闘」フラグや，「イベント発生」フラグなどである．そのような場合，「c」の後にさらに「B」や「E」などを付与できるようにすることで，種類ごとの管理を可能にした．これにより，「cB」というタグを見れば，「フラグ管理用のタグである．特に戦闘フラグを管理しているものだ」と分かるようになった．

（＊文責：中村祥吾）

3.1.2.2 対象を絞った経緯

ただし，場面内の全ての要素を書き起こすのは現実的ではない．よって RPG のある 1 場面の中でも，後々自動生成を行うにあたって必要となると予測された要素を優先することとした．今回のプロジェクトにおいて優先された要素は，物語の進行・面白さに必要な要素や自動生成に影響を与える要素である．優先された要素に関しては，各班のメンバーと随時話し合い，どの要素が各班に必要か，中間発表までに用意するべきもの，最終的に用意するべきものの見通しを立てた．

物語の進行に必要な要素として，どこにいるか（背景），どのキャラクターを表示するか，現在誰が喋っているか，フラグの管理など．物語の面白さに必要な要素として，キャラクターがどんな表情か，

どんな行動をしているか（立ち絵の差分表示），BGM や SE の管理など．自動生成に必要な要素として，BGM の自動生成に必要な情報，主体が誰か，シーンの切れ目など．現在時点で必要と思われる要素のタグを表.3.1.2-1 に示す．

表.3.1.2-1

タグ	タグの意味
pMap	場所の宣言（背景）
pT	一枚絵の宣言（背景）
c	フラグ
r	キャラの指定
a	モーション
d	キャラの向き
f	キャラの位置
e	キャラの表情
g	カット情報
s?@	セリフ．？には数字が入る．
w	座標宣言
z	シーンの終わり
end	イベントの終わり
mc	音楽班に渡す要素の宣言
WP	イベント終了時にマップ移動する際に使用

（＊文責：中村祥吾）

3.1.2.3 記述内容の各部分の解説

また，記述した内容は，1:イベント起動部分 2:宣言部分,3:キャラクター表示部分,4:主体決定・テキスト表示部分,5:管理部分,6:シーンの開始部分,7:イベント終了部分に分かれている．以降，図.3.1.2-1 に対応させて説明する．

1 はそのイベントが起こるための条件・イベントの形式などを書いている部分である．書いてある場所としては最初である．ここに書いてある条件に合致しなければ(false であった場合），イベントは発生しないようになっている．今回の場合，［Event(cE4,Map2,OB2,1)］とあった場合，イベント 4 のフラグが true であり，Map2 におり，OB2 というオブジェクトに触ったら，形式 1（立ち絵形式）でイベントを始める，ということである．

2 は現在のイベント内でどのキャラクターが出るかを書いている部分である．書いてある場所としては 1 の直下のみである．ここでモデルや立ち絵とキャラクターの名前を紐づけて，宣言された順番にナンバリングしていく．

今回のように書いてあれば，ヒロインという名前の立ち絵を，トーシャという名前と紐付けし，1 番目のキャラクターとする．主人公という名前の立ち絵を，ディアリオという名前と紐付けし，2 番目のキャラクターとする．ということである．また，ここでつけられた順番は，記述部分 3,4 に関わってくる．

3 はどのキャラクターがその場面にいるか，どのような行動をしているかを書いている部分である．書いてある場所としてはイベントのあらゆる箇所で記述されている．ここにキャラクターの表情や行動や立ち位置などを記述している．これを読み込むことで，立ち絵などの各班が用意したデータを呼び出す．今回の[r1,a 立つ,dE,f(0),e 悲しみ]の場合，1 番目のキャラクター（今回の例の場合はトーシャ）が，立つという動作をして，右を見ながら，画面左手前に位置し，悲しみという感情を表現している，ということである．

4 は現在のイベント中のその一瞬において，誰が主体（語り手や行動主）であるかを書いている．書いてある場所としては 3 と同じく，イベントのあらゆる箇所で記述されている．またこの部分には同時にセリフや地の文なども併記している．これは喋っている人物がほぼその瞬間の主体であるという予測によるものである．ただし，地の文においては誰が主体であるかを考えた上で，手動で主体を決めている．主体が存在しない場合などもあるため，そのような場合も対応できるようにしている．例えば，[g2,s2@「みんなを探そう」]と書かれていた場合，2 番目のキャラクター（今回の例の場合はディアリオ）が主体で，2 番目のキャラクターが@以降のセリフ（「」も含む）を話しているように表示する．ちなみに，[g9]のようにすると，暗転する．また，[s@]のように s の後に数字を入れなかった場合，名前を表示せずに地の文として表示される．

5 はゲームの進行に必要なフラグ部分，座標などの管理を書いている．書いてある場所としては先頭部分と末尾部分である．先頭の方では操作権を剥奪する記述のみが存在する．最後の方では操作権の復帰と諸々のフラグ管理を記述している．フラグの例としては，現在のイベントのフラグを取り消したり，次のイベントのフラグを立てたり，戦闘の有無などを記述して管理する．更に，イベント終了後のキャラクターの座標，マップなども合わせて記述している．今回の場合，上部分[cC(0)]は操作権を剥奪することを意味している．下部分は，イベント 4 のフラグを false にし，イベント 5 のフラグを true にし，操作権を戻す,イベント後に座標 4 に移動，ということを意味している．

6 はシーンという塊を表すのに必要な部分です．書いてある場所としては主にシーンの初めで記述されている．イベントとは複数のシーンが含んでいる．そのシーンとシーンの切れ目を明示的に記述したり，またどこにいるのか（背景）などを書いたりしている部分である．さらに，BGM の選択に必要な要素を記述したりなどもする予定である．これらをひとまとめにした理由としては，「場所の移動」がシーンの切れ目に大きく影響しているからである．また，シーンが切り替わればBGM 選択のための情報が再び必要と判断したからである．今回の場合，上部分は新しいシーンとして Map2 を展開し，それ

は非戦闘である．そしてその場面における音楽選択に必要な情報を音響班側に示している．下部分は，z 以上のシーンが終わった，その後の場面における音楽選択に必要な情報を音響班側に示している．

7 はそのイベントが終了することを伝える部分です．書いてある場所としては，イベントの最後である．この部分を読み込むことで，現在のイベントを終了する．つまり，現在のイベントの中で，1 で記述されていた現在のイベント起動に必要なフラグが消去された（false になった）としても，この部分を読み込まない限りイベントは終了しない．これは，[end]とのみ書き起こされる．end 以降に書き起こされている要素はない．

（＊文責：中村祥吾）

3.1.3 世界観設定について

RPG の世界観とは一括りに絞れるものではない．中世ヨーロッパのような世界観の「ドラゴンクエスト」や，ファンタジー世界を題材にした「ファイナルファンタジー」シリーズ，現代を舞台にした「ペルソナ」シリーズなど，物語の舞台は多岐にわたる．しかし世間一般的に RPG の舞台は「中世ヨーロッパ」の世界観を有している．それでいて「魔法」またはそれに近しい科学で説明の付かない力が存在していることが非常に多い．これらのことを総称して「王道 RPG」と言われる．それは RPG の元祖である「ドラゴンクエスト」シリーズの影響を強く受けているからだろうと推測し，早期に世界観を決めなければ背景や音楽の選別にも大きく影響が出ることを考慮し，シナリオもその世界観に則した．「ドラゴンクエストのような」という言葉で大凡のイメージがつけられるのも，大きな利点だった．イメージを共有できればそれぞれの班が各々作業を進めることができた．世界観の共有が終わった後は各自で作業を進め，時折できあがったキャラクターモデルや町のモデルを確認して，世界観の崩れないように努めた．

（＊文責：石川一稀）

3.1.4 キャラクター設定について

・トーシャ・リベルタ

本プロジェクトで一番始めに構想したキャラクターであり，本質的に本作の主人公といって差し支えないキャラクターである．ストーリー構成時「キャラクターは二人」という話はあったのが，デザインが完成していたのはこのキャラクターだけだった．まだ見ぬキャラクターからシナリオは想像し難かったので，このキャラクターを元にシナリオを構成していった．この物語ではプレイアブルキャラの幼なじみとして登場し，プロローグではこのキャラクターが視点となり，物語が進んでいく．決してプレイヤーが操作することはないキャラクターだが「物語序盤では自分で何かを決めることをしてこなかった彼女が進路に迷い，自分の未来を決めかねている」状態から「ストーリーを経て，自分の将来を自分で決める」という簡単だがセントラルクエスチョンを背負うキャラクターにしようと構想した．孤児院が襲われるという最初のエピソードも，幼なじみと慣れ親しんだ孤児院での毎日が愛しくて，一生このままの日々が続けばいいと考えた彼女にとって最大の事件と

いう位置づけにしたかった.

性別は主人公が男がいいという話し合いの結果からヒロイン枠として女性に，年齢は高校生くらいの想定が扱いやすいと決まり 17 歳前後ということになった．明確に決めない方が，動かしやすいと考えた．名前はフルネームでトーシャ・リベルタ．「トーシャ」の部分に意味はなく，完全に発音の響きの良さに任せて命名した．「リベルタ」の部分はイタリア語で「自由」という意味があり，「孤児院を含めて色々なモノに縛られてきた彼女が，物語を通して名前に相応しいくらい自由に伸び伸び生きていってほしい」という願いを込めてつけた.

・ディアリオについて

今作の主人公であり，プレイヤーが操作することになるキャラクターだ．トーシャのキャラクター設定を先行して作っていたため，トーシャの設定に合わせるようにキャラクターを設定した．ゲームの長さを考慮して，物語を通しての精神的変化を描くのは難しいと考え，精神的に年相応の完成度を持ったキャラクターに仕上げようと作成した．「幼なじみ」という設定が根底にあったので，「トーシャが自分で決めるのではなく，誰かの決定についていってしまう」というある種の欠点の理由をディアリオに帰属するものにした．「同時期に孤児院に連れてこられ，何かと一緒に扱われるようになり，自分で決めるのではなくディアリオの背中にくっついていくようになった」という形にすれば，それなり説得力が出ると考えた．またプロローグまでにはこのキャラクターが中心となるシーンはないので，後期に執筆予定のエピローグの方で視点にした話を構想していた．そうした方がトーシャの物語としていい見せ方ができると考えた.

性別は上記に記した通り，話し合いで男性に決まった．男性の名前をつけたことが経験上無かったため難航したが，「どうせならトーシャの好きな物を由来にしよう」となり，「毎日」の意味を持つ「ディアリオ」となった．男性キャラクターを描くのが数年ぶりだったので設定など大変苦心したが，どうにか形にした.

（＊文責：石川一稀）

3.1.5　中間発表で使用された脚本執筆

今回の脚本を担当した私として一番の難関は，問題は物語の登場人物の少なさにあった．本プロジェクトでは開発機関の短さを考慮し，「RPG として最低限成り立つ素材だけでゲームを作る」方針だった．よって人型キャラクターモデルは二体，エネミーモデル二体のみであり，マップも大きく分けて 3 つほどであった．プロローグを回す分には申し分ないマップ量であるが，問題はキャラクターモデルの数である.

これが RPG でない場合なら大した問題ではない．例を挙げるなら「夜廻り」（制作：日本一ソフトウェア）というゲームがある．公式ホームページを参照すると，ジャンルは「夜道探索アクション」となっている．「いなくなった姉と飼い犬を探しに，少女が夜の街を探索する」というストーリーの元，主人公の少女は夜の街風マップを探索する．探索する中で夜の街に住むお化けのような存在に襲われ，逃げながら物語を進めていくというホラーテイストの作品でもある．人型のキャラクターモデルは主な

登場人物である主人公の少女，その姉しかいない．一見同じ条件のようにみえるが，実はまったく別物である．

「夜廻り」の場合はホラーテイストでありつつ，その不思議で怪しげな夜の街を探索し，解き明かしていくゲーム性故に「世界観の説明」や「目的の明確化」「次の目的地への誘導」などが極力行われない．情報を開示しないことでプレイヤーの考察や自己解釈を深めていくことが想定されて作られている．しかし RPG の場合はそうはいかない．「旅に出る理由」に加え「世界観の説明」や「目的の明確化」「次の目的地への誘導」など，前者とは対照的に積極的に行われる．主人公にキャラクターモデルを一つ使ってしまうと，それらの役目をもう一人のキャラクターモデルに担ってもらうしかない．しかし「体験版のシナリオの想定プレイ時間は 30 分」ということが会議で決まっていた．さらに初期案には「恋愛要素をメインプロットと同等なサブプロットとして両立したい」と話に出ていたので，希望の時間内にキャラクター同士の親密性を高めていくことは難しいと考えて「幼なじみ」というある程度親密度の高い間柄を設定として取り込んだ．これによってある程度の強引な誘導が可能になると考えた．

素材も少なく，想定プレイ時間も少なかった為，物語や世界観に奥行きを持たせることは難しいと考え，最低限「話」として筋が通る程度のクオリティを目指して執筆を開始した．物語を構想するだけならそれほど難しい作業ではないのだが，「考えた構想がゲームで表現できるか？」という点を考慮すると困難を極めた．「演出的に不可能だ」とシステム班やデザイン班の要望を聞き入れながらシナリオを構築していった．

一番始めに考えたのは「三幕構成におけるどの部分を自動生成するか」という点だ．三幕構成というのは，脚本の構成術の一つであり，物語は三つの幕に分かれており，三つそれぞれの幕は設定（Set-up），対立（Confrontation），解決（Resolution）の役割を持つという物語モデルである．これは主に映画の脚本に適応されることが多いが，小説やコミックス，そしてゲームにも適応される．

主に設定（Set-up）では「主人公は誰で，何をする物語で，どのような状態であるか」を提示する一幕にする必要がある．この一幕は物語の基盤になるので，ここを自動生成することも挑戦としては非常にやりがいのある物になるとは思うのだが，ここが遅くなってしまうと，先述した通りデザイン班の町のモデリングや音響班の BGM 選出などに影響が不安視され，さらに物語構築で最重要になるセントラルクエスチョンの提示をしなければいけないので対称から外れた．セントラルクエスチョンとは，物語中に主人公が解決しなければいけない問題のことである．物語が終わるときに，この問いに Yes/No で答えられるのが物語の前提条件となっている．ドラゴンクエストシリーズに準えていうなら，「勇者は魔王を倒し，世界を救えるか？」というのがセントラルクエスチョンに該当し，ドラゴンクエストシリーズでは Yes といえるエンディングを迎えている．これは決して Yes で答えなければいけないということはない．

第三幕にあたる解決（Resolution)は，物語のクライマックスを担い，セントラルクエスチョンのへの回答はもちろん，同時並行で進むサブプロットの解決など役割が多く，一，二幕を経たキャラクター達の変化の証明をしなければいけない．ストーリーに解決をもたらす三幕も自動生成するにはあまり向いていないと考えた．それを行う為には一幕，二幕での出来事を意味合いを含めて理解していなければ作れないからである．

よってほぼ消去法的に第二幕の部分を自動生成することに決まった．けれど第二幕は三幕構成の中ではかなり自動生成に適しているともいえる．二幕は三幕中で一番長くなることが多く，前半と後半に分けて考えられる．前半では基本的に物事が順調に進み，障害を乗り越えていく．物語のちょうど中間に位置するミッドポイントを経て後半へ進むが，進むにつれて直面する障害は大きくなっていく．わかりやすくゲームシナリオで例えるなら，「ポケットモンスターシリーズ」（制作：ゲームフリーク）の「チャンピオンになるためにポケモンリーグの出場資格である８つジムバッチを１つずつ入手していく」過程がそれに該当する．この二幕ではメインストーリーの進行に含め，サブプロットも進行しなければいけないのでかなり多くの出来事が起こる．サブプロットとはメインストーリーとは違ったストーリーラインの話であり，同じように「ポケットモンスターシリーズ」で例えると「悪の組織との対決とその決着」がサブプロットとして語られている．多くの出来事を一人で構想するのは難しく，時間も多く消費してしまうので，ここを自動生成することが一番いいということが話し合いで決定した．
よって中間発表までの一幕の部分，ゲームのプロローグを執筆することになった．二幕の部分を自動生成するため，それに繋げやすい形で終わらせ，さらに想定プレイ時間が短いことも考えて，簡単にストーリーラインに乗せられる導入が必要だった．そこで「主人公の生活圏が襲われ，そこから巻き込まれていく」導入を採用した．さらに主人公たちを「孤児院育ち」という設定にすることで，「親を亡くしていることがそれほど珍しくない，村や町が襲われることのある世界観なんだ」とプレイヤーに伝える役割もあった．なるべく手短にＲＰＧの主軸となる「マップの探索」と「モンスターなどとの戦闘」を取り入れるためには効率のいい方法だった．元々プロローグ時点で使用できるフィールドマップは一つ，ダンジョンが一つだったので「孤児院が襲われる」というイベントを起こすことによって一度に解決できた．さらに「孤児院に住む人達が攫われた」ことで，本来いなければ不自然な人のキャラクターモデルを準備できない欠点を補った．それらを済ませた後は冒険する方向へ向かわせなければいけないので，「孤児院の院長が，攫われたまま」というわかりやすい旅路の目的を提示した．大凡，ここまでがプロローグとして書き起こしたシナリオ執筆の過程だ．

（＊文責：石川一稀）

3.2　システム班

中間発表までの開発では，開発予定のシステム全体を大きく「シナリオ読み取り」「戦闘」「マップ」「UI」の 4 つの部分に分けて分担し，物語生成班が自動生成を行わずに作成したテキストデータを読み込み，RPG として一連の流れを出力するシステムの開発を行った．

（＊文責：宍戸建元）

3.2.1 シナリオ読み取りシステム

3.2.1.1 シナリオ読み取りシステムの概要

今回実装したシナリオ読み取りシステムでは，物語分析班と音響班が作成したテキストデータをもとに，各イベントで起こるイベントシーンの読み取りや，音楽の再生等を行う．イベントシーンの読み取りは，

マップ情報が与えられた場合にのみ行う．そのため，勝手に物語が進んだりすることがないようになっている．

（＊文責：稲垣武）

3.2.1.2 シナリオ読み取りシステムにおけるクラスの定義

クラス名：God

機能：ゲーム起動時に必ず呼ばれるクラス

　　オブジェクト名とマップ名を受け取りその情報を scenarioCut クラスに送信

　　戦闘シーンの切り替えも行う

メソッド名：textMove

機能：tagProcess クラスの tagCut メソッドの呼び出し

引数：なし

返り値：なし

クラス名：Main

メソッド名：first

機能：物語班と音響班が作成したテキストデータの読み取りを行う

　　scenarioLoader クラスの textCut メソッドの呼び出しを行う

　　eventInfo クラスの変数 sText に値を送る

　　eventInfo クラスの Extraction メソッドの呼び出しを行う

引数：なし

返り値：なし

クラス名：scenarioLoader

メソッド名：textCut

機能：Main 関数から受けとったテキストデータを X で区切る

　　つまり，イベントごとに分割する

引数：なし

返り値：なし

クラス名：scenarioCut

メソッド名：textCut

機能：イベントを分割し，タグごとに読み取れるようにする

　　God クラスから受け取ったオブジェクト名とマップ名をもとに，該当するイベントを選択する

イベントをシーンごとに分割し，さらに，改行ごとに分割，コンマごとに分割する

分割したものをキューに保存し，順次読み込みを可能とする

引数：なし

返り値：なし

クラス名：eventInfo

機能：キーがマップ情報で，値がイベント名のディクショナリー，EventMapObj の定義を行う

キーがシステム側で用意したイベント名で，値がイベント名のディクショナリー，EventName の定義を行う

キーがイベント名で，値がフラグのディクショナリー，Event の定義を行う

キーがフラグで，値が bool 値のディクショナリー，EventFlag の定義を行う

キーがマップ名で，値が bool 値のディクショナリー，EventMap の定義を行う

キーがマップ名とオブジェクト名で，値がマップ名のディクショナリー，nextMap の定義を行う

メソッド名：Extraction

機能：イベントごとにイベント名とオブジェクト名とマップ名の抽出を行う

抽出したイベント名，オブジェクト名，マップ名を用いて，定義したディクショナリーに格納していく

引数：なし

返り値：なし

メソッド名：GetMapFlag

機能：map が移動可能か確認する

引数：抽出したマップ名

返り値：1.引数の値が EventMap に含まれている場合，EventMap のキーを引数にして得られる値

2.引数の値が EventMap に含まれていない場合，false

メソッド名：SetMapFlag

機能：map の移動可能情報をセットする

引数：抽出したマップ名と bool 値

返り値：なし

メソッド名：DoEventCheck

機能：map と obj 情報からイベントが実行可能か確認する

引数：int 型の整数と，抽出したマップ名とオブジェクト名

返り値：1.引数の値が EventMapObj に含まれている場合，EventMapObj のキーを引数にし，得られた値を Event のキーとして得られた値を EventFlag のキーとして得られた値

2.その他の場合，false

メソッド名：DoEvent

機能：map と obj 情報からイベントのテキストを取得する

引数：int 型の整数と，抽出したマップ名とオブジェクト名

返り値：1.引数の値が EventMapObj に含まれている，かつ，引数の値が DoEventCheck に含まれている場合，EventMapObj のキーを引数として得られた値を，Event のキーとして得られたものを Flags クラスの GetText メソッドを用いて得られたテキスト

2.その他の場合，””

メソッド名：ChangeFlag

機能：フラグの変更を行う

引数：Flags クラスの flag 変数と bool 値

返り値：なし

クラス名：Flags

メソッド名：GetText

機能：テキストのゲッターメソッド

引数：なし

返り値：text

クラス名：tagProcess

メソッド名：tagCut

機能：scenarioCut でキューに入れたテキストデータを 1 つずつ出して，タグの読み取りを行う

引数：なし

返り値：なし

（＊文責：稲垣武）

3.2.1.3 シナリオ読み取りシステムの詳細説明

シナリオ読み取りシステムは，ゲームが開始された際にシナリオの読み込みが開始される．また，シナリオの読み込みが開始された場合，同時にイベントごとに区切られる．そして，イベントごとに区切られたものをディクショナリーに保存する．ディクショナリーに保存することによって，欲しい時に欲しいイベントを再生することができる．

マップから送られたマップ名とオブジェクト名と一致するイベントをディクショナリーから検索し，そのイベントが再生可能なフラグが立っていれば，イベントの再生を開始する．この時に，イベントをシーン，文章，タグごとに分割を行う．そして，タグを読み取ることで，イベントシーンを再生していく．

（＊文責：稲垣武）

3.2.2 戦闘

戦闘システムの開発工程では，RPG の戦闘について，主に「ステータス管理」と，「戦闘の実行」，2 つの部分について担当した．戦闘システムでは，テキストデータから受け取った編成 ID と場所 ID を元に戦闘画面と敵の情報を作成し，プレイヤーの行動に合わせて各キャラクター固有のパラメータ(以下，ステータス)の内容を変更させることで戦闘を進めて行く．

（＊文責：宍戸建元）

3.2.2.1 戦闘の実行

RPG の戦闘は，キャラクターごとの固有なパラメータによって行動の選択肢が変わり，その結果についてもパラメータによって算出され反映される．また，戦闘の終了条件も，自身のパラメータである体力が 0 にならないように攻撃を加え，敵の体力を 0 にすることである．このように，ステータスを管理する部分は，RPG の重要な部分であり，その管理を行いながらゲームを進める必要がある．

今回，ステータスの管理方法として，HP や攻撃力などの共通のパラメータを持つ baseStatus と名付けたクラスを作成し，そのクラスのインスタンスをそれぞれの List に入れることで管理を行った．baseStatus クラスには，名前，体力(以下 HP)の最大値，マジックポイント(以下 MP)の最大値，現在の HP，現在の MP，攻撃力，防御力，素早さ，の 8 つのパラメータが定義されている．それぞれのパラメータについて，baseStatus クラスのインスタンスが作成される際に初期値が設定され，GetStatus 関数で値を参照し，SetStatus 関数で値を変更することが可能である．また，戦闘の際のダメージ計算は，baseStatus クラスの中に Damaged 関数が定義されており，攻撃する側の攻撃力を引数として渡すことで，攻撃で受けるダメージの量を自身の防御力との計算で求め HP を変更させる．この baseStatus クラスを用いて，敵・味方，双方のステータスの情報を定義した．

プレイヤーが操作できるキャラクターのステータスについては，charactorStatus というクラスを作成し，その中で管理を行う．charactorStatus クラスには，baseStatus クラスのインスタンスの配列である playerList という配列を作成し，操作できるキャラクターのステータスをその中に格納することで管理を行う．characterStatus クラスはゲーム起動中，常に存在しており，それぞれの HP，MP などの情報を保持しているため，UI 上やマップ上からも参照し，アイテムを使用するなどしてその値を変更することが可能である．

characterStatus クラスの中には，後述する item クラスを格納する itemList が存在し，プレイヤー所持している item の管理も行っている．

また，同様に敵キャラクターのステータスについても，enemyStatus というクラスを作成し，その中で管理を行う．enemyStatus クラスには，baseStatus の配列であり，敵のステータスを格納する enemyList が存在している．enemyList に格納すべき敵のステータスは，eBooks という配列に定義されており，ゲーム中で登場する敵のパラメータと固有の ID が紐づけられている．戦闘を開始する Battle 関数が呼び出された時，引数として受け取った enemy_id を受け取り，enemy_id に照合する ID の敵ステータスを eBooks 配列から探し出し，その内容にそったステータスの baseStatus クラスが作成され，敵として enemyList に格納される．

戦闘の際は，characterStatus クラスと enemyStatus クラスの情報を参照して，プレイヤーの行動の結果を，パラメータの変化によって反映させていく．

（＊文責：宍戸建元）

3.2.2.2 戦闘の実行

中間発表までの戦闘システムでは，プレイヤー側と敵側で 2 対 1 の戦闘を行うシステムを作成した．図 3.2.2.2-1 は Unity 上での実行画面である．

図 3.2.2.2-1 Unity 上での実行画面

戦闘は，マップやイベントなど，他のシーンから呼び出されて使用されることを前提として作成されている．戦闘を開始する Battle 関数が実行される際，forBattlePrefub という，戦闘の処理にのみ必要なものが作成される．forBattlePrefub の中には，表示される敵のイラストや UI 表示のために必要なテキスト枠の画像，敵のステータスを管理する enemyStatus クラス，戦闘全体の流れを管理する BattleController クラスが含まれている．forBattle 作成後，Battle 関数の引数である enemy_id から敵の

ステータスを作成し enemyStatus クラスに格納，background_id から戦闘時に使用する背景が決定され，作成される．

戦闘中は BattleStatusText と呼ばれる部分にプレイヤーの操作するキャラクターの名前，HP，MP の情報を表示し，BattleUIText と呼ばれる部分に行動できる選択肢の表示や，プレイヤー，敵の行動とその結果を表示していく．

戦闘に必要なものや情報が作成された後，戦闘中に変化する UI やグラフィック，パラメータの管理などの戦闘を実行する際の変更を BattleController クラスを中心として行う．BattleController クラスには，battleQueue と呼ばれるキューが存在する．キューには，プレイヤーが行動を選択し，行動の結果を反映し，戦闘を続行と終了の判定を行う，という戦闘の一連の流れの呼び出す enemyEncount というコルーチンが格納されている．このコルーチンを Queue から取り出して処理を行い，戦闘続行判定の際に再度コルーチンを格納することで，戦闘の一連の流れが循環するように処理を行っている．

戦闘中，プレイヤーはキャラクターごとにとるべき行動を選択することでゲームを進めていく．プレイヤーの行動選択部分の実装は，キャラクターが行うことが可能な行動を数値に対応させ，draw 関数の中で選択した行動を数値として command 配列に格納する．command 配列は行動の結果を表示させるタイミングでその中身を参照し，中身によって回復・攻撃など行う処理を変化させ，パラメータの管理を行う．

プレイヤーがとることのできる行動は，中間発表までの時点で攻撃 2 種類(攻撃，ライトニング)，回復(キュア)，逃げるの 4 つが存在する．行動結果の反映順は，素早さの値にランダムな値を掛け合わせ，その値の大小でプレイヤー側と敵側，どちらが先に動くかを決定する．決定されたあと，その順番通りに行動が実行され，プレイヤー側は command 配列に記憶された行動を，敵側ならプレイヤー側のキャラクターをランダムに選びそちらに攻撃行動を行う．

中間発表までの制作では，戦闘の終了条件を敵の HP が 0 になった場合のみに制限した．また，終了の際に forBattlePrefub を削除することで演出に関わる部分を削除し，戦闘後のキャラクターのステータスのみを反映させ，戦闘を終了する．

中間発表までは，ここまで説明した戦闘の一連の流れを作成し，イベントから戦闘部分が呼び出されて実行から終了されるまでの実装を行った．問題点としては，プレイヤーが操作するキャラクターの HP が 0 になった際，敗北となった際の処理が完成していないことがあげられる．また，プレイヤーが操作するキャラクターの処理はその総数によって可変な動作になっているが，敵側の行動については可変ではないため，今後は敵対プレイヤーで 2 対 2 で戦う等，より多くのキャラクターに対応できるようにシステムを変更していく必要がある．加えて，アイテムの使用やキャラクター固有の選択肢，敵キャラクターのパラメータや行動による難易度の調整等も改善点としてあげられるため，今後はそれらについて改善を進めていく．（＊文責：宍戸建元）

3.2.3 マップ

上下移動，左右移動に応じてフラグを建て，キーが押されたとき，別のキーが押されながら，別のキーが離されたとき，両方のキーが離されたときの三つの分岐を用いて，滑らかな移動を実装した．また，それ以外にもアニメーションの制御，追跡するキャラクターのための座標記録，物体に衝突したときの処理など，移動に関する処理を行う．

3.2.3.1 マップのクラス定義

クラス名：box

メソッド名：SceneLoaded

機能：シーンをロードしたときに，値の初期化や，シーン移動前の座標を記録する．

引数：Scene, LoadSceneMode

返り値：なし

メソッド名：Move

機能：マップ移動上の全体制御を行う．

引数：なし

返り値：なし

メソッド名：MoveCheck

機能：物体に対する衝突の有無によって，プレイヤーの移動を制限する．

引数：Vector3

返り値：float

メソッド名：HitCheck

機能：物体に衝突しているか判定する．

引数：Vector3

返り値：bool

メソッド名：HitItemName

機能：衝突したオブジェクトを取得する．

引数：Vector3

返り値：GameObject

メソッド名：HitTile

機能：足元のオブジェクトを取得する．

引数：Vector3

返り値：GameObject

メソッド名：EventCheck

機能：イベントの有無を判定し，実行可能であれば実行する．

引数：Vector3

返り値：なし

クラス名：SubPlayer

メソッド名：SubMove

機能：操作キャラの座標を任意の距離を置いたうえで，追跡する．

引数：なし

返り値：なし

（＊文責：太田和宏）

3.2.4 UI 開発

UI の開発にあたり，まずメニュー画面の作成を行った．メニュー画面はマップ上で開くことができ，「どうぐ」を使用したり，キャラクターのステータスを確認することができる．中間発表の時点では，複数のウィンドウを開閉させる機能，矢印で選択肢を選ぶことができる機能，キャラクターたちへ「どうぐ」を使い，ステータスを変動させる機能まで作成した．

（＊文責：西川和真）

3.2.4.1 矢印の表示

最初に，選択肢を矢印で選ぶと，特定のウィンドウが開閉する機能の開発に取り組んだ．

図.3.2.4.1 マップ上のメニュー画面

矢印がどの選択肢を選んでいるかを知るために Cursor クラスを作成した．Cursor クラスの Move_cur メソッドによって矢印を PC の方向キーで移動できるようにし，移動先のポジションを整数として返すことができる．移動先のポジションは，図を例にすると，「どうぐ」が 1，「そうび」が 2，「なかま」が 3，「じゅもん」が 4，「とくぎ」が 5，「セーブ」が 6 となっている．Move_cur メソッドは現在の矢印自身のポジションとポジションの最大値を引数とする．方向キーが押されると矢印の xy 座標を移動させ，ポジションを増減させ，その数値を返す．

矢印のポジションを任意の選択肢に移動させ，E キーを押すとポジションに対応したウィンドウが開かれる．開いたウィンドウは，X キーで新しいものから消せるように，E キーで全ての開かれたウィンドウを消せるようにした．

（＊文責：西川和真）

3.2.4.2 ウィンドウの作成

Unity におけるオブジェクト間の情報のやり取りは，オブジェクト同士が親子関係であると管理がしやすい．そのためウィンドウの開閉は，親ウィンドウから子ウィンドウへの管理で行えるようにした．最初に開かれる「Some back」ウィンドウを親ウィンドウとし，6 つの子ウィンドウを制作した．子ウィンドウの中には，さらに子ウィンドウが存在し，入れ子構造のような状態である．

その中の「どうぐ」を選ぶと，「Item back」ウィンドウが開かれる．

（＊文責：西川和真）

3.2.4.3 作成クラス

「どうぐ」を使用，破棄するために，アイテムを管理する item クラス，baseStatus クラス，charactorStatusClass クラスを作成した．

BaseStatus クラスは，キャラクターのパラメータを設定し，管理するクラスである．baseStatus クラスは，キャラクターの名前，最大 HP（キャラクターの体力を表す数値），最大 MP（キャラクターが「じゅもん」を使うために必要な数値）などの整数を引数として，キャラクターのパラメータのインスタンスを生成する．「どうぐ」を使うにあたって，Healed メソッドと，Recoveried メソッドを作成した．Healed メソッドは，HP の回復量を表す整数を引数として，その数値分キャラクターの HP を増加させる．Recoveried メソッドは，MP の回復量を表す整数を引数として，その数値分キャラクターの MP を増加させる．

item クラスは，「どうぐ」の名前と，種類を識別する整数，効果量を表す整数の 3 つの引数を用いてインスタンスが生成される．種類を識別する整数は，0 を HP 回復，1 を MP 回復，2 を戦闘不能になったキャラクターの復活，3 を捨てることができないフラグ管理の「どうぐ」と分けた．「どうぐ」を使うにあたって，ItemEffect メソッド，GetItemName メソッドを作成した．ItemEffect メソッドは，baseStatus 型のデータを引数として，引数としたキャラクターのパラメータをアイテムの効果に沿って増減させる．もしインスタンスの種類識別値が 0 であれば，HP の数値を効果量分，増加させる．2 で

あれば，キャラクターの HP が 0 のとき，HP を増加させる．3 であれば，「それを捨てるなんてとんでもない！」というログが出力され，何も起こらない．GetItemName メソッドは，「どうぐ」の名前を返すメソッドである．

charactorStatusClass クラスは，baseStatus 型データと item 型データの集合を管理するクラスである．baseStatus をキャラクターの名前をキーとした Dictionary で管理している．item を List で管理している．「どうぐ」の使用，破棄をするにあたって，UseItem メソッドと，RemoveItem メソッドを作成した．UseItem メソッドは整数と baseStatus 型データを引数としている．List の整数番目の item を参照し，baseStatus を引数として item の ItemEffect メソッドを実行する．その後，整数番目の item を消去する．RemoveItem メソッドは，整数を引数として，List の整数番目の item を消去する．

この 3 つのクラスを用いて，「どうぐ」を使う処理を行う．

（＊文責：西川和真）

3.2.4.4 「どうぐ」の使用

「Item back」を開くと，item の List から名前を取り出し，表示する．矢印で「どうぐ」を選んだあと，「Use back」が開かれる．「つかう」を選ぶと，「Who back」が開かれる．そこでキャラクターを選ぶと，そのキャラクターの名前をキーとして baseStatus を検索する．その baseStatus を用い，矢印ポジションの整数番目の List の item を使用する．「すてる」を選ぶと矢印ポジションの整数番目の List の item を破棄する．

図 3.2.3.4　メニュー画面

（＊文責：西川和真）

3.2.4.5 今後の展望

このように前期では，複数のウィンドウを開閉させる機能，矢印で選択肢を選ぶことができる機能，キャラクターたちへ「どうぐ」を使い，ステータスを変動させる機能までを制作した．これを踏まえ，後期では以下の機能の実装を考えている．

- メニュー画面のデザインの向上
- キャラクターの攻撃力等を増減させる「そうび」の実装
- 「どうぐ」や「そうび」の解説を行うウィンドウの作成

（＊文責：西川和真）

3.2.5 音楽

音楽システムでは，音楽について「BGM」「SE」の二項目に分け，MusicController クラスという BGM と SE の両方を管理するクラスを作成した．このクラスはゲーム起動中は常に存在し，他のクラスからも StartSE 関数，StartBGM 関数を用いることで BGM や SE を変更することが可能である．

MusicController から，BGM や SE の管理を行うため，BGMController クラス，SEController クラスを作成した．どちらのクラスも AudioSource 配列という，ゲーム中で使用するすべての BGM，SE が格納された配列が定義されている．BGM や SE を変えたい場合は，引数として BGMNumber，SENumber という整数値を受け取り，AudioSource 配列で添え字がその数値と同じである BGM や SE を流す仕組みを用いている．

BGMController は，BGM の変更を行う際，ぶつ切りに曲を変化させるのではなく，フェードアウトやフェードインを用いて変更を行う必要がある．そのため，BGMController に FadeIn 関数と FadeOut 関数を定義した．BGM を変更する際は，まず FadeOut 関数で BGM のボリュームが 0 以下になるまで下げ続ける．その後 BGM を，AudioSource 配列で添え字が BGMNumber と同じである BGM に変更する．最後に FadeIn 関数によって BGM のボリューム徐々に上げていくことで，フェードアウト・フェードインを使用した音楽変更を実装している．

中間発表までの制作の中では，戦闘から MusicController の関数を呼び出すことで，BGM や SE を演出に併せながら鳴らす部分までを実装した．しかし，現在はマップ画面や UI，イベント画面との連携が取れておらず戦闘画面への実装にのみとどまっているため，今後は音響班の作成するテキストデータからの音楽の変化や，マップの切り替えによる BGM の変化，UI 画面やマップ画面で起きる細かな SE にも対応できるよう改修を進めていく必要がある．

（＊文責：宍戸建元）

3.3 音響班

3.3.1 音楽分析

3.3.1.1 分析作品の検討

物語に対して自動選曲するためには場面の傾向を取り，曲の詳細を参照して相互関係を分析する必要があると考えた．本項では音響班が分析に取り組むまでに行った作業と分析作品の選定について記述する．

始めに音響班は音楽について学ぶことにした．音響班には音楽についての知識がある人が 1 人しかおらず，全体として知識不足であると考えた．そのため音楽の基礎知識を得ることで曲の詳細について自動選曲に必要であると思われる曲の要素を考察出来るようにするためである．

その後分析する作品について検討した．分析対象として，物語分析班が自動生成した物語に対して自動で選曲するために同系統の作品を分析することが一番良い分析結果が得られるのではないかと考えた．そのため「ドラゴンクエスト 5」や「ファイナルファンタジーシリーズ」等の物語分析班の分析する世界観に則したゲームタイトルを選択した．ドラゴンクエストやファイナルファンタジーは世界的にも認知度があり，数多くのナンバリングタイトルが存在する作品群である．

（＊文責：長野恭介）

3.3.1.2 分析作品の予行

ゲーム分析を始める前に，アニメでの分析を行った．理由としては，場面における分析におけるタグの模索をするため，班全員の分析傾向を把握し同じ場面で同じ分析が出来るよう読み取りの統一化を図るためである．アニメ作品は「転生したらスライムだった件」の 2 話で分析を行った．この作品は 2018 年 10 月に放送され，原作の売り上げも良く続編の製作も決定しているという点で分析の予行をするには適切だと考えた．この時，分析項目としてシーンの概要，シーンの機能，主体の動作，主体の感情，場所，雰囲気を分析のタグとして使用した．この分析の予行でタグの内容を細かく設定すると分析結果にバラつきが生まれることが分かった．その後タグの再考を行うため，各自で必要だと判断したタグを用意し，個別に分かれて RPG の分析を行った．各自が用意したタグにはウラジミール・プロップが発見した「昔話の構造 31 の機能分類」を用いたものもあった．この RPG 分析で，物語分析の手法を用いて主体の感情を細分化したタグを用意したが，主体の感情と場面に用いられる曲との関連性が見られないという知見が得られた．最終的に決定したタグの詳細については後述の分析方法で説明する．

（＊文責：長野恭介）

3.3.1.3 分析方法

まず分析方法について記述する．我々は分析作品の傾向から BGM のテンポが場面の雰囲気を左右するのではないかという仮説を立てた．そこで，ゲームシナリオにおいて場面の要素にタグを付けて分析し，その場面で流れている曲の詳細な情報を場面の状況に結びつけるという手法を用いた．この手法を用いることにより，物語上の場面の要素からその場面で流れる曲を推定するための構造を探ることにした．

次に場面の要素に付けたタグについて記述する．今回の場面の分析では，タグを「Situation」「Character」「Place」「Mode」の 4 つに分けた．その中でも「Situation」と「Character」については明るい，暗い，

無しの 3 つの要素で分析した．「Place」は前述の 3 つに加えて町の要素を加えた 4 つに，「Mode」はイベントシーン，戦闘シーン，マップでの移動シーンの 3 つの要素で分析を行った．

（＊文責：長野恭介）

3.3.1.4 分析結果

まず，上記の仮説に対しての結果について記述する．戦闘シーンにおけるテンポは殆どのゲームで 150 以上の数値であった．さらに全体として暗いシーンでは 90 未満のシーンが多く，マップでの移動シーンでは 90 以上 125 未満が殆どであるという結果となった．

ただし，今回の分析はデータ量が少なく「ドラゴンクエスト 5」「ファイナルファンタジー7」「ポケットモンスターlet's go シリーズ」の 3 作品のみであった．また，プロローグ付近のシナリオでは曲の使いまわしが多く，場面の分析に比べると曲の分析データ量が少なかった．このため，さらなる分析を進めて十分なデータ量を揃えるまでは仮説の域を出ないだろうという結論となった．

（＊文責：長野恭介）

3.3.2 曲決定アルゴリズム

分析によって生み出された曲のテンポがシーンの雰囲気を左右するという仮説のもと，BGM の音楽的要素のうちテンポのみに着目し，シーンが含む情報からそのシーンに妥当とされるテンポを推定し，その結果に基づき BGM を決定するアルゴリズムを作成した．

（＊文責：山内拓真）

3.3.2.1 決定木によるテンポの推定

シーンに妥当なテンポを推定するために今回は決定木を使用した．決定木とは教師データあり機械学習の手法の一つであり，目的変数を最もよく分類する説明変数の分岐を生成することで，観察結果から目標に関する結論を予測する木のモデルである．決定木を用いた理由としては，結果を可視化することで分類に至る過程の解釈が容易なこと，今までの講義で学んだ手法であることなどが挙げられる．今回は決定木を作成するために Python のライブラリ scikit-learn を使用した．また，分類アルゴリズムは CART 法，分類の指標はエントロピーを用い，木の深さを 3 に制限して木を作成した．説明変数を Situation, Character, Place Mode，目的変数にテンポを設定し，シーン情報によってテンポ帯が定まるモデルを作成し，得られたモデルを図 3.3.2.1-1 に示す．

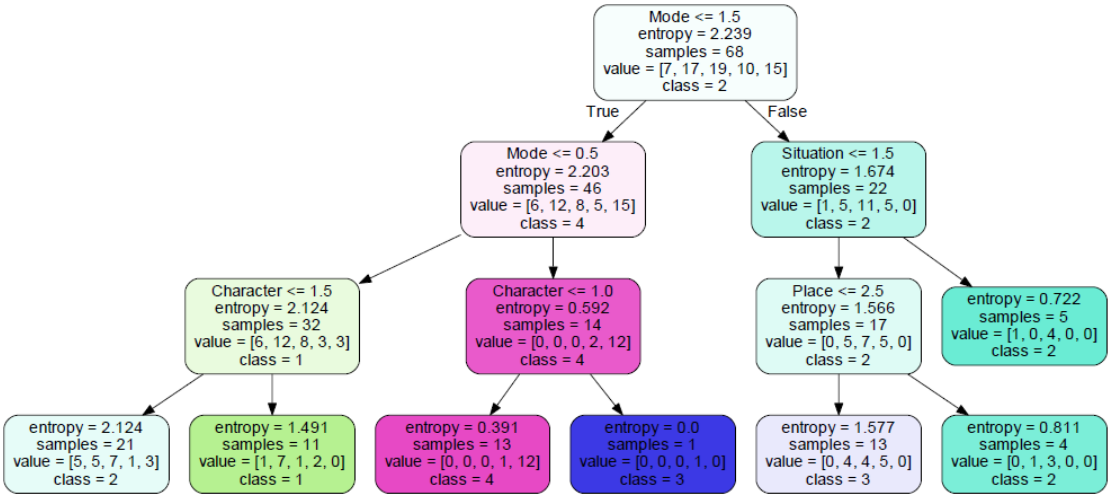


図 3.3.2.1-1

結果，最も上手く要素を分類できたルートノードは，Mode: Battle であるか，すなわち戦闘状態にあるか否か，となった．今回のテストシナリオ中のシーンから具体的に例を示すと，Mode: Battle, Place: Dark のシーンでは，妥当なテンポは 160 以上という結果になった．

（＊文責：山内拓真）

3.3.2.2 ルールベース

決定木によって妥当とされるテンポ帯が決定した後，最終的に BGM を決定するためにルールベースを用いた．ルールベースとは，人が登録したルールの集合である．

登録したルールにはテンポ帯から曲を決定する場合と特殊なシーンの曲を決定する場合の 2 パターンが考えられるが，中間発表まででは前者のみの実装となった．

例として，テンポから曲を決定する場合のルールのうち 1 つを挙げると，決定木により妥当なテンポが１６０以上と推定された場合，BPM165 の曲の番号が選択されるという If 文である．

（＊文責：山内拓真）

3.3.2.3 結果

現段階では決定木モデルについての評価を行っていないため，シーンにあった BGM を選択できているかどうかは判断できていない．しかし，分析シーン数が 70 シーン程しかなく十分な学習用データを収集できていない点，CART 法を採用していることにより説明変数を連続的に捉えている点，また説明変数とした属性はどれも主観で選択したものであることを考えるとシーンに最適な曲を決定するアルゴリズムとしてはまだ不完全であると考えられる．

（＊文責：山内拓真）

3.3.3 後期への課題

今回は分析結果に基づきシーンに妥当な曲のテンポを決定するアルゴリズムを作成したが，現段階では分類したテンポ帯の数と同様の 5 種類の曲という非常に狭い範囲の自動選択しか実現できていない．曲に含まれる音楽的要素はテンポだけではなく，リズムや安定度などが存在する．シーン毎に最適な BGM を決定するためには複数の音楽的要素を考慮し曲を選択する必要がある．また，BGM のテンポに影響を与えていると考えたシーン中の属性 SItuation，Character，Place，Mode が妥当であるかどうかは分かっていないので，評価を繰り返し属性を吟味していくことにより，シーンの情報と BGM の音楽的要素の関係性が明らかになって行くと考えられる．

（＊文責：山内拓真）

3.4 視覚班

3.4.1 コンセプトアート

ゲーム制作を行う上で，プロジェクトメンバー全員が制作するゲーム作品に対して共通の世界観とイメージを持つことが重要である．特に，視覚班の作業は元となるコンセプトアートを参考にする作業が多く，早急に世界観の可視化をし，メンバー間で認識を共有させる必要があった．今回は物語分析班が書き起こしたテストプロットを元に，物語の可視化をすることで全体の作業をより円滑にするために，ステージ・キャラクター・アイテム等のコンセプトアート複数枚を描いた．大まかなワークフローとして，紙と鉛筆でコンセプトアートのラフスケッチを行い，メンバーからのフィードバックを参考にしつつ修正や追記を行い，Photoshop とペンタブレットでペイントをし，仕上がったものを Google Drive に共有するワークフローをとった．メンバー間との話し合いにより，本作業は，物語分析班からテストプロットを貰ってから，一番時初めに着手すべき作業であると認識し，早い段階で取り掛かった作業である．早い段階でコンセプトアートを仕上げたことによって，モデリング作業やゲームのシステム設計の作業に早い段階で取り掛かることができた．

（＊文責：小川卓也）

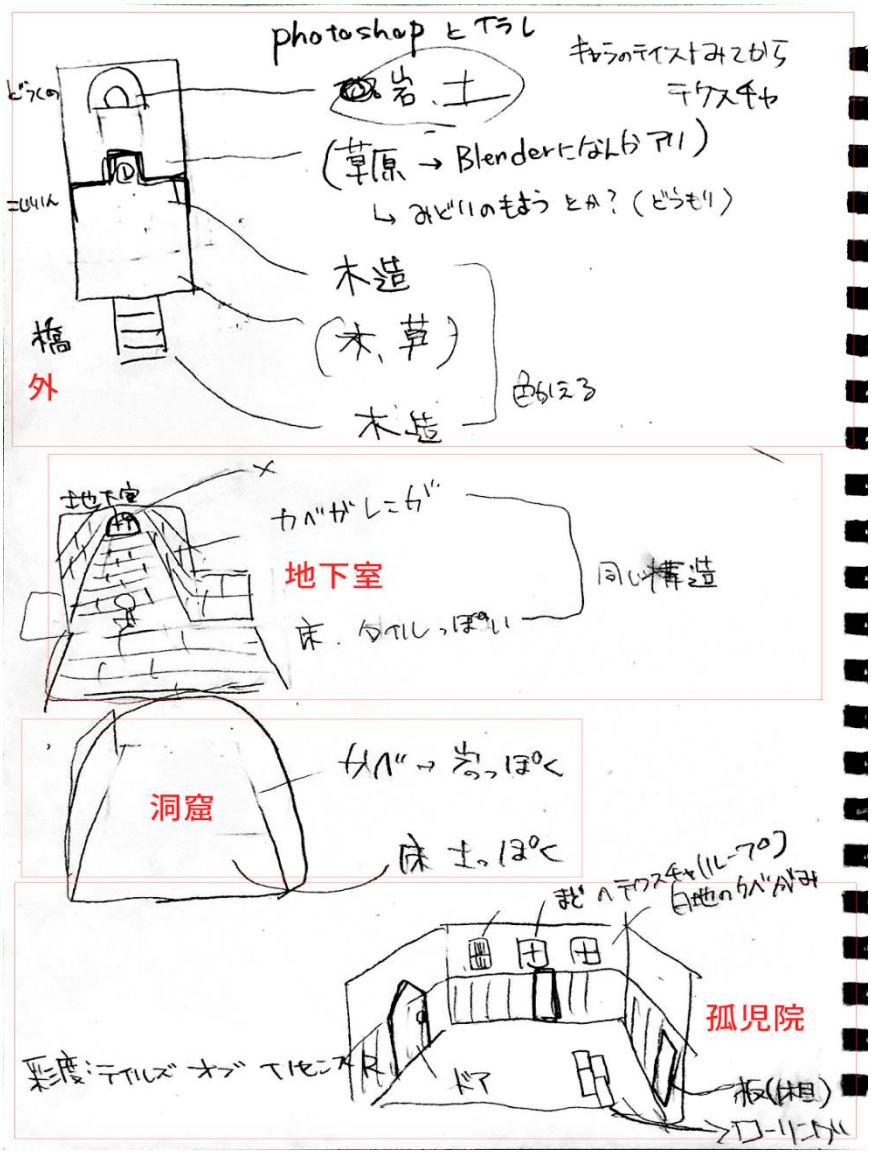
3.4.1.1 世界観設定について

3.4.1.1.1 ステージの美術設定

物語分析班が書き起こしたテストプロットを参考に，どんなステージが必要なのかを決定した．今回必要とされたステージは，孤児院と洞窟を含む外のステージ，孤児院内部のステージ 2 種，地下室のステージ，洞窟内のステージの 4 つである．ステージデザインをする上で，作業スケジュールとマップのスケール感によるゲーム進行のロスを考慮して，マップのスケール感が大きくなりすぎないこ

とを心がけた．初めに，舞台となる中世ヨーロッパの建物を参考に，紙におおよそのステージのイメージ画をスケッチし，どんなステージを制作するのかを全員に共有した．全員に承諾を貰った後，Photoshop とペンタブレットで具体的なステージ設定を制作した．ステージ設定を全メンバーに共有する際は，ビジュアルの上に具体的な美術設定をわかりやすく記述することを心がけた．反省点としては，建物のマテリアルやアイテムについての書き込みが足りなかったことで，美術設定を見たメンバーの中で，これらに関する不明点や疑問点が度々生じたことである．美術設定をする際には，建物の壁の種類やアイテムについての書き込みをもう少し増やすことは好ましいと感じた．また，一部美術設定で，インターネットから拾ってきた画像を加工して使用しているものがあるため，実績としては好ましくない手法である．

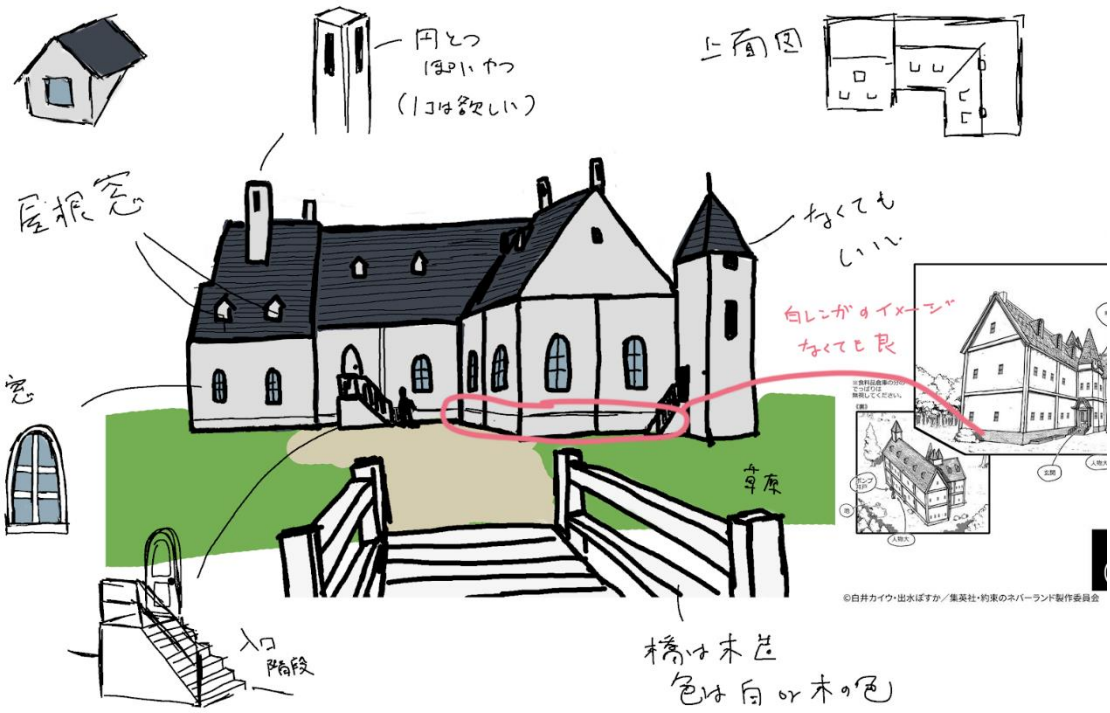
（＊文責：小川卓也）



3.4.1.1.1 おおよそのステージのイメージ画

3.4.1.1.2　孤児院（外装）とその周辺

孤児院は，実在した中世ヨーロッパの建造物と，孤児院が登場するアニメーション作品を参考にした．階数は２階とし，在籍人数十数名の規模のものを想定してデザインした．外装の着色に関しては，中世のイメージを損なわないように薄暗めの色に設定した．孤児院の建物の次に，周辺のステージのデザインを行った．まず，物語分析班のメンバーが仕上げたテストプロットから必要なマップの要素を抜き出し，見取り図としてラフスケッチしたものをテストプロットを作成した本人に相違がないか確認してもらった．相違がないのを確認し，Photoshop でラフをなぞり，孤児院とその周辺の美術設定を制作した．



CQ 孤児院 全体設定資料

図 3.4.1.1.2 孤児院（外装）とその周辺のステージ設定

3.4.1.1.3　孤児院（内装）

孤児院内部は，廊下と食堂の２種デザインした．内装画に加えて，プレイヤーの移動可能範囲を指定して記述した．反省点としては，Photoshop でトレースとペイントを施してデータ化する時間がなかったので，ラフスケッチでしか共有できなかったことである．ラフスケッチだと細部についての説明や色の指定などがないので，あまり好ましくない．

（＊文責：小川卓也）

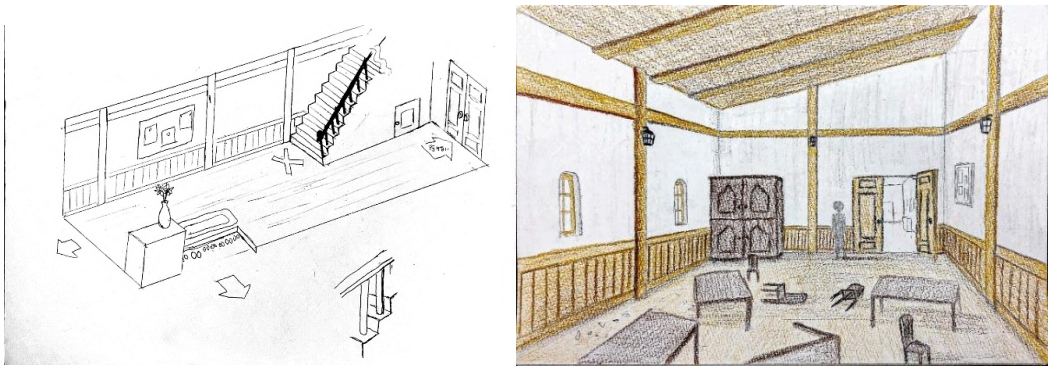


図 3.4.1.1.3 孤児院（内装）のステージ設定

3.4.1.1.4　地下室

地下室は，実在するレンガ調の地下通路を参考にデザインした．物語序盤に登場するステージなので，比較的シンプルな構造設計にした．地下室奥には，プレイヤーが拾える武器のイメージを記述した．しかし，地下室と後述の洞窟の画においては，インターネットで拾った画像に手を加えて制作しているため，この手法は，個人の実績として公開できるものではないため，もしこの手法を参考にするのであれば，著作権の観点で注意が必要である．

（＊文責：小川卓也）

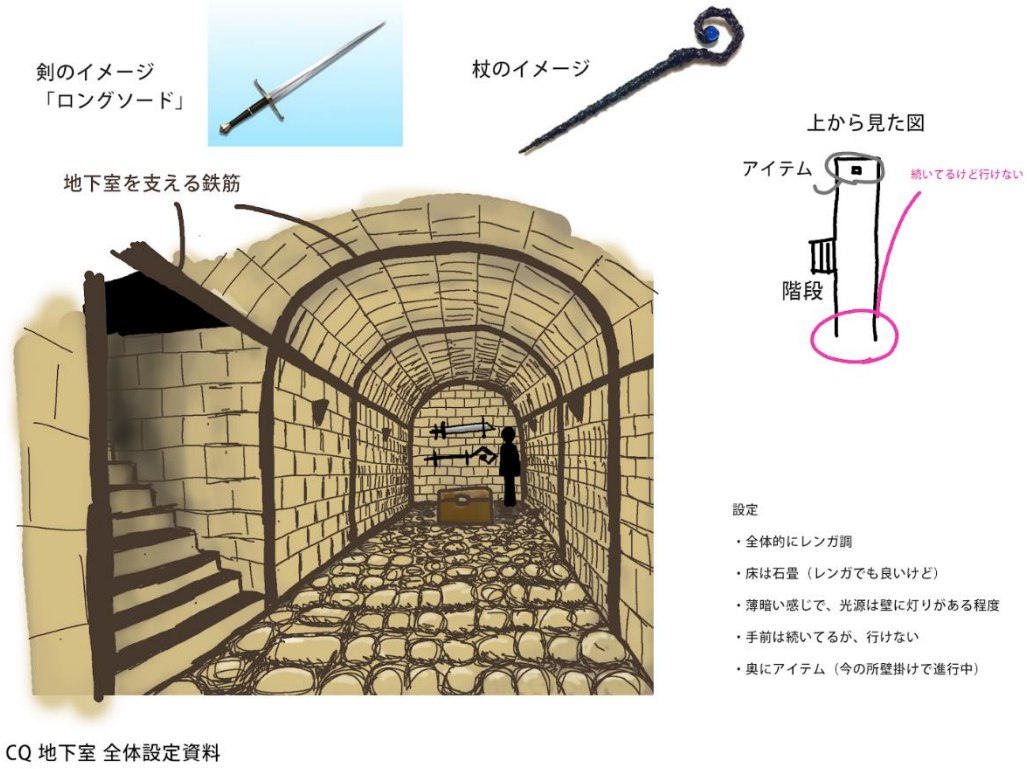


図 3.4.1.1.4 地下室のステージ設定

3.4.1.1.5　洞窟

洞窟は，外観，内部ダンジョンのイメージ，最深部のイメージの 3 種を制作した．ダンジョンの具体的なイメージや設定が比較的可変なものであったため，具体的な設定記述は施さなかった．

（＊文責：小川卓也）

図 3.4.1.1.5 洞窟内部のステージ設定

（＊文責：小川卓也）

3.4.2 モデル（モーション含む）

3.4.2.1 モデリング

モデリングとは 3 次元コンピューターグラフィックスで立体的な形状の物質を形成することである．本プロジェクトでは視覚的な表現を 3 次元空間で表現するため，オブジェクトに 3D モデルを使用している．視覚班ではゲーム内で扱うモデルの規格を統一するために素体作成し，それをベースにして作成した．

（＊文責：蓬畑旺周）

3.4.2.1.1 モデリングの手順

モデリングの際に使用したツールは無料で公開されている 3DCG 作成ソフトの Blender を用いた．また，いくつかのモデルのモーション作成にも用いた．以下にモデル作成の手順を記す．

（＊文責：蓬畑旺周）

3.4.2.1.1.1 素体の作成

初めに資料として，インターネット上で無料配布されている三面図の入手と，公開されている素体モデルの観察を行った．それらをもとに服や髪のついていないベースとなる素体モデルを作成した．作成した素体モデルは男性用，女性用，ゲーム内で動作する 2 頭身モデルの 3 種類である．

（＊文責：蓬畑旺周）

3.4.2.1.1.2 キャラクターのラフモデリング

作成した素体をベースに，キャラクターデザインとしてアウトプットされたイラストから，髪型や服装等を大まかに設計した．キャラクターデザインの製作者と小まめに連絡を取りながら作業をすることで，製作者のイメージから大きく逸れることがないよう心掛けた．

（＊文責：蓬畑旺周）

3.4.2.1.1.3 細部・小物の作成

ラフモデリングで作成したモデルの情報量を増やすために，より細部の部分に手を付けた．ラフモデリングの際に作成しなかった小物の類や衣装の調整を行った．前述のラフモデリング同様にキャラクターデザインの製作者との小まめな連絡をすることで，最終的なイメージを決定し，視覚化させた．2頭身モデルは一部を等身大モデルから衣装データーを流用することで作業量の削減と，衣装イメージの統一を図った．

（＊文責：蓬畑旺周）

3.4.2.2　モーション

キャラクターのモーションの作成にはモデリング同様 Blender を用いた．今回は人型キャラクターには歩行モーションを適応した．瞬き，待機モーション等の実装は時間削減のために見送ることとした．キャラクターにモーションを適応，出力する際の手順を以下に記す．

（＊文責：蓬畑旺周）

3.4.2.3.1　ボーンの実装

モデルにモーションを適応させるために，人間の骨に値するボーンを配置する．ボーンを配置，実装することでモデル自身に関節部分を作成する．2頭身モデルのボーン配置については，腕と足のボーンを削減して関節部分を減らすことでモーション作成の難易度を下げた．

（＊文責：蓬畑旺周）

3.4.2.3.2　ウェイトペイント

モデルにボーンの配置を適応させた後にウェイトペイントを行う．ウェイトとは特定のボーンが移動した際に，モデル部分に動きを反映させるものである．ウェイトには 0 から 1 の値が存在し，その間の値は小数で表される．ウェイトが 0 の場合は全くモデルの動きに反映されない．対して値が 1 の場合は対応するボーンに対してモデルが追従するようになる．小数点の値では 0 に近ければモデルの動きの反映は少なく，値が 1 に近ければモデルの追従は大きくなる．

（＊文責：蓬畑旺周）

3.4.2.3.3　キーフレームの設定

キーフレームとは，CG アニメーションの中で定義されるフレームのことである．数フレームごとに物体の位置や回転をすることで，その間を補完するアニメーションである．今回はモデルにモーションを付与する際に，フレームごとにボーンを動かして一連のモーションを作成した．一つのモデルに複数のモーションを設定することもあり，各モーションにループ可能なフレーム数の調整を行った．今回は 1 フレームから 40 フレームの間でモーションを組み込んだ．20 フレームをモーションの中間点とし，10 フレーム目と 30 フレーム目にそれぞれ左右対象となるような歩行モーションを設定した．40 フレーム以降は 1 フレーム目に戻るような設定にすることで，1 モーションをループさせることにした．

（＊文責：蓬畑旺周）

3.4.2.3.4　モデルの出力

Blender で作成したモデルデータとモーションデータをシステム班に受け渡すことが必要である．モデルとモーションデータの含まれる blend ファイルを，Unity 側で読み込ませるために fbx 形式で出力した．

（＊文責：蓬畑旺周）

3.4.2.4 仮データの制作

仮データとは実際にゲーム内で使用するモデルではなく，システム班に後々共有するであろう簡略化されたモデルの事である．モデルを作りこまず簡略化することで，短い期間のうちに共有をすることが可能であった．仮データの作製の意義は，早めにデータをシステムに共有するためである．なぜ早めにシステム班に渡す必要があったのかというと，ゲーム制作における動作確認を進めるためである．動作確認を早く進めることで，システム内でバグなどがでても早め早めに対処することが可能である．

（＊文責：友広純々野）

3.4.3 ステージの制作

制作されたステージデザインを基に Blender と Unity を用いてステージの制作をおこなった．大まかなステージ製作法は，Blender で制作したモデルを fbx 形式で出力し，この fbx 形式のデータを Unity の Scene に反映させるという流れである．Unity の Scene とは，ゲームの環境とメニューが含まれているものである．各 Scene ファイルは一意のレベルであるため，構築する環境やモデルを各 Scene 毎に設定，配置をすることが可能である．ステージごとに各 Scene ファイルを制作することで，ステージ分けを容易におこなうことができる．なぜ Unity に反映させる作業をおこなったかというと，システム班への共有をする際にデータの崩れを抑えることができるらである．例えば，すべて Blender 側で調整をし，Unity へ反映させると，データが崩れてしまう部分が出てきてしまった．そこで，Blender でモデルを製作し，Unity 上でライティングなどの調整をおこなう事で，データの崩れを防ぎ，無駄な手間を省くことができた．中間発表までの成果物は，テストシナリオの一番最初のシーンで使用する，全体のステージ，2 番目のシーンで使用する孤児院内部のステージ，3 番目のシーンで使用する孤児院地下室のステージ，最終シーンであるダンジョン内の道に使用するモデルである．

（＊文責：友広純々野）

3.4.3.1　ステージの詳細について

3.4.3.1.1　はじめのステージ

このシーンは地形，橋，川，孤児院，ダンジョンの 5 要素で構成されている．地形は Unity で標準に搭載されている Terrain で制作した．前期の時点では，このステージの地形は森の設定であったが，後期で決定した町の地形に合わせて緑の少ない砂地のような地形に変更した．その他の橋，川，孤児院，ダンジョンのモデルは Blender で制作したものを使用した．

（＊文責：友広純々野）

3.4.3.1.2 孤児院内部のステージ

このステージは，建物 1 件に対して 3 つの部屋，部屋の中には机や椅子，本棚等の小物で構成されている．ストーリーの整合性を取るために，3 部屋のうち一部屋には裏口があり，一部屋には地下へと続く道があり，最後の部屋は敵の破壊行動により，進むことができないように設定をしてある．

（＊文責：友広純々野）

3.4.3.1.3　孤児院地下のステージ

このステージは，階段，部屋，主人公たちが手に入れることができる武器などで構成されている．また，他のステージより暗めのライティングである．

（＊文責：友広純々野）

3.4.4 マテリアル

モデルのマテリアルを制作するのは，Substance painter 上でのテクスチャ，マテリアルの編集をする段階と，それらを Unity 上で適応する段階に分かれる．Substance painter で制作したマテリアルはダンジョンに用いる通路のモデルのものとヒロインの二頭身モデルのものである．
通路のモデルは岩の質感を再現するためフラクタルノイズをハイトマップに用いた．マップの自動生成を行うため地面のテクスチャは滑らかにつながるようにした．
ヒロインのマテリアルは肌や髪，服など複数の質感を用いた．デフォルメされた 2 頭身キャラのため肌はリアルではなく 多少のノイズがある程度の密度にした． また，二頭身にデフォルメしたため，顔の比率が大きくなるそこで目はイラストソフトを用いて密度を上げた．服の装飾の金属部分は あまり目立ちすぎない程度に金属感を与えた．布部分は 布目のノイズを考慮した上で密度を決定した．
テクスチャの書き出しは Unity で使用することを前提に行った．ノーマルマップは Unity の仕様に合わせて Open CV のものを用いた．Unity ではラフネスの値を設定するために メタリックマップのアルフ

ァ値を用いるためそれに合わせて書き出しをした．これらを Unity にインポートしマテリアルに適用した．

（＊文責：白石智誠）

3.4.5 まとめ

3.4.5.1　前期の活動の流れ

3.4.5.1.1　作業項目の洗い出し，画面仕様の決定

全体で決定した世界観やジャンルを基にプロトタイプを制作することが決定したため，それに合わせて（1）ゲームに必要な要素の洗い出し，（2）その要素の中から視覚化すべき要素の絞り込みをし，制作すべき優先度を決定した．この優先度から，プロトタイプ制作に最低限必要な要素を決定した．

次に２D か３DCG で視覚部分を制作するか検討した．初期の頃は２D のドットで表現をする案も出たが，生成されるシナリオに対してより自由度の高い見せ方を実現することができる３DCG に決定した．自由度の高さにおける利点として，下記が上げられる．

・３D 素体を一度作ることでキャラクターに様々なポーズを付与することが可能

・モデル制作の際，一つのモデルを他のモデルに派生することが可能

前期の製作は，視覚化した成果物を早めに共有するため，キャラクターやステージのデザインをおこなう者と，そのデザインを基にモデリングをおこなう者の，大きく２つに作業を分担した．

分担することで，作業効率が上がり，早い段階でメンバーにゲーム内のイメージを共有することができた．

（＊文責：友広純々野）

3.4.5.1.2　目的

前期の目的として，最終的に開発するゲームイメージの可視化をすることで，メンバーの認識を統一し，作業を円滑にする目的があった．

（＊文責：友広純々野）

3.5 中間発表

2019 年 7 月 19 日に公立はこだて未来大学にて中間発表が行われた．中間発表を行うにあたって，プロジェクトの活動内容を説明するための A1 サイズのポスター５枚とスライド，制作したゲームのデモ動画を用意し，20 分のプレゼンを６回行った．また，プレゼンの聴講者に発表技術と発表内容についてのアンケートに回答してもらい，評価を受けた．

アンケートは発表技術と発表内容の項目に分け，10 段階評価とコメントを回答してもらった.

中間発表で 82 人から受けた評価は表 3.4.1 に示す結果となった.

表 3.4.1 中間発表での評価

評価点	発表技術	発表内容
1	0	0
2	0	0
3	1	0
4	1	0
5	2	1
6	4	5
7	13	7
8	26	26
9	16	15
10	11	16
無回答	8	12
平均	8.03	8.39

発表技術についてのコメントの一例を示す.

- 全体説明良かったです．視覚班のポスターもよく出来てました.
- プレゼン資料がとてもシンプルで伝わりやすい内容でよかった．説明も全体を見渡しながら行っていたためとても分かり易かった.
- スライドの字が小さくて見づらい部分があった
- 聞き手に向かって発表出来ていた.
- 内容を人に説明できるまで理解しているのを感じた．専門予後の説明もかみ砕いていて分かり易くなっていた.
- 分かり易い発表だと思う．チラチラスマホ見てるのが気になった.
- 明らかにカンペをガン見するのは我慢しよう！→内容が飛んだら，スライドの内容説明でもOK．スライド見やすい←ストーリーがしっかりしている.
- デザイン的には見やすくていいけど，文字量が多く具体的に把握しづらかった．班ごとに詳細発表を分けたのは分かり易くて良い.

- ボードやスライドを指で追ってくれているのでどこの説明かが分かり易い．手ぶりもあるからうまい．相手を見ている．

発表内容についてのコメントの一例を示す．

- 「面白い」ことを目指すうえで拡販で工夫しようとしている点がわかるともっとよかったです．
- ヒットゲームを分析してより面白いゲームを作るという発想はよかった．プログラミングの開発環境が限られるため十分ではないがここまでよくやっていると思う．
- 必要となる技術から十分にできる見込みのある計画が練られていてよかった．
- システムの説明の間に工夫話があるのは集中が途切れる．恐らくプロジェクトで最も重要である「イベントの自動生成」の内容が不明に感じた．
- やっていることは面白そうなので頑張って完成させてほしい．
- 課題まで含めて面白いゲームになりそう
- プロジェクト学習の期間は限られているので，システムの最低要件を設定したことはとても良いことだと感じる．
- 困難そうではあるが，実現不可とは断言できないため，期待が持てる．

発表技術については，アンケートの結果から発表者が発表原稿を見ているのが気になるという意見が多かった．また，スライドの文字が小さいという意見もあったため，発表をするスペースを考慮した発表準備が課題であると考える．

発表内容については，面白いゲームの定義やそれらを分析して本当に面白いシナリオができるのかといった意見が多かった．また，決定した事項についてなぜそのようにしたのかという理由をあまり説明していなかったため，RPG を題材にしたことや 3 DCG を用いることについてなどの理由を細かく説明する必要があると感じた．これらのことについて，後期は具体的に方針を決める必要があると考える．

（＊文責：白石智誠）

第 4 章 最終発表までの開発

4.1 物語分析班

4.1.1 作品分析

4.1.1.1 分析作品の再検討

今回，「売り上げが多い」，「シリーズが長く続いている」「認知度が高い」の 3 つに当てはまるものを「面白い」RPG 作品であると定義して，選定を行った．また，ゲームジャンルを決定した際の意見から，恋愛要素が含まれているということや，実際にそのゲーム作品が入手できるということも条件とした．以上の条件から，前期には，「FINAL FANTASY IV」，「FINAL FANTASY VI」，「FINAL FANTASY

Ⅶ」，「FINAL FANTASY Ⅷ」，「FINAL FANTASY Ⅸ」，「ドラゴンクエスト」，「ドラゴンクエスト Ⅴ 天空の花嫁」，「テイルズ オブ エクシリア」，「シャイニング・ハーツ」，「サモンナイト」，「ルーンファクトリー4」の 9 作品を分析対象としていたが，後期では，前期の分析結果から，よりよい分析データを抽出するべく，「ドラゴンクエスト」シリーズ（スクウェア・エニックス）と「FINAL FANTASY」シリーズ（スクウェア・エニックス）に対象を絞ることにした．

最終的に，分析対象は「FINAL FANTASY」シリーズ（スクウェア・エニックス）から，「FINAL FANTASY」，「FINAL FANTASY Ⅱ」「FINAL FANTASY Ⅲ」「FINAL FANTASY Ⅳ」「FINAL FANTASY Ⅳ」「FINAL FANTASY Ⅴ」，「FINAL FANTASY Ⅵ」，「FINAL FANTASY Ⅶ」の 7 作品，「ドラゴンクエスト」シリーズ（スクウェア・エニックス）から，「ドラゴンクエスト」，「ドラゴンクエスト Ⅱ 悪霊の神々」，「ドラゴンクエスト Ⅲ そして伝説へ…」，「ドラゴンクエスト Ⅳ 導かれし者たち」，「ドラゴンクエスト Ⅴ 天空の花嫁」，「ドラゴンクエスト Ⅵ 幻の大地」，「ドラゴンクエスト Ⅶ エデンの戦士たち」，「ドラゴンクエスト Ⅷ 空と海と大地と呪われし姫君」，「ドラゴンクエスト Ⅸ 星空の守り人」「ドラゴンクエスト Ⅺ 過ぎ去りし時を求めて」の 9 作品，合計 16 作品を分析対象とすることにした．

（＊文責：斉藤勇璃）

4.1.1.2 分析方法

中間以前に行った分析結果をもとに新しい分析方法を検討した．クエストシナリオを自動生成するにはクエストがどのような形で成り立っているかを定義する必要があった． また，クエスト同士がどのような形で成り立っているかも同時に定義する必要があった． 今回はロールプレイングゲームのシナリオはクエストがいくつも連続することで成り立っていると定義した． 「ドラゴンクエスト」を例として説明する． 「ドラゴンクエスト」では竜王の城に行くためのアイテムである虹のしずくを手に入れるために，太陽の石と雨雲の杖を入手する必要がある． この虹のしずくを入手するために「洞窟内の魔物を倒して雨雲の杖を入手する」クエストを終わらせ，「太陽の石を入手する」クエストを行わなければいけない．この 2 つのクエストを終わらせて初めて「竜王を倒す」クエストに進むことができる． このようにロールプレイングゲームのシナリオは前のクエストが終わらないと進めることができない連続性を持っているといえる． また中間以前の分析結果から 1 つのクエストは発生・経過・結末の 3 つの段階で成り立っていると考えた． 「敵の討伐を依頼される」というクエストの発生が起これば，必ず敵を倒すという経過が得られ，敵の討伐を達成したことを依頼主に言えば「情報」「アイテム」など何らかの報酬を得ることができるというこの構造がすべてのクエストにあるといえる． これらをもとに発生には 5 つ，経過には 4 つ，結末には 3 つのパターンがあると定義した． 発生のパターンは「アイテム入手依頼」「敵討伐依頼」「捜索依頼」「偵察依頼」「アイテム入手による道の開放」「敵討伐による道の開放」「討伐すべき敵が出現するハプニング」「人が拉致・監禁されるなどのハプニング」の 8 つ，経過のパターンは「目的地の提示あり・敵の出現あり」「目的地の提示あり・敵の出現なし」「目的地の提示なし・敵の出現あり」「目的地の提示なし・敵の出現なし」の 4 つ，結末は「アイテムの入手」「情報の入手」「道の開放」の 3 つである．「アイテム入手依頼」は何らかのアイテムを探してくる依頼である． 「敵討伐依

頼」は敵を倒してくることを誰かに頼まれることである．「捜索依頼」は行方不明の人物や生き物を探してくる依頼である．「アイテム入手による道の開放」は船などに乗る際に乗船券が必要だった際，その乗船券を入手することで次の街などに進めるようになることである．「敵討伐による道の開放」は道をふさいでいる敵を倒すことで新しい街に行けるようになることである．「目的地の提示あり」はクエスト発生時にどこに行くべきか提示されているパターンである．「目的地の提示なし」はクエスト発生時に何をやるべきか提示されているが行き先が明確でないパターンである．この定義に合わせて次のような分析方法をとった．まず，RPG のシナリオを中間以前に行ったものと同じように物語全体をシーンごとに分割し，その中で起こったことを決定したタグに当てはめて分析を行った．ここでいうカテゴリータグとは物語を分析するために決定したある程度のパターンをまとめて表記したものである．次にこの分析結果を，先ほど述べた発生・経過・結末の 3 段階に分ける分析を行った．これを行うことにより，RPG のクエスト 1 つはどのような形で成り立っているかを明確にすることができる．また RPG 内のクエスト同士がどのような形で連続しているかについても同時に分析を行った．

（＊文責：宇田朗子）

4.1.1.3 分析結果

今回分析を行ったのは「ドラゴンクエスト」「ドラゴンクエストⅡ」「ドラゴンクエストⅢ」「ドラゴンクエストⅣ」「ドラゴンクエストⅤ」「ドラゴンクエストⅥ」「ドラゴンクエストⅧ」「ドラゴンクエストⅨ」「ドラゴンクエストⅪ」「ファイナルファンタジー」「ファイナルファンタジーⅡ」「ファイナルファンタジーⅢ」「ファイナルファンタジーⅣ」「ファイナルファンタジーⅤ」「ファイナルファンタジーⅥ」「ファイナルファンタジーⅦ」の 16 作品である．この 16 作品から 412 個以上のクエストの分析を行った．ここから前述した分析方法を利用して RPG 内にあるクエストシナリオがどのような 3 段の構造で成り立っているか，またクエストの連続性について分析した．最初に 3 段構造の分析結果について記述する．前述した 16 作品のゲーム内のクエストを発生・経過・結末の 3 段階に分け，それぞれの段階でどのカテゴリータグがどれほどの数があるかを集計した．この結果よりロールプレイングゲームのクエスト内では多くの特徴があることが発見された．例として，ロールプレイングゲームのクエストの大半は「アイテム入手依頼」が多いこと，「討伐すべき敵が出現するハプニング」と「敵討伐依頼」が起こった時は必ずボスとなる敵が出てくることがあげられる．また，分析結果をもとに，クエストシナリオがどのように連続しているかの遷移確率モデルを抽出した．

（＊文責：宇田朗子）

4.1.2 データフォーマット

4.1.2.1 概要

シナリオの自動生成をするにあたり,様々なシナリオの書き方を定義する必要があった.本項では,中間発表までの活動,データフォーマット,タグ（タグについては後ほど）,クエストの生成方法について説明することとする.それぞれについて背景と経過,詳細を書いていく.

（＊文責：中村祥吾）

4.1.2.2 データフォーマットの必要性

今回ゲームのために用意されたシナリオは,キャラクターの動きや感情,セリフなどをすでに明示的,もしくは暗示的に示している.しかし,シナリオテキストをそのまま機械に通しても,機械は処理ができない.また,ゆくゆくはシナリオの書き起こしまで自動で行うため,書き起こしの規則を決めなければならない.そのため,機械でもどのような要素がこの場面に存在しているかわかるように,また誰が書いても同じような形式で書き起こせるような規則を決定する必要があった.

まず,発表用に用意したシナリオを機械でも読み取れるようにシナリオデータを整形する作業の必要が出てきた.この作業を,「データフォーマット」とした.渡されたシナリオにおいて,1:場面に存在する要素をカテゴリーごとに分類,2:カテゴリーごとにタグ付け,を行なった.例えば,必要な要素として「キャラクターの立ち位置」があった.立ち位置はfのタグで表すことにして,f(0),f(1),のように記した.これは,「f」が「位置を表すタグ」であり,fの後に続く数字が「キャラクターの位置」である.このようなタグを前述の場面ごとの要素に割り振っていった.タグの割り振り方,つまりはデータフォーマットの規則を図.3.1.2-1に示した.図中の1〜7の番号は,後の「記述内容の各部分の説明」でも扱う.

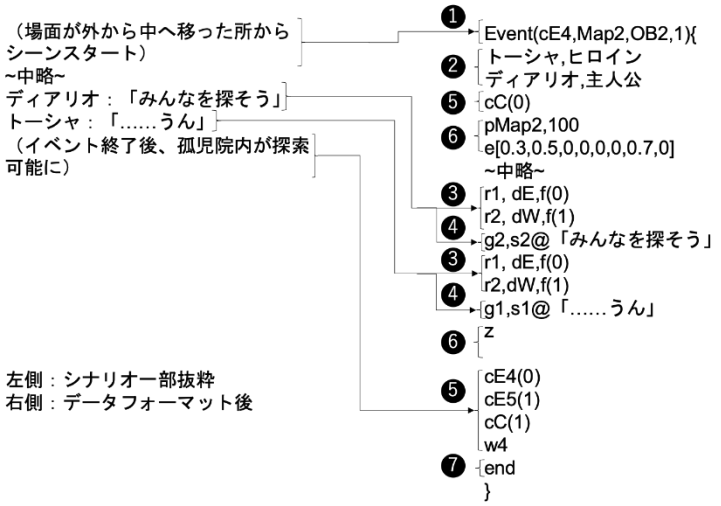


図.3.1.2-1.シナリオーデータフォーマットの対応

データフォーマットの規則については,大方は前年度の規則を踏襲しつつ,前年度にはなかった要素（様々なフラグの管理,音楽班に渡す情報など）を新たに含めるように手を加えていった.また,カテゴリーの中でも,更に種類を分けてタグ付けを行った.これによって,よりわかりやすく詳細な管理,書き起こしを可能にした.例えば,フラグ管理を行うために新たなタグ「c」を作った.しかし,フラグとは様々な種類が存在する.「戦闘」フラグや,「イベント発生」フラグなどである.そのような場合,「c」の後にさらに「B」や「E」などを付与できるようにすることで,種類ごとの管理を可能にした.これにより,「cB」というタグを見れば,「フラグ管理用のタグである.特に戦闘フラグを管理しているものだ」と分かるようになった.

（＊文責：中村祥吾）

4.1.2.3 対象を絞っていった経緯

ただし,場面内の全ての要素を書き起こすのは現実的ではない.よって RPG のある 1 場面の中でも,後々自動生成を行うにあたって必要になると予測された要素を優先することとした.優先する要素に関しては,各班のメンバーと随時話し合い,どの要素が各班に必要か,最終的に用意するべきものの見通しを立てた.今回のプロジェクトにおいて優先された要素は,「物語の進行・面白さに必要な要素」や「自動生成に影響を与える要素」であった.

物語の進行に必要な要素として,どこにいるか（背景）,どのキャラクターを表示するか,現在誰が話しているか,フラグの管理などが挙げられた.物語の面白さに必要な要素として,シーンの主体となるキャラクターがどんな感情を持っているか,BGM や SE の管理,シナリオの変化などが挙げられた.自動生成に必要な要素として,BGM の自動生成に必要な情報,主体が誰か,シーンの切れ目などが挙げられた.また,柔軟かつ矛盾のないシナリオ生成のために,テキストの挿入箇所を示すことも挙げられた.現時点で必要と思われるタグの概要を表.3.1.2-1 に示す.それぞれのタグの詳細は後の「タグの詳細」に書く.また,テキストの挿入については「クエストの生成方法」に書く.

表.3.1.2-1

タグ	タグの意味
P	背景・場所の宣言
C	フラグ
R	キャラクターの指定
D	キャラクターの向き
F	キャラクターの位置
E	シーンの感情
G	カット情報
s@	セリフを話すキャラクター名
W	座標宣言
Z	シーンの終わり
End	イベントの終わり
WP	イベント終了時にマップ移動する際に使用
M	BGM 番号
#n	メインテキスト内とサブテキスト内に存在する.テキストの挿入場所,内容を表す.サブテキスト内の#nと#nの間にある内容

	をメインテキスト内の#nの場所に挿入する. #nには数字ごとにそれぞれ決まった種類の内容を記述するようにしている. 詳細は「クエストの生成方法」に記述する.

中間発表時点からの変更点を上げていく.視覚班と話し合った結果キャラクターの立ち絵の差分を用意しないこととなったため,立ち絵の種類の宣言が不要になったので「a」タグを削除した.音楽班に渡す情報は,新たに変更された「e」タグにまとめることが可能になったため「mc」タグを削除した.音楽班がテキストに BGM の番号を記述することとなったために必要になったため「m」タグを追加した.中間時点では BGM の選択基準にキャラクターそれぞれの感情を使用することとなっていたが,主体 1 人の感情を使用することになったため「e」タグの表記内容を変更した.中間発表時点では RPG として必要最低限のフラグのみに絞って記述していたが,RPG のシナリオの自動生成に必要なフラグの種類が増えたため「c」タグの種類を追加した.RPG のシナリオの自動生成において,柔軟性と物語の前後での矛盾を減らすため,「#」タグの追加をした.

（＊文責：中村祥吾）

4.1.2.4 記述内容の各部分の説明

データフォーマットで記述した内容は,1:イベント起動部分 2:宣言部分,3:キャラクター表示部分,4:主体決定・テキスト表示部分,5:フラグ管理部分,6:シーンの開始部分,7:イベント終了部分に分かれている.以降本項では,数字は図.3.1.2-1 に対応させて説明する.

1 はそのイベントが起こるための条件・イベントの形式などを書いている部分である.書いてある場所としてはテキストの最初である.ここに書いてある条件に合致しなければ(false であった場合）,イベントは発生しないようになっている.今回の場合,[Event(cE4,Map2,OB2,1)］とあるので,イベント 4 のフラグが true であり,Map2 におり,OB2 というオブジェクトに触ったら,形式 1（立ち絵形式）でイベントを始める,ということである.

2 は現在のイベント内でどのキャラクターが出るかを書いている部分である.書いてある場所としては 1 の直下のみである.ここでモデルや立ち絵とキャラクターの名前を紐づけて,宣言された順番にナンバリングしていく.

今回のように書いてあれば,ヒロインという名前の立ち絵を,トーシャという名前と紐付けし,1 番目のキャラクターとする.主人公という名前の立ち絵を,ディアリオという名前と紐付けし,2 番目のキャラクターとする.ということである.また,ここでつけられた順番は,記述部分 3,4 に関わってくる.

3 はキャラクターをどのように表示するかを書いている部分である.書いてある場所としてはイベントのあらゆる箇所である.ここに 2 で定義した何番のキャラクターが,左右どちらを向いているか,どの立ち位置にいるかを記述している.これを読み込むことで,視覚班が用意した立ち絵を適宜呼び出す.今回の

[r1,dE,f(0)]の場合,1 番目のキャラクター（今回の例の場合はトーシャ）が,右を見ながら,画面左手前に位置している,ということである.

4 は現在のイベント中のその一瞬において,誰が主体（語り手や行動主）であるかを書いている.またこの部分には同時にセリフや地の文なども併記している.書いてある場所としては 3 と同じく,イベントのあらゆる箇所である.セリフなども併記しているのは,セリフを話している人物がほぼその瞬間の主体であるという考えによるものである.ただし,地の文においては誰が主体であるかを考えた上で,手動で主体を決めている.主体が存在しない場合などもあるため,そのような場合も対応できるようにしている.例えば,[g2,s2@「みんなを探そう」]と書かれていた場合,2 番目のキャラクター（今回の例の場合はディアリオ）が主体で,2 番目のキャラクターが@以降のセリフ（「」も含む）を話しているように表示する.ちなみに,[g9]のようにすると,暗転する.また,[s@]のように s の後に数字を入れなかった場合,名前を表示せずに地の文として表示される.

5 はゲームの進行に必要なフラグ部分,座標などの管理を書いている.書いてある場所としてはテキストの冒頭と末尾である.冒頭では操作権を剥奪する記述のみが存在する.末尾の方では操作権の復帰やアイテムの取得・喪失など諸々のフラグ管理を記述している.フラグの管理例としては,現在のイベントフラグの取り消し,次のイベントフラグを立てる,戦闘のフラグを立てるなどである.更に,イベント終了後のキャラクターの座標,マップなども合わせて記述している.今回の場合,冒頭の[cC(0)]は操作権を剥奪することを意味している.末尾の[cE4(0)cE5(1)cC(1)w4]は,イベント 4 のフラグを false にし,イベント 5 のフラグを true にし,操作権を戻し,イベント後に座標 4 に移動,ということを意味している.

6 はシーンという塊を表すのに必要な部分である.書いてある場所としては主にシーンの初めで記述されている.イベントには複数のシーンが含まれている.そのシーンとシーンの切れ目を明示的に記述したり,またどこにいるのか（背景）などを記述したりしている部分である.さらに,BGM の選択に必要な要素である感情情報を書くなどもしている.これらをひとまとめにした理由としては,「場所の移動」がシーンの切れ目に大きく影響しているからである.また,シーンが切り替わればBGM 選択のための情報が再び必要と判断したからである.今回の場合,冒頭では新しいシーンとして Map2 を背景に表示している.そしてその場面における音楽選択に必要な感情情報を音楽班側に示している.下部分は z 以上のシーンが終わったこと,その後の場面における音楽選択に必要な情報を音楽班側に示している.

7 はそのイベントが終了することを伝える部分です.書いてある場所としては,イベントの最後のみである.この部分を読み込むことで,現在のイベントを終了する.つまり,現在のイベントのテキスト内で,1 で記述されていた現在のイベント起動に必要なフラグが消去された（false になった）としても,この部分を読み込まない限りイベントは終了しない.これは,end とのみ書き起こされる.end 以降に書き起こされる要素はない.

中間発表時点からの変更点として「e」タグの位置が 3 から 6 に変更された.「e」タグの表記内容が変更されたことによって,記述するタイミングがキャラクターの表示毎ではなくシーンの開始毎に変わったためである.

（＊文責：中村祥吾）

4.1.2.5 タグの変更背景

今回使用したタグは前期に設定したものと比べると，多くの変更が行われている．何故このような変更をしたのかと言うと，前期における開発段階では，チュートリアル的なシナリオであり，自動生成を行う部分が無かったために必要最小限のタグののみで動作確認をしていた．しかし，イベントの自動生成が行われると，マップの番号が足りない，他班の仕様変更でタグ付けが必要になった項目が増えた，可能な行動が増えたことによるフラグ数の増加などが起こった．その為，全体でタグの仕様変更を連絡し，柔軟にタグへの対応をすることで解決した．後期からの変更を含めたタグの詳細については下記に示す．

（＊文責：長野恭介）

4.1.2.6 タグの詳細

Event(フラグ名,マップ名,オブジェクト名,1|2|3)

フラグ名：このフラグが true か

cE1：発生

cE2_1,2,3：経過

cE3：結末

マップ名：このマップにいるか

Map?：?はマップ番号に対応．1~8．下記参照.

オブジェクト名：このオブジェクトを感知したか

OB?：?はオブジェクト番号に対応.

1|2|3：どの表示形式で行うか

1：立ち絵表示

2：一枚絵

3：テキストウィンドウのみ.

Map 番号

1：孤児院前(Koziin-mae)

2：孤児院中(Koziin-naka)

3：地下室(Tikasitu)

4：孤児院裏(Koziin-ura)

5：初めの街

アルトリーヨ(Arutori-yo)

6：のどかな農村

トフー(Tofuu)

7：城下町

オズ城下町(Oz)

8：ダンジョン(Dungeon)

OB 番号
OB1:ランドマーク１：町ごとに指定が異なる.
OB2:ランドマーク２：町ごとに指定が異なる.
OB3:町の家の中：家の中の特定の場所
OB4:アルトリーヨとトフーを行き来する場所：水辺付近
OB5:アルトリーヨとオズを行き来する場所：町の入り口付近
OB6:オズとトフーを行き来する場所：町の入り口付近
OB7:ダンジョンのスタート位置
OB8:ダンジョンの最深部
OB9:町の中のダンジョンに行く入り口

pMap：場所の宣言．pMap1 や p 街などのように表す．立ち絵のあるイベントの時に使用.
pT：一枚絵の宣言．pT0,pT10 などのように表す．一枚絵で表すシーンの時に使用.

c：フラグ．（）内で 0|1 またはテキストを用いて管理する.
cC；操作可能かどうか cC(0) 0,1 で管理する
cB；この後にどんな戦闘が起こるか．（）内に名前で指定することが可能.
cE：どのイベントのフラグが立ったか，もしくはおろすか．cE2(0)など．0,1 で管理する.
cM；どのマップに移動できるか cM1(0) 0,1 で管理する.
cI：どのアイテムを取得したか 名前で指定することが可能．cI(アイテム名)
cI0：どのアイテムを喪失したか 名前で指定することが可能．cI0(アイテム名)

r：キャラの指定．r1 で「1 番目のキャラクター」を指定している.
立ち絵のあるイベントの時に使用.
d：キャラの向き　左右のみで dE（East）,dW(West)で表す.
f：キャラの位置(0,1,2,3 で管理．）例 f(2)
0：左手前
1：右手前
2：左奥
3：右奥

g：カット情報
g1 はヒロイン，g2 は主人公で固定されている．テキストウィンドウの左上部に表示されるキャラクター名の指定をする

s?@：セリフ．s1@と書くと，「1 番目のキャラのセリフ」を表す．
w：座標移動：イベント後に，指定したマップに移動するために使用．ダンジョン奥地から街の依頼主など．
z：シーンの終わり．1 つのイベントの中に，複数のシーン（場面）があるとみなしている．これはその複数あるシーンそれぞれの終わり，を表している．
End：イベントの終わりを示す．

e：感情を表す．[]内に 8 種の感情を 0.0~1.0 で小数第一位までで表す．
e[1,2,3,4,5,6,7,8]
1：宗教的，荘厳，真面目
2：悲哀，暗さ，失意
3：感傷，優しさ，幻想
4：平穏，満足，鎮静
5：ユーモア，奇抜，優美
6：喜び，陽気，明るさ
7：興奮，劇的，激情
8：力強さ，たくましさ，堂々

#n：メインテキスト内とサブテキスト内に存在する．テキストの挿入場所，内容を表す．
サブテキスト内の#n と#n の間にある内容をメインテキスト内の#n の場所に挿入する．
#n には数字ごとにそれぞれ決まった種類の内容を記述するようにしている．
#0~#9：本文．
特に#3 は発生から結末へ挿入する文章が指定されている．
#10：アイテム名．依頼で要求されるアイテムの名前が入る
#11：モンスター名．イベント中に登場する敵モンスターの名前が入る．
#12：情報．敵モンスターの弱点，アイテムの在処などが入る．
#13：アイテム(報酬)．依頼を達成した時に貰えるアイテムの名前が入る．
#14：依頼人．依頼主の名前が入る．これはキャラクターの口調などに参照される．
#15：マップ(探索区域)．今回のイベントにおいて，どのマップを探索するのかを名称で指定する．
#16：マップ(開放区域)．今回のイベントを達成すると行くことが可能になるマップを名称で指定する．
#17：マップ(探索区域) 数字．探索先のマップをマップ番号で表す，主にシステム側が使用する．
#18：マップ(開放区域) 数字．行けるようになるマップをマップ番号で表す，主にシステム側が使用する．
#19：クエスト発生の Map 番号．どのマップで依頼が発生したのかを指定する，OB 番号とペアにして使うことで依頼発生場所を特定する．
#20：クエスト発生の OB 番号．どのオブジェクトで依頼が発生したのかを指定する，Map 番号とペ

アにして使うことで依頼発生場所を特定する.

（＊文責：長野恭介）

4.1.2.7 クエストの生成方法

分析方法の項で説明されたように,RPG のシナリオを分析した考察として,RPG のシナリオは複数のクエストの連続で構成され,1 つのクエストは「発生」「経過」「結末」の 3 過程からできていると考えられた.よって,本プロジェクトでは「発生」「経過」「結末」の 3 つのシナリオを作り,それらを合成することでクエストのシナリオを生成した.そして生成したクエストのシナリオをいくつもつなぎ合わせることで全体のシナリオを生成することとした.また,作業量を減らすため自動生成するシナリオ範囲を制限した.今回,物語は三幕構成を元に作ることになっており,その中でも第二幕を自動生成することとした.理由として,第二幕が最も物語においてバリエーションが多く変化をつけられること,自動生成するのが簡易であると考えられたことがあげられた.

クエストシナリオを生成するにあたって,「メインテキスト」と「サブテキスト」というものを作ることとした.「メインテキスト」は「イベントの種類毎の大まかな流れを記述したテキスト」である.イベントの大筋は書いているが,詳細部分,例えば「入手するアイテム名」や「戦う敵の名前」や「イベントが発生するマップ番号」などは書かれていない.その代わり,それぞれに対応するテキストの位置に「#」タグを記述している.そして,「サブテキスト」は「イベント毎の詳細部分を記述したテキスト」である.サブテキストに記述している詳細部分は,それぞれ「#」タグで挟まれている.このメインテキストとサブテキストを合成することで 1 つのイベントを作成した.前述のメインテキストに書かれていない詳細部分を種類毎に記述している.この 2 つのテキストを合成することで,前述の「発生」「過程」「経過」のどれか 1 つのイベントのテキストを作成する.そしてイベントのテキストを 3 つ合成することで,1 つのクエストのテキストを作成した.

ここで,中間発表時点からの最大の変更点であり,メインテキストとサブテキストの合成において重要である「#」タグについて説明する.「#」タグは「#13」などのように 0 から 20 まで数字が割り振られており,数字毎に必要な要素と対応させた.例えば「#13」は「クエストの報酬アイテム」を記述するようにした.つまり,メインテキストで記述されていない詳細部分はサブテキストから抜き出され,「#」タグによって適切な位置に挿入される.1 つのメインテキストに複数のサブテキストを用意することで詳細部分を変化させ,「発生」「経過」「結末」のシナリオをいくつも組み合わせることで約 1800 個のイベントを生成することができるようになった.

（＊文責：中村祥吾）

4.1.2.8 結果と課題

最終的なフォーマットとしては,前年度のものを踏襲しつつ,前年度にはなかった要素を新たに含めるように手を加えた.また,細かな管理が必要になったため,より詳細に要素を記述できるようにも手を加えた.新たに加えた要素としては,オブジェクトの番号,シーンの切れ目,感情の情報,フラグ管理,テキストの挿入箇所である.今後とも,タグの種類は増減していくと思われる.また,データフォーマットについてはか

なり複雑になってしまい,理解が困難になってしまった.よりシンプルかつ柔軟にテキストを記述できるようにするためにも,シナリオの違和感をなくすためにも,データフォーマットとタグの改良は必要である.改良をするためには,シナリオを記述する前に分類やタグの定義をより厳密に話し合ったり,何度かシナリオを書き,必要だと思う要素を洗い出したりすることが必要だと思われる.また,まだ正確に決まっていないタグ要素や非効率的な点もあるため,その部分を改善していく必要がある.

（＊文責：中村祥吾）

4.1.3 テキスト作成

4.1.3.1 中間発表までに使われた脚本執筆

今回の脚本を担当した私として一番の難関は，問題は物語の登場人物の少なさにあった．本プロジェクトでは開発機関の短さを考慮し，「RPG として最低限成り立つ素材だけでゲームを作る」方針だった．よって人型キャラクターモデルは二体，エネミーモデル二体のみであり，マップも大きく分けて 3 つほどであった．プロローグを回す分には申し分ないマップ量であるが，問題はキャラクターモデルの数である．

これが RPG でない場合なら大した問題ではない．例を挙げるなら「夜廻り」（制作：日本一ソフトウェア）というゲームがある．公式ホームページを参照すると，ジャンルは「夜道探索アクション」となっている．「いなくなった姉と飼い犬を探しに，少女が夜の街を探索する」というストーリーの元，主人公の少女は夜の街風マップを探索する．探索する中で夜の街に住むお化けのような存在に襲われ，逃げながら物語を進めていくというホラーテイストの作品でもある．人型のキャラクターモデルは主な登場人物である主人公の少女，その姉しかいない．一見同じ条件のようにみえるが，実はまったく別物である．「夜廻り」の場合はホラーテイストでありつつ，その不思議で怪しげな夜の街を探索し，解き明かしていくゲーム性故に「世界観の説明」や「目的の明確化」「次の目的地への誘導」などが極力行われない．情報を開示しないことでプレイヤーの考察や自己解釈を深めていくことが想定されて作られている．しかし RPG の場合はそうはいかない．「旅に出る理由」に加え「世界観の説明」や「目的の明確化」「次の目的地への誘導」など，前者とは対照的に積極的に行われる．主人公にキャラクターモデルを一つ使ってしまうと，それらの役目をもう一人のキャラクターモデルに担ってもらうしかない．しかし「体験版のシナリオの想定プレイ時間は 30 分」ということが会議で決まっていた．さらに初期案には「恋愛要素をメインプロットと同等なサブプロットとして両立したい」と話に出ていたので，希望の時間内にキャラクター同士の親密性を高めていくことは難しいと考えて「幼なじみ」というある程度親密度の高い間柄を設定として取り込んだ．これによってある程度の強引な誘導が可能になると考えた．

素材も少なく，想定プレイ時間も少なかった為，物語や世界観に奥行きを持たせることは難しいと考え，最低限「話」として筋が通る程度のクオリティを目指して執筆を開始した．物語を構想するだけならそれほど難しい作業ではないのだが，「考えた構想がゲームで表現できるか？」という点を考慮すると困難を極めた．「演出的に不可能だ」とシステム班やデザイン班の要望を聞き入れながらシナリオを構築していった．

一番始めに考えたのは「三幕構成におけるどの部分を自動生成するか」という点だ．三幕構成というのは，脚本の構成術の一つであり，物語は三つの幕に分かれており，三つそれぞれの幕は設定（Set-up），対立（Confrontation），解決（Resolution）の役割を持つという物語モデルである．これは主に映画の脚本に適応されることが多いが，小説やコミックス，そしてゲームにも適応される．

主に設定（Set-up）では「主人公は誰で，何をする物語で，どのような状態であるか」を提示する一幕にする必要がある．この一幕は物語の基盤になるので，ここを自動生成することも挑戦としては非常にやりがいのある物になるとは思うのだが，ここが遅くなってしまうと，先述した通りデザイン班の町のモデリングや音響班の BGM 選出などに影響が不安視され，さらに物語構築で最重要になるセントラルクエスチョンの提示をしなければいけないので対称から外れた．セントラルクエスチョンとは，物語中に主人公が解決しなければいけない問題のことである．物語が終わるときに，この問いに Yes/No で答えられるのが物語の前提条件となっている．ドラゴンクエストシリーズに準えていうなら，「勇者は魔王を倒し，世界を救えるか？」というのがセントラルクエスチョンに該当し，ドラゴンクエストシリーズでは Yes といえるエンディングを迎えている．これは決して Yes で答えなければいけないということはない．

第三幕にあたる解決(Resolution)は，物語のクライマックスを担い，セントラルクエスチョンのへの回答はもちろん，同時並行で進むサブプロットの解決など役割が多く，一，二幕を経たキャラクター達の変化の証明をしなければいけない．ストーリーに解決をもたらす三幕も自動生成するにはあまり向いていないと考えた．それを行う為には一幕，二幕での出来事を意味合いを含めて理解していなければ作れないからである．

よってほぼ消去法的に第二幕の部分を自動生成することに決まった．けれど第二幕は三幕構成の中ではかなり自動生成に適しているともいえる．二幕は三幕中で一番長くなることが多く，前半と後半に分けて考えられる．前半では基本的に物事が順調に進み，障害を乗り越えていく．物語のちょうど中間に位置するミッドポイントを経て後半へ進むが，進むにつれて直面する障害は大きくなっていく．わかりやすくゲームシナリオで例えるなら，「ポケットモンスターシリーズ」（制作：ゲームフリーク）の「チャンピオンになるためにポケモンリーグの出場資格である 8 つジムバッチを 1 つずつ入手していく」過程がそれに該当する．この二幕ではメインストーリーの進行に含め，サブプロットも進行しなければいけないのでかなり多くの出来事が起こる．サブプロットとはメインストーリーとは違ったストーリーラインの話であり，同じように「ポケットモンスターシリーズ」で例えると「悪の組織との対決とその決着」がサブプロットとして語られている．多くの出来事を一人で構想するのは難しく，時間も多く消費してしまうので，ここを自動生成することが一番いいということが話し合いで決定した．

よって中間発表までの一幕の部分，ゲームのプロローグを執筆することになった．二幕の部分を自動生成するため，それに繋げやすい形で終わらせ，さらに想定プレイ時間が短いことも考えて，簡単にストーリーラインに乗せられる導入が必要だった．そこで「主人公の生活圏が襲われ，そこから巻き込まれていく」導入を採用した．さらに主人公たちを「孤児院育ち」という設定にすることで，「親を亡くしていることがそれほど珍しくない，村や町が襲われることのある世界観なんだ」とプレイヤーに伝える役割もあった．なるべく手短にＲＰＧの主軸となる「マップの探索」と「モンスターなどとの戦闘」を取り入

れるためには効率のいい方法だった．元々プロローグ時点で使用できるフィールドマップは一つ，ダンジョンが一つだったので「孤児院が襲われる」というイベントを起こすことによって一度に解決できた．さらに「孤児院に住む人達が攫われた」ことで，本来いなければ不自然な人のキャラクターモデルを準備できない欠点を補った．それらを済ませた後は冒険する方向へ向かわせなければいけないので，「孤児院の院長が，攫われたまま」というわかりやすい旅路の目的を提示した．大凡，ここまでがプロローグとして書き起こしたシナリオ執筆の過程だ．

（＊文責：石川一稀）

4.1.3.2 最終発表に使われた脚本執筆

中間発表を経て，後期はエピローグ，三幕構成で言うところの第三幕に該当する部分の執筆に着手した．第二幕へ上手く橋渡しをすることや，強引に簡略化したプロローグの設定たちに説得力を持たせるためのシーンを考えつつ，極力既存のマップを使い回せて，新しいマップモデルを使用しないシナリオを作成しなければいけなかった．モデルが用意できない事態になった時に，シナリオを書き直している時間はないと考えたからだ．しかし前期の段階で完成していたのは「孤児院」と「その地下空間」だけだった．新しく 3 つ町のマップを作るとのことだったが，それは大凡第二幕の自動生成部分に割くことを予定していたので，最終対決の舞台としてはインパクトに欠けてしまう．かといって新たに最終対決用のマップの作成してもらうのはデザイン班への負担的に不可能に近かった．キャラクターモデルのことも同じであり，ダンジョンで遭遇するエネミーキャラクターのモデルは複数用意できていたのだが，それをシナリオの最後に戦う相手としては役不足感を否めない．それらの点を考慮しながらシナリオを作成するのは困難を極めた．

そこでまず画策したのが第一幕から第二幕，第二幕から第三幕への橋渡しに関する部分である．第二幕を自動生成する関係上，執筆する私自身が第二幕での詳細な出来事を知らないというシナリオ作成において前代未聞の欠点が存在する．第二幕は対立（Confrontation)とされ，主人公たちに試練や成長するシーンが与えられる重要なシーンであり，これの解決編として第三幕が存在する．第二幕に話の本筋を仕込めない以上，第一幕の最後と第三幕の最初でそこをカバーして行くことが求められた．今作はそこを「戦闘に関して不慣れな主人公のための修行」という名目で大量のクエスト部分（ここが第二幕）の導入とした．しかしこれはかなり説得力に欠ける手法でもある．なので第二幕終了後に「クエストをこなさせた本当の理由」を明かすことで，シナリオの仕掛けとして落とし込めた．

物語の締めとして，主人公またはそれに近い人物が成長・変化していなければいけない．「何も変わらない」というのは，物語に意味がないことになってしまうからだ．この物語を通して，主人公たるディアリオは身体ともに成長しているが，それは顕著ではない．一番変化があったのは，ヒロインのトーシャのほうだった．物語冒頭で「進路が決まってない」「自分で物事を決められない」といったような彼女が，物語終了時には進路を自分で決める．ただ敵を倒して終わりではなく，ささやかだが未来に希望のあるエンドに落とし込みたかったので，なんとか及第点を出せた．

（＊文責：石川一稀）

4.1.3.3 自動生成用の自然言語のテキスト

クエストシナリオを自動生成するために元となるシナリオデータが必要であった．シナリオデータは分析方法の項で述べた分類に合わせて複数用意することで，どのようにシナリオテキスト同士を組み合わせても矛盾が起きないようにした．シナリオテキストはメインテキストとサブテキストに分かれている．メインテキストではクエストでどのような人物が出てくるか，その先どのようなクエストが起こるかなどの自動生成するテキストの大まか流れを記述している．サブテキストではクエストでどのような敵が出てくるか，報酬としてどのようなアイテムがもらえるか，そのクエストの続きはどこで発生するかなどのクエストの詳細な内容を具体的に記述した．メインテキストは 8 種類，それに合わせたサブテキストが 350 個程度あり，1800 個程度のクエストを自動生成することが可能である．サブテキストに関してはなるべく多くのシナリオクエストのパターンを自動生成するために，似たようなクエストの内容でも口調を変える，報酬としてもらえるアイテムを変える，ボスとして出てくる敵を変えるなどの工夫を行った．口調のパターンは男の子・女の子・丁寧な口調のおじさん・荒い口調のおじさんの 4 種類を用意した．口調を用意したことで，女の子であれば花・ぬいぐるみなどの入手を頼まれる，おじさんなら本・杖などの入手を頼まれるなどその人物に応じて違和感のない依頼等を用意することが必要となった．

（＊文責：宇田朗子）

4.2 システム班

システム班は，中間発表までの開発での改善点を見直し，成果発表までの開発では主に「クエスト自動生成システム」「戦闘」「UI」「マップ」についてシステムの開発・改修を行ってきた．特に，戦闘班は難易度調整，UI 班はマップ上でのメニューシステム，マップではダンジョン生成について新たな要素の追加を行い，システムの改良を行ってきた．以下では，成果発表までの開発について新たに行った仕様やその成果などを記述していく．なお，中間発表までの開発で作成した「シナリオ読み取りシステム」「音楽」については説明を省き，主な変更点について記述していくものとする．

（＊文責：宍戸建元）

4.2.1 クエスト自動生成システム

今回実装したクエスト自動生成システムは，シナリオ班が分析を行ったシナリオデータと，遷移確率行列を用いてクエストの自動生成を行う．クエストの作成方法は前述のデータフォーマットに記載してあるのでここでは割愛する．クエストの整合性を保つために，マルコフ性を仮定し，マルコフ連鎖を用いてクエストの選定を行う．そのため，遷移確率行列から確率モデルを作成する．また，簡単なものだが，シナリオ班から受け取るシナリオデータに不備があった場合，修正を行うシステムや作成したクエストのチェックするシステムも存在する．これらは，情報共有が円滑に行えていない場合や，システムの仕様の周知が及ばなかった際に役にたつ．

（＊文責：稲垣武）

4.2.1.1 クエスト自動生成システムにおけるクラスの定義

ファイル名：QuestWriter.py

機能：クエストの作成を行う

　　作成されたクエストは音響班へ

メソッド名：roder

機能：テキストファイルの中身を１行ずつ改行付きで配列に挿入して返す

引数：テキストファイルの名前

返り値：文字列の配列

メソッド名：get_main

機能：メインテキストの情報からサブテキストを選択するための情報を抽出する

引数：メインテキストの情報

返り値：サブテキストを選択するための情報

メソッド名：typer

機能：メインテキストの情報からサブテキストを選択するための情報を抽出する

引数：　メインテキストの情報

返り値：サブテキストを選択するための情報

クラス名：make_model

機能：確率モデルの作成を行う

メソッド名：get_Q

機能：エクセルから遷移確率行列の値をリストで取得

引数：討伐等に振り分けられた値

返り値：遷移確率行列の値をリストにしたもの

メソッド名：get_S

機能：エクセルから遷移確率行列の値をリストで取得

引数：討伐等に振り分けられた値

返り値：遷移確率行列の値をリストにしたもの

メソッド名：change

機能：遷移確率行列の値を大きさ順にソートを行い，行ったソートの手順を保存する

引数：遷移確率行列のリスト
返り値：ソートを行なった後のリスト

メソッド名：get_model
機能：遷移確率行列のリストから確率モデルを作成し，次の討伐等の数字を選出する
引数： 繊維確率行列のリスト
返り値：討伐等の数値

メソッド名：get_type
機能：数字の数に応じて，討伐等を返す
引数：get_type で得られた数値
返り値：討伐等の文字列

メソッド名：IDreader
機能：選択されたクエストの ID を読み取る，デバッグ用
引数：クエストの名前
返り値：ID

メソッド名：get_text
機能：選ばれた討伐等のサブテキストを選出する
引数：討伐等の文字列
返り値：引数で受け取った文字列を含むサブテキストのリスト

メソッド名：get_dif_sub
機能：フラグリストの中の登場人物から口調データを決定し，その登場人物が含まれているサブテキストを抽出する
引数：サブテキストのリストとフラグのリスト
返り値：抽出されたサブテキストのリスト

メソッド名：select_sub
機能：クエストの確定遷移を行うための準備
引数：発生等のタイプと選ばれたサブテキスト
返り値：確定遷移に必要な情報

メソッド名：get_data
機能：使用するサブテキストの選択．この時に確定遷移があれば，確定遷移をおこなう

引数：確定遷移に必要な情報と抽出されたサブテキストのリスト
返り値：選ばれたサブテキスト

メソッド名：main_choice
機能：メインテキストの選択
引数：繊維確率行列で選択された討伐等の情報とメインテキストが含まれているリスト
返り値：メインテキストの名前

メソッド名：choice
機能：発生，経過，結末の中から 1 つを選びそれが含まれている配列の要素を新しい配列に追加
引数：発生，経過，結末の情報と，プロットからとったクエストリスト
返り値：引数の情報を持つクエストのリスト

メソッド名：find_tag
機能：サブテキストのリストからメインテキストへ挿入するテキストを抽出する
引数：サブテキストのリスト
返り値：挿入するテキストの文字列

メソッド名：find_flag
機能：サブテキストの中からメインテキストの中に挿入するタグの抽出を行う
引数：サブテキストの中の文字列のリストと，メインテキストに挿入するテキストと発生等の情報
返り値：メインテキストに挿入するタグのリスト

メソッド名：insert_list
機能：抽出されたタグをメインテキストに挿入する
引数：メインテキストの文字列のリスト
返り値：置き換えが行われたメインテキストのリスト

メソッド名：get_number
機能：討伐等の情報に対応した数字を返す
引数：討伐等の情報
返り値：討伐等の情報に対応した数字

メソッド名：make_quest_text
機能：完成されたメインテキストのリストを文字列にする
引数：メインテキストに書き込む文字列のリスト

返り値：メインテキストの文字列

メソッド名：write_quest
機能：メインテキストの文字列をテキストファイルに書き込む
引数：メインテキストの文字列
返り値：なし

メソッド名：do_main
機能：メイン関数を動かすためのメソッド
引数：制御用の数値
返り値：なし

メソッド名：main
機能：ほぼ全てのメソッドを動かし，クエストを作成する
引数：発生等の情報と制御用の数値と確率モデルが導き出した文字列
返り値：なし

ファイル名:tag_changer.py
機能:シナリオ班から受け取ったデータの中でタグに不備の修正や，タグの値を一括で変更したい場合に用いる

メソッド名：roder
機能：テキストデータの内容を 1 行ずつ配列に挿入する
引数：テキストデータの名前
返り値：テキストデータのリスト

メソッド名：get_path
機能：テキストデータのリストからそれぞれのパスを取得する
引数：roder により得られたリスト
返り値：パスのリスト

メソッド名：changer
機能：タグの値を変更すし，テキストデータを書き換える
引数：パスのリストと変換テーブル
返り値 なし

ファイル名:ID_changer.py
機能:シナリオ班から受け取ったデータの中で，口調データを参照する際の準備に用いる
メソッド名：roder
機能：テキストデータの内容を１行ずつ配列に挿入する
引数： テキストデータの名前
返り値：テキストデータのリスト

メソッド名：ID_Change
機能：口調データに応じて ID を書き換える
引数：roder により得られたリスト
返り値：なし

（＊文責：稲垣武）

4.2.1.2 クエスト自動生成システムの詳細

クエスト自動生成システムは，ゲームを実行する前に処理を開始する．クエストの作成数は任意の数を入力し，作成を行う．

このとき，シナリオデータに不備があった場合，システムはエラーを検出してしまうので，修正システムを使用します．修正システムは，空のタグ等のエラーの原因を削除する．

遷移確率行列から作成した確率モデルをもとに，クエストの発生段階のメインテキストファイルを，プロットファイルの中から探し出して決定する．決定したメインテキストのサブテキストの探索に必要な情報を抽出し，サブテキストを決定する．この時メインテキストとサブテキストは，複数存在する可能性があるため，複数存在する場合は，ランダム関数を用いて，１つに決定する．選択されたサブテキストからタグの抽出を行い，メインテキストのタグの場所に代入を行い，クエストの発生段階を作成する．次に，経過段階，結末段階のクエストを作成する．基本的には，経過段階，結末段階のメインテキストはランダムに選択される．しかし，発生段階で経過段階，結末段階への遷移が確定する場合がある．確定している場合には，経過段階，結末段階のメインテキストはランダム関数を用いた選択ではなく，一意に定まるように決定する．メインテキストが決定すれば，発生段階と同じように処理を行う．これがクエスト１つを作成する流れとなっている． ２つ以上のクエストの作成を行う場合，クエストの整合性を保つために，確率モデルを用いて，現在作成したクエストの発生段階のメインテキスト情報から，次のクエストの発生段階のメインテキストを決定する．メインテキストが決定すると，クエストの作成が行われる．

したがって，任意の数のクエストを，クエストの整合性を保ちつつ作成可能となっている．作成されたクエスト群をプロローグとエピローグの間に挿入することによって，シナリオの作

成が行われる．

作成されたシナリオに不備があると，シナリオ読み取りシステムでエラーが検出され，正しく物語が再生されない可能性があるので，チェックシステムを使用する．チェックシステムは，無駄な空白や，文字の重複を削除し，シナリオ読み取りシステムでエラーが検出されないようにするシステムになっている．

（＊文責：稲垣武）

4.2.2 戦闘

戦闘は中間発表までの開発では，戦闘の一連の流れとキャラクターのステータス・管理の部分について大枠を作成し，その中でRPGとしてゲームの流れが実行できるよう開発を行ってきた．成果発表までの開発は，その大枠を基に細部の開発や追加機能の実装を行っていく部分がメインであり，主に「プレイヤーステータスの変更」「戦闘実行の変更」「アイテムの変更」「魔法・特技の実装」「難易度調整」の5つの部分を担当した．以下では，それらについてどのような追加や変更がなされたかを説明していく．なお，文中には中間発表までの開発で使用したクラス・関数・変数の名前が出てくることがあり，加えて変数のゲッター関数やセッター関数などの基本的な関数や，関数内の一部の仕様については説明を省くものとし，実際のプログラムを確認することを推奨するものとする．

（＊文責：宍戸建元）

4.2.2.1 プレイヤーステータスの変更

成果発表までの開発では，プレイヤーのステータスに関するbaseStatusについていくつか変更点が存在する．中間発表までの開発ではHP・MPなどの合計8つの変数でキャラクターのステータスが定義されていた．成果発表までの開発では，属性(type)，弱点(weak)，int型のListで使用できる魔法を定義するリスト(avaMagic)が追加され，11個の変数でキャラクターステータスが定義される．Typeはキャラクターの属性である．属性には4種類あり，0が無，1が炎，2が風，3が水となっている．無属性以外のキャラクターは，それぞれの属性に対応した弱点を持ち，属性が炎なら弱点は水，風なら弱点は炎，水なら風と3すくみの構造となっている．また，avaMagicは利用できる魔法をint型で定義されており，それぞれの数値は後述するmagicList内の魔法・特技の番号を表している．

また，RPGにはプレイヤーキャラクターのみが使用する要素がいくつか存在する．今回は，主にレベルの上昇のための経験値に関係する値を記録しておくため，playerStatusクラスを作成した．playerStatusクラスは，baseStatusクラスを継承したクラスであり，baseStatusクラスで定義されている12のパラメータに加えて，int型のレベル上昇に必要な経験値の総数(mustExp)，レベル上昇に必要な経験値の残量(restExp)，獲得した経験値の総数(totalExp)，boolean型のキャラクターが戦闘可能か不可能かを記憶する真偽値(alive)の4つのパラメータが定義されている．

playerStatusクラスには，レベルアップのための関数が2つ定義されている．AddExp関数は，引数にint型の経験値量をとり，経験値を加算することができる．AddExp関数はレベルアップをするかどうかをbool型で返し，LevelUp関数は，経験値が一定量を超えるとAddExp関数から呼び出される．LevelUp

関数はレベルアップ分までに必要な経験値の超過量を引数に取り，レベルの加算と次のレベル上昇に必要な経験値の残量，レベルに合わせたステータスを再計算する．この際，再度超過量を AddExp 関数の引数として呼び出すことで，AddExp 関数と LevelUp 関数を交互に呼び出しあいながら経験値の加算とレベルアップを実装を行った．

また，プロジェクト全体の方針として RPG 内に登場しプレイヤーが操作できるキャラクターについて，シナリオ上の都合から 2 人，戦闘中の敵のキャラクターが 1 人になった．そのため，難易度調整の項で後述するが，プレイヤーキャラクターに特性を持たせるため，playerStatus 内に CalcStatus 関数を用意し，ステータスの値を level に合わせて変更を行うよう実装を行った．この playerStatus を用い，プレイヤーキャラクターの内部ステータスやレベルアップを実装している．

（＊文責：宍戸建元）

4.2.2.2 戦闘実行の変更

成果発表までの開発では，中間発表までに開発した戦闘について変更点が存在する．

まず，StartBattleSet 関数についていくつか変更点が存在する．中間発表までの開発では Battle 関数には編成 ID と背景の ID を引数としていた．しかし，開発の中で敵が 1 体までに固定にする仕様となったため，敵の編成 ID の代わりに string 型の敵の名前を受け取るように変更された．

それに伴い，BattleInterface クラスの Battle 関数にも変更が加えられた．Battle 関数も，内部で StartBattleSet 関数が呼び出されるよう実装されているため，引数が敵の名前と背景 ID に変更された．加えて，ランダムエンカウントの実装のため通常のバトルの呼び出しの関数と，ランダムエンカウントの場合の関数と二種類を作成する必要があった．そこで BattleInterface クラスに Battle 関数とは別に RandomEncount 関数を作成した．RandomEncount 関数は，引数に string 型の背景 id のみをとり，戦闘に必要な敵の名前については，仕様として設定された 3 種類の敵キャラクターの名前がランダムに決定されたのち，StartBattleSet 関数がその名前と引数の背景 ID で呼び出されるよう実装を行った．また，イベント上での戦闘であるボスキャラクターとの戦闘では，BGM を変更する場合や敵のステータス生成をランダムエンカウントによる戦闘と処理を変化させる必要がある．そのため，BattleInterface クラスに bool 型の isRandomBattle という真偽値を作成し，これが false ならイベント戦闘，true ならランダムバトルとして扱うようにし，他のクラスからもこの真偽値を参照して処理を変化させるように実装した．

次に，戦闘の順序が変更された．中間発表までの戦闘ではすべてのキャラクターの行動が決定してから速度順に行動し，パラメータに反映させる仕様であった．成果発表までの開発では ，まず戦闘の開始時に行動順番を決定してから，一人ずつ行動を選びパラメータに反映させていく仕様に変更された．また，行動順の決定方法も変更されている．行動順は，まず味方側，敵側の中でそれぞれ速度順に並べ替え string 型の List である playereOrder，enemyOrder にキャラクターの名前を追加していく．その後，それぞれの List のうち未行動で最も早いキャラクター同士の speed の数値を，乱数(0.8~1.0)を掛け合わせて比べる．最終的に早い方のキャラクターが，行動選択，パラメータ反映を行い，行動を終了する．行動が終了したキャラクターは List から削除され，再度未行動のキャラクターがいる場合は速度比較，行動選択，パラメータ反映を行い，最終的にどちらの List からも要素がなくなった際に，再度すべてのプレイ

ヤーを並び替え，再度戦闘が終わるまで続けていく．

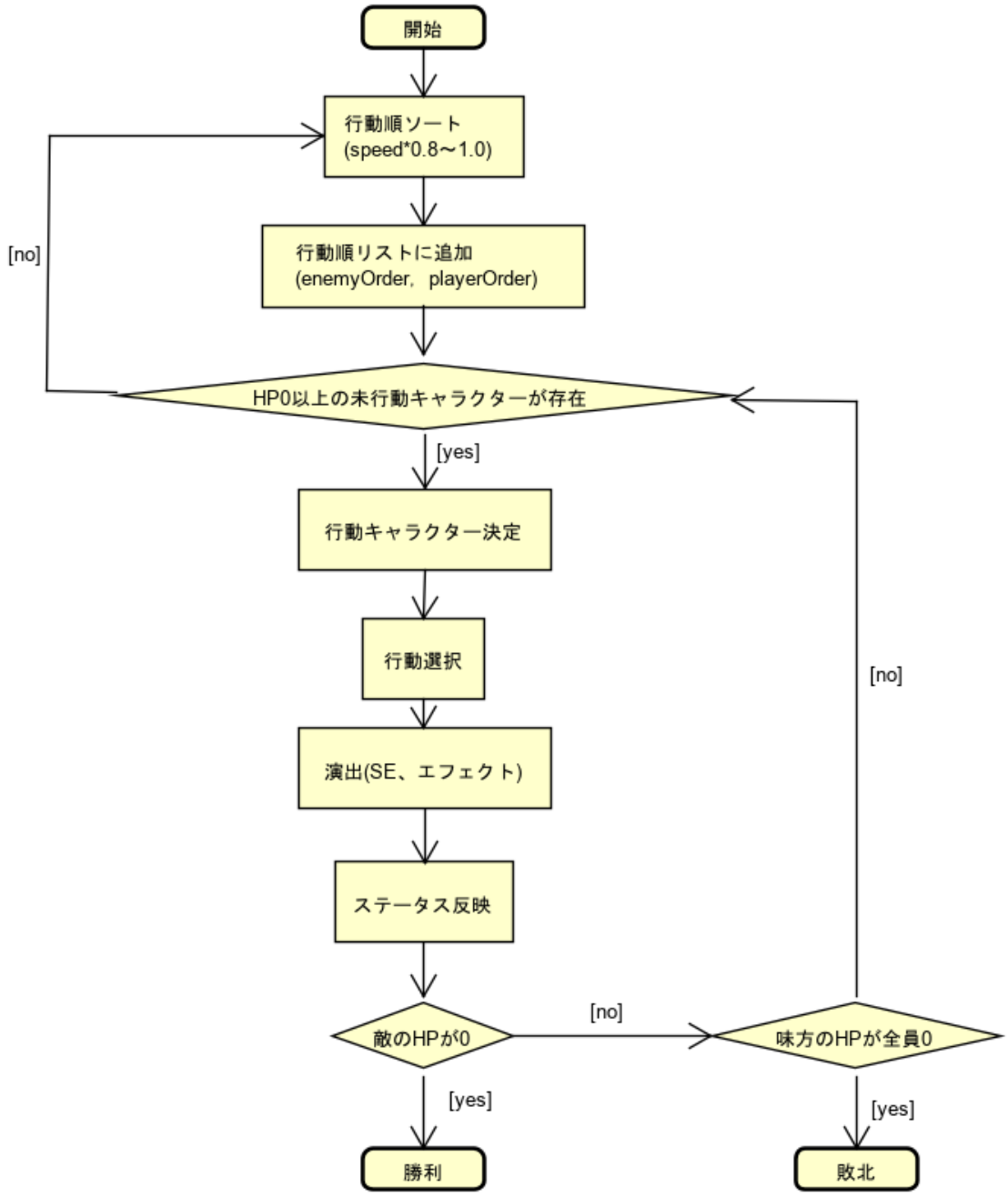


図 4.2.2.2-1 メニュー画面

更に，プレイヤーの行動決定についても変更がなされた．中間発表までの開発は，command 配列にすべてのキャラクターの行動を記録し，command 配列の数値ごとにパラメータ反映を行っていた．成果発表までの開発では，キャラクターごとに行動選択，パラメータ反映となったため，command は int 型の

変数に変更された．また，アイテムや魔法の実装に伴い，どの魔法・アイテムを使用するのか，行動の対象が誰かを決定する必要があった．そこで，どの魔法・アイテムを使用するかを記録する use_object と行動の対象を記録する target_num という int 型の変数が用意された．use_object は itemList，magicList などの List のインスタンスの番号を表し，target_num は行動の対象であるキャラクターの playerList 上での番号に相当する．プレイヤーが行動する際，プレイヤーの行動が command に記録される．その行動が魔法・特技や道具の使用だった場合には，該当の List の中身を選択させ，その List の番号を use_object に記録する．さらに，use_object の指しているインスタンスについて，単体効果であれば使用する対象をプレイヤーに選ばせ，target_num に記録する．最終的にステータス反映の際には，この 3 つの変数を参照してそれぞれのキャラクターの行動を定義した．またこれらの選択肢の分岐についても battleController 内 SelectMenue 関数，commandProcess 関数の内部にも改善を加えた．

加えて，中間発表では実装されていなかった敗北処理，リスタート処理の実装を行った.playerList のすべてのキャラクターの HP が 0 になった時，ゲームオーバーとなり，画面が暗くなって戦闘は終了する．プレイヤーはゲームオーバーになった際，バトル直前のステータスの状態で再度戦闘を行うか，HP・MP を最大まで回復して再度戦闘を行うか，ゲームを終了させるかの 3 種類の選択肢を用意した．バトル直前のステータスからの再戦では，chracterStatus クラスに定義された RebornPlayer 関数を呼び出し，変数として保存されている戦闘直前の HP・MP の数値を変更し，再度戦闘を始める．同様，HP・MP を最大まで回復して再度戦闘を行う際には，chracterStatus クラスに定義された FullRebornPlayer 関数を呼び出し，HP・MP を maxHp・maxMp の値に変更し，再度戦闘を始める．ゲームを終了させることを選んだ場合は ProcessEndBattle 関数を呼び出し，戦闘を終了させる．

（＊文責：宍戸建元）

4.2.2.3 アイテムの変更

中間発表までの開発では，アイテムの実装について item クラスを作成し，その中で item に必要な要素を定義し，マップ上から関数を呼び出すことで使用可能になっていた．成果発表までの開発では，item クラスについて，必要な情報を追加し，戦闘から利用できるよう幾らか変更を加えた．

まず，item クラス内に変数を追加した．中間発表までの開発では，string 型の名前，int 型の効果量，enum で定義された itemType 型のうち，どのタイプなのかを定義した type の 3 要素が定義されていた．成果発表までの開発では，さらに bool 型の itemRange を定義した．itemRange は ，true の場合は，アイテム効果の範囲が味方全体に及び，false の場合はアイテム効果の範囲が単体として定義した．また，int 型の item_id も用意した．この数値は chracterStatus クラスの SortItemList 関数で使用し，chracterStatus クラス内の itemList を item_id の小さい順にソートする．この 5 つの変数を利用し，item を定義した．

次に，item を使用する際に使用する関数に変更を加えた．中間発表までの開発では ，ItemEffect 関数のみを使用してキャラクターのステータスを変更していた．成果発表までの開発では，ステータス反映だけを行う関数として ItemEffect 関数，AllItemEffect 関数を作成した．baseStatus を引数に取り単体のキャラクターのステータスを変更させる場合は ItemEffect 関数を使用し，Dictionary を引数に取り Dictionary 内のすべてのキャラクターに対して変化させる場合には AllItemEffect 関数を使用した．また，

それぞれの効果を及ぼす関数について，戦闘中では効果が反映されたことをプレイヤーにテキストとして表示させる必要がある．EffectAndMessage 関数と AllEffectAndMessage 関数は，それぞれ表示させるべき文章を返す関数である．EffectAndMessage 関数は ItemEffect 関数，AllItemEffect 関数は，itemType が HPHEAL，MPHEAL の場合の AllItemEffect 関数と同じように効果を反映させ，その後文字列を返す．この文字列を表示させ効果を反映させることで戦闘中のアイテム使用行動の演出を作成した．

次に，アイテムの取得が追加された．アイテムは charactrerStatus クラス内の itemList で管理されるため，characterStatus クラス内に PickUpItem 関数を定義した．成果発表までの開発では，アイテムは名前・効果・回復量・効果範囲を変化させ，主に戦闘に使うためのアイテムを 10 種用意し，PickUpItem 関数の引数としてアイテム名をもらい，そのアイテムの item クラスのインスタンスを作成して itemList に登録されるようにした．またシナリオ上で自動的に生成されるアイテムについては，名前は引数と同じで itemType が特殊なものとして item クラスのインスタントを作成・登録を行い，マップやシナリオの進行に必要なアイテムを item クラスで実装した．今回，アイテムを取得できる条件は，戦闘終了後，宿での回復後，イベント上の 3 種類とした．シナリオ上では PickUpItem 関数が呼び出され，アイテムが追加され，戦闘上でも getLootItem 関数を用意し，ランダムにアイテムが手に入るよう実装したが，これらが一度に取得できるアイテムは 1 個のみである．一方，宿での回復後については，複数のアイテムを同時に受け取れるような仕様が求められた．そのため，characterStatus クラス内部に，宿での回復を選択した際の変更を記述した Hotel 関数を用意した．Hotel 関数は呼び出されると，プレイヤーのキャラクターすべての HP・MP を回復し，アイテムの取得も行えるようになっている．また，プレイヤーのレベルに応じて取得できるアイテムを変化させるよう調整をおこなった．

このように，成果発表までの開発では，アイテムに関係する様々な部分について調整を行った．

（＊文責：宍戸建元）

4.2.2.4 魔法・特技の実装

成果発表までの開発では，新たに魔法・特技の実装を行った．魔法・特技はキャラクターが MP を消費することで使用できる行動である．まず，magicData を定義し，魔法・特技に必要なパラメータや関数を定義した．変数に，string 型の魔法名(magicName)，int 型の MP 消費量(useMp)，効果量(magicQuantity)，使用した際の SE を指す音声配列の番号(magicSE)，bool 型の効果範囲を表す真偽値(magicRange,true なら全体効果，false なら単体効果)，enum 型で定義された magicEffect の魔法効果(effect)，magicAttribute の魔法属性(attribute)，この 7 つの変数を定義した．また，item クラスと同様に，効果を反映させる際に呼び出される関数を定義している．MagicEffect 関数と AllMagicEffect は，魔法・特技の効果を使用する関数である．MagicEffect 関数は魔法・特技使用者の baseStatus と使用対象の baseStatus を引数に取り，使用者の MP を減らし使用対象のステータスを変更させる．AllMagicEffect 関数は，使用対象として必要な引数が List になっており，List 内のすべてのキャラクターについてステータス変更を行う．また，魔法・特技を使用した際ステータスの変化を文字列として表示したい際には，item クラスと同様に MagicEffectAndMessage 関数や AllMagicEffectAndMessage 関数を使用する．

魔法・特技は，RPG では統一された世界観の中で同じ名前の魔法・特技を複数キャラクター間で共有

する場合が存在する．そこで magicController クラスを定義し，その中にすべてのキャラクターが使用できる魔法・特技の種類を定義した magicList を定義した．magicList の内部には，magicData クラスのインスタンスがあり，今回は 15 種類の魔法・特技をあらかじめ用意されてある．実際に魔法・特技を使用する場合には，baseStatus 内に定義されている avaMagic の数値が magicList 内の magicData の番号を指しているので，その magicData の関数を呼び出すことでステータスへの反映を行った．

（＊文責：宍戸建元）

4.2.2.5 難易度調整

成果発表までの開発では，プレイヤー・敵のキャラクターの難易度調整を行った．

まず，プレイヤーキャラクターについて，レベルによる調整の実装を行った．レベリングの要素は，前述したように playerStatus クラス内で，AddExp 関数や LevelUp 関数を用意し，レベルや経験値の管理を実装している．playerStatus クラスにある，CalcStatus 関数はレベルからステータスの再計算を行う関数である．関数内には，キャラクターごとの初期ステータス(status 配列)，成長テーブル(playerStatus 配列)，使用できる魔法・特技(magicOrder)が定義されており，キャラクターの level の値に成長ステータスを掛け合わせ，初期ステータスを足し合わせることで各要素を計算している．今回は，プロジェクトの方針上，プレイヤーが操作できるキャラクターが 2 人なため，主人公に該当するキャラクターが“平均タイプ”，ヒロインに該当するキャラクターを“魔法タイプ”として，内部的に成長テーブルや初期ステータス，成長時のパラメータの変化量や使用できるようになる魔法を調整している．これらのステータスについては，『ドラゴンクエスト』シリーズの中でも複数人戦闘が追加された，『ドラゴンクエスト 2』のパラメータから分析を行っている．そのため，今回の制作でのキャラクターの最高レベルは，作品内でプレイヤーキャラクターの最大レベルに指定されている 50 までに設定されている．ステータスについては，作品内のうち平均タイプ，魔法タイプに該当するようなキャラクターのレベルごとのステータスから分析し，作品内で特に成長するステータスについては同様に顕著に上がるよう，成長しにくいステータスについてはレベル差変化による量が小さくなるよう，キャラクターの成長テーブルを変更している．また，1 レベルごとの上昇値についても，レベル 20〜30 の中ごろに一番大きく変化しているという傾向が分析から見られたため，成長ステータスの特定の部分を除き，レベル 10 ごとに 1 レベル毎の変化量を変え，0〜10，40〜50 レベルの成長では変化量が小さく，20〜30 レベルの間で一番変化量が大きくなるよう変更を施した．同様に，使用できる魔法についても，魔法タイプの方が使用できる魔法・特技が多く成長が早いため，最終的に使用することのできる魔法・特技の数，新しい魔法が追加されるレベルのキャラクターごとの調整も行っている．これらの要素を変更しながら，成果発表までの開発でキャラクターの調整を行った．

次に，敵キャラクターの難易度調整として，敵ステータスの自動生成を行った．enemyList に格納される敵のステータスは，characterStatus の createEnemy 関数で行われる．createEnemy 関数は，敵の名前を引数に取り，各敵のステータスの初期値を記述した eBooks 配列と，敵ごとの level 毎の変化量を記述した eStatusBook 配列からステータスを求めていく．戦闘を開始する Battle 関数が呼び出された時，引数として受け取った name を受け取り，name に照合する名前の敵ステータスの初期値を eBooks 配列か

ら探し出す.

その後，playerList に格納されている全プレイヤーのレベルの平均を求める．その後，イベント戦なら平均レベル，ランダムエンカウントの場合は，その値から-3〜+3 の数値をランダムに選出し，敵のレベルとする．この際，ランダムエンカウントの場合には，敵のレベルが味方の平均レベル+2 以上の場合，敵の名前を『ラージ OO』とし，キャラクターグラフィクを通常のものから変更することで，視覚的に強いキャラクターだとわかりやすいようにしている．そのレベルに，eStatusBook 配列のキャラクタータイプごとの変化量を掛け合わせ，eBooks のステータス各値の初期値と足しあわせることで敵のステータスを決定する．eStatusBook 配列は，プレイヤーキャラクターの調整と同じように，『ドラゴンクエスト 2』を参考にステータスが調整されている．成果発表までの開発では，モデル班との協議の結果敵が 3 種類のタイプに分類される仕様となった．そのため，敵を「バランス型」「攻撃型」「魔法型」の 3 特性に分類し，『ドラゴンクエスト 2』から近しい特性を持つ 3 種類のモンスター，3 種類のボスモンスターのステータスを参考にし，eStatusBook，eBooks を作成した．eStatusBook で定義された変化量は，6 つあるうちの 3 つはランダムエンカウントの際に使用し, ボスキャラクターの場合は残りの 3 つを使用する．この際，敵のレベルに合わせて eStatusBook 配列の変化量も変化し，計算の際に 0〜10，40〜50 レベルの成長では変化量が 0.75 倍, 20〜30 レベルの間で一番変化量が 1.25 倍になるよう変更を施した. また，敵キャラクターが使える呪文・特技についてもそれぞれのタイプについて，キャラクター特性ごとに覚えるべき魔法，魔法を使用できるレベルを難易度を考慮して手動で調整を行った．最終的にその内容にそったステータスの baseStatus クラスが作成され，敵として enemyList に格納される．

最後に, 敵キャラクターの難易度調整として, 簡易な敵 AI の作成を行った. 中間発表までの開発では，敵は味方キャラクターのうちどちらかを選び，攻撃のみを行うだけだった．成果発表までの開発では，enemyStatusClass 内に新たに enemyAI という int 型の変数を定義した．enemyAI は敵 AI の行動パターンを表しており，先述したキャラクター特性のうち，「バランス型」の場合は 0，「攻撃型」の場合は 1，「魔法型」を 2 としている．enemyAI は createEnemy 関数が呼ばれた際に決定される．この数値は，敵キャラクターの行動を定義した battleController の EnemyAttack 関数で使用される．敵キャラクターの行動は，攻撃，魔法・特技のどちらかの行動を行う仕様となっている．そのため，EnemyAttack 関数の中では，どちらの行動を選ぶかをランダムに決めている．そこで，キャラクター特性を行動に反映させるため，EnemyAI の値によって，「バランス型」の場合はどちらの行動も半分ずつ，「攻撃型」の場合にはより通常攻撃を行いやすく，「魔法型」は魔法・特技を選びやすいよう，重みづけを行っている．
このようにして，プレイヤー・敵キャラクターステータス・敵 AI の要素によって難易度を調整し，RPG の戦闘として成立するよう成果発表までの開発を行った.

（＊文責：宍戸建元）

4.2.3 後期 UI 開発

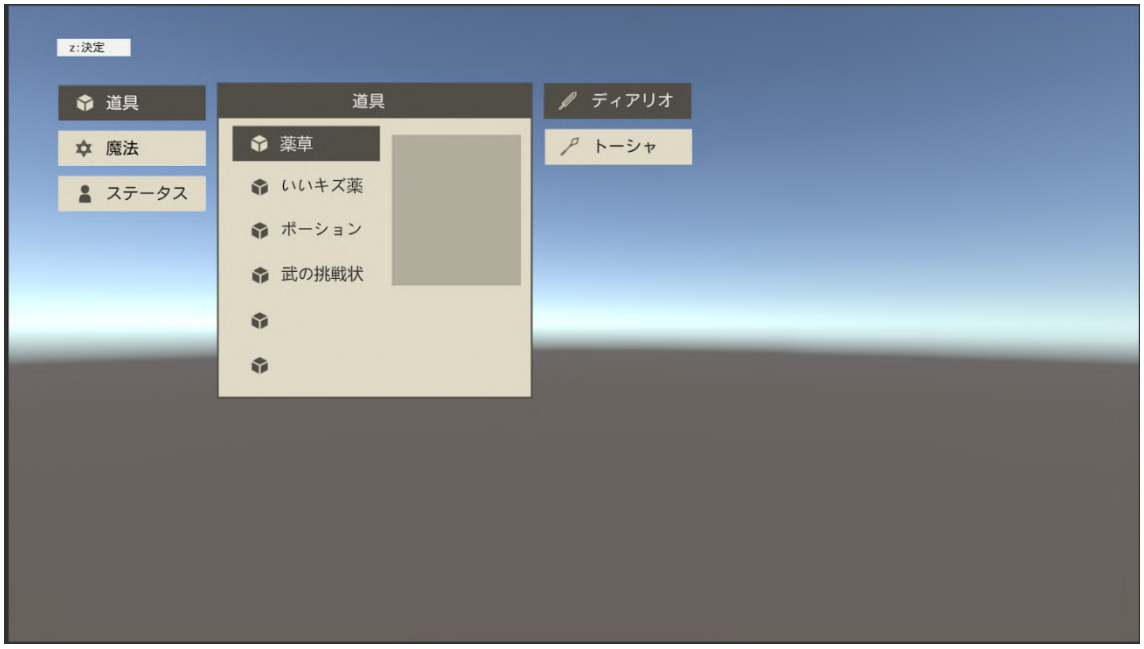
後期の目標として，メニュー画面のデザインの向上，キャラクターの攻撃力を増減させる「そうび」の実装，「どうぐ」や「そうび」の解説を行うウィンドウの作成を挙げた．この中で最初に，メニュー画面のデザインの向上から作成を行った. 視覚班にゲームの世界観を再現した UI のデザインを作成してもら

い，そのデザインを実際にゲーム上で動かせるようシステムを開発した．開発にあたっては，前期のプログラムを参考にしつつ，指定されたデザインを動かせるプログラムを新規に作成した．

（＊文責：西川和真）

4.2.3.1 新メニューシステムの開発

視覚班がデザインした画像を基に，新たな UI 画面を作成した．矢印によって操作していた前期とは異なり，後期では現在選択されている項目の色を反転させ操作をする方式をとった．この方式に変えることによって，矢印の座標変更をする際の複雑なプログラムを排すことができた．また，デザイン面においてもより無駄を省くことができたと考える．また，新しくキャラクターたちの「まほう」を表示させる項目を新規に開発した．このプログラムは，戦闘担当のメンバーが作成した「まほう」を管理するプログラムを基に作成を行った．



画像 4.2.3.1-1 メニュー画面

このシステムを開発するために，主に 3 種類の新たなプログラムを追加で作成した．

まず，現在プレイヤーが何を選んでいるかを示す Item_select（魔法の場合 Magic_select）クラスである．このプログラムは前期と同じく，入力された矢印キーの方向によって数値を増減させ，その数値を他のプログラムが参照できるものである．また，一画面に表示できる「どうぐ」または「まほう」の数が 6 つになったため，6 つ以上の「どうぐ」を持っている状態でも正常に表示ができるようプログラムの改修を行った．この機能は，後述する Item1~6_text（魔法の場合 Magic1~6_text）と連携をして行われる．

次に，選択されている項目の色を反転させる Item1~6_now（魔法の場合 Magic1~6_now）クラスである．このプログラムは，色の反転を管理する空オブジェクトにコンポーネントする．空オブジェクトには

子オブジェクトとして，項目が選択されている画像，選択されていない画像が入っている．Item_select 上で自らの項目が選択されていることを認識すると，選択されている画像を表示させ，選択されていない画像を消す．この状態で決定キーを押すと，「どうぐ」「まほう」を誰に使うかを表示させる．項目が選択されていないと，選択されていない画像を表示させ，選択されている画像を消す．この機能を作成したことで，よりプレイヤーは直感的にメニュー選択をすることができると考える．

次に，「どうぐ」や「まほう」の名前を表示させる Item1~6_text（魔法の場合 Magic1~6_text）クラスである．「どうぐ」の名前の取得の仕方は前期と同じである．後期は「まほう」の名前を表示させるために，機能を追加した．MagicController クラスに魔法の一覧があるため，このクラスを利用して名前を取得した．キャラクターたちのステータスを管理する CharactorStatusClass に，playerStatus というキャラクターのステータスの数値を管理するコンストラクタがある．そこにそのキャラクターが覚えている魔法の種類を示す配列が存在する．例えば，配列に 2 という数値があれば，そのキャラクターは MagicController の二番目の魔法を使用することができるということである．Magic1~6_text クラスは，この配列を参照し，魔法の情報を扱う MagicData クラスのゲッターを用いることで，そのキャラクターが覚えている魔法を表示させる．また，一画面に表示できる「どうぐ」または「まほう」の数が最大６つになったため，６つ以上の「どうぐ」または「まほう」を表示できるシステムを開発した．「どうぐ」または「まほう」をメニュー画面で開くと，１～6 個目の項目が表示される．６つめの項目を選択している状態で，更に下方向キーを押すと，Item1~6_text（魔法の場合 Magic1~6_text）クラスが表示する項目を切り替え，７～13 個目の項目を表示させる．この機能を開発したことで，表示スペースを省力化でき，画面の見やすさをさらに向上させることができたと考える．

（＊文責：西川和真）

4.2.3.2 ステータス画面開発

次に，キャラクターたちのステータスを表示させるステータス画面を開発した．ステータス画面のデザインは視覚班から全体像を指定してもらい，それを実際に動かせるようシステムの開発を行った．また，ステータス画面にはそのキャラクターの性格，説明などをする紹介文が表示される．この紹介文は，物語分析班がキャラクターの設定，物語の背景などを基に作った．この紹介文によって，キャラクターへの感情移入がしやすくなり，よりゲームストーリーに没頭することができるという効果があると考える．

画像 3.4.2.2-1 ステータス画面

ステータス画面に表示させるのは，キャラクターの名前，容姿，レベル，最大 HP と現在の HP，最大 MP と現在の MP，攻撃力，守備力，経験値，キャラクターの紹介文である．キャラクターのレベル，最大 HP などの数値は，キャラクターの基本的な数値を管理している baseStatus クラスから取得して表示させている．経験値は，経験値を管理する playerStatus クラスから取得している．キャラクターは二人いるため，表示させるステータスを切り替えるシステムを開発した．メニュー画面で「ステータス」を選択すると，「ディアリオ」か「トーシャ」を選ぶ画面になる．例えば，「ディアリオ」を選ぶと，Status_ade クラスが「ディアリオ」の playerStatus と baseStatus からレベル，HP などのステータスを取得する．そして，対応する座標にテキストデータを表示させる．

（＊文責：西川和真）

4.2.3.3 今後の展望

今後の展望としては，実装することができなかった，キャラクターの攻撃力を増減させる「そうび」の実装，「どうぐ」や「そうび」の解説を行うウィンドウの作成が挙げられる．また，マップ移動中でも表示される HP や MP などの重要性の高い情報をより見やすくした専用のウィンドウを開発することがあげられる．これを開発することで，プレイヤーが HP などを確認したいときに，一々メニュー画面を開くという作業を省くことができる．それによって，プレイ中の負担を減らせることができると考える．また，ステータス画面からキャラクターたちの覚えている「まほう」を表示させるシステムの作成も必要であると考える．

（＊文責：西川和真）

4.2.4 マップ

上下移動，左右移動に応じてフラグを建て，キーが押されたとき，別のキーが押されながら，別のキーが離されたとき，両方のキーが離されたときの三つの分岐を用いて，滑らかな移動を実装した．また，それ以外にもアニメーションの制御，追跡するキャラクターのための座標記録，物体に衝突したときの処理など，移動に関する処理を行う．さらに，マップ上のオブジェクトに応じて，必要な動作をさせる．

クラス名：box
メソッド名：SceneLoaded
機能：シーンをロードしたときに，値の初期化や，シーン移動前の座標を記録する．
引数：Scene,LoadSceneMode
返り値：なし

メソッド名：Move
機能：マップ移動上の全体制御を行う．
引数：なし
返り値：なし

メソッド名：MoveCheck
機能：物体に対する衝突の有無によって，プレイヤーの移動を制限する．
引数：Vector3
返り値：float

メソッド名：HitCheck
機能：物体に衝突しているか判定する．
引数：Vector3
返り値：bool

メソッド名：HitItemName
機能：衝突したオブジェクトを取得する．
引数：Vector3
返り値：GameObject

メソッド名：HitTile
機能：足元のオブジェクトを取得する．
引数：Vector3

返り値：GameObject

メソッド名：EventCheck
機能：イベントの有無を判定し，実行可能であれば実行する．
引数：Vector3
返り値：なし

クラス名：SubPlayer
メソッド名：SubMove
機能：操作キャラの座標を任意の距離を置いたうえで，追跡する．
引数：なし
返り値：なし

クラス名：Windmill
機能：マップ上の風車を回転させる．回転方向を指定可能.
引数：なし
返り値：なし

クラス名：GoNextMap
機能：一定のルールに沿うことで，移動するシーン，移動先を指定してプレイヤーを移動させることができる．
引数：なし
返り値：なし

クラス名：GoHouse
機能：マップ上の家の内外を行き来させる．
引数：なし
返り値：なし

3.2.4 ダンジョン

ダンジョンを生成するため，一定のアルゴリズムに沿って数列を作成，そのデータに応じて適切な 3D データをマップ上に配置する．また，次回呼び出しのためのデータ保存を行うなど，ダンジョンの自動生成に関する処理を行う．

クラス名：AutoDungeonCreator
メソッド名：reMake

機能：ダンジョンを再生成するためにデータを初期化する.
引数：int, int, int
返り値：なし

メソッド名：GetStart
機能：ダンジョンの始点を返す.
引数：なし
返り値：Vector3

メソッド名：GetDungeonData
機能：今までに生成したダンジョンのデータを取得する.
引数：int
返り値：int[,]

メソッド名：GetEndX
機能：今までに生成したダンジョンのデータの終点 X 座標を取得する.
引数：int
返り値：int

メソッド名：GetEndZ
機能：今までに生成したダンジョンのデータの終点 Z 座標を取得する.
引数：int
返り値：int

メソッド名：DungeonDataMaker
機能：作成，取得したデータに基づいてダンジョンを生成し，データを別クラスに保存する.
引数：なし
返り値：なし

メソッド名：DungeonDataSet
機能：今までに生成したダンジョンのデータをセットする.
引数：int[,], int, int
返り値：なし

メソッド名：DungeonMaker
機能：データに基づいて，マップ上にオブジェクトを配置する.

引数：なし
返り値：なし

メソッド名：makeBigRoom
機能：部屋ごとの中心座標を決め，部屋を生成する．
引数：int
返り値：なし

メソッド名：roomToRoad
機能：部屋のつながり方を指定する．
引数：なし
返り値：int[,]

メソッド名：makeRoad
機能：部屋同士の道を生成する．
引数：int[,]
返り値：なし

メソッド名：putRoom
機能：オブジェクトを部屋に配置するため，生成した数列に応じて文字列を生成する．
引数：なし
返り値：なし

メソッド名：putTreasure
機能：宝箱を配置する位置を指定する．
引数：int
返り値：なし

クラス名：DungeonController
メソッド名：GetData
機能：今までに生成したダンジョンデータを取得する．
引数：int
返り値：int[,]

メソッド名：startAddData
機能：今までに生成したダンジョンの始点データをセットする．

引数：int, int
返り値：なし

メソッド名：GetStartDataX
機能：今までに生成したダンジョンの始点 X データを取得する.
引数：int
返り値：int

メソッド名：GetStartDataZ
機能：今までに生成したダンジョンの始点 Z データを取得する.
引数：int
返り値：int

メソッド名：GetDataPos
機能：今までに生成したダンジョンの入場回数からどのデータを取得すればよいか返す.
引数：int
返り値：int

メソッド名：AddData
機能：生成したダンジョンの終点データを保存する.
引数：int, int
返り値：なし

メソッド名：GetEndX
機能：今までに生成したダンジョンの終点 X データを取得する.
引数：int
返り値：int

メソッド名：GetEndZ
機能：今までに生成したダンジョンの終点 Z データを取得する.
引数：int
返り値：int

（＊文責：太田和宏）

4.3 音響班

分析対象であるゲーム中のイベントシーンに感情特徴量を定義し，その時鳴っているＢＧＭの音響特徴量を抽出して，入力が感情特徴量で出力が音響特徴量であるニューラルネットワークに学習させた．そしてその学習済みのニューラルネットワークを用いた選曲システムを作成した．

（＊文責：山内拓真）

4.3.1 分析

4.3.1.1 分析の再検討

前期の活動から分析によって出たデータを使用した場合，決定木を用いた曲選択システムでは導出される曲番号に偏りが生まれた．更に，選ばれた曲もその場の雰囲気や感情にそぐわないものが多数見られた．そのため選択木を用いた曲選択システムでは今回のゲームに使用できるような結果が得られない事がわかった．この為，後期の活動開始時に物語分析班と合同で分析内容の再検討をし，再度分析を行うことにした．

（＊文責：長野恭介）

4.3.1.2 分析内容の再定義

はじめに，分析タグの再検討をした．前期の手法では，場面の状況やキャラクターの感情，そのイベントシーンの場所などから感情を読み取り，曲と対応付けを行うことで選曲しようとしていた．この手法では求めるような結果が出なかったため，前期の結果を改めて考察した．出力されたデータを見返したところ，「ファイナルファンタジー」シリーズは主人公がキャラクターとして確立しており，プレイヤーが物語を楽しみながら動かすタイプのゲームであったのに対し，もう一つの「ドラゴンクエスト」シリーズでは，主人公は物語内で語らず，プレイヤー自身が主人公であるかのように楽しめるようなシステムとなっており，主人公の行動に大きな差があった．そのため場面における感情という前期の分析項目では作品間で分析データが極端に偏る結果となり，結果として期待した結果とならなかったと考えた．そのため，後期では感情を機械音楽の観点から求め，音楽と場面の両方に注目した分析を行うことにした．

（＊文責：長野恭介）

4.3.1.3 後期における場面の分析

後期での場面分析では，前期と同様に感情を取ることにしたが，感情を取る対象をイベントシーンのみに絞った．そして感情の項目には今回は 2 つの案が出た． 1 つは hevner という心理学者が提唱した 8 つの印象語群というものである．これは，宗教的・荘厳・真面目や悲哀・暗さ・失意といった 3 つの印象を 1 つの語群として扱い，8 つの語群を用いて感情を分けるという手法である．もう一つは Russell の円環構造モデルというもので，感情を覚醒状態か非覚醒状態か，快か不快かの 2 次元でマッピングすることで分類するという手法である．はじめに私たちは Russell の円環構造モデルを使用して，感情分析を行った．だが，複数人での分析では，覚醒状態と快，不快の状態が統

一化しにくいという問題が発生した．そのため，データとして前期と似たようなあまり有用ではないデータが出来てしまった．この問題を受けて，Russell の円環構造モデルの使用を断念し明確な分類があり，数値で管理しやすいという利点もある hevner の印象語群を用いることにした．今回は 8 つの印象語群にそれぞれ 0〜1 までを 0.1 刻みで数値化してデータとして用いた．

（＊文責：長野恭介）

4.3.1.4 音響特徴量の抽出

ゲーム中のイベントシーンに対しそのシーンに合っているＢGMを付加するために，ＢGM側の情報の指標として楽曲の波形情報から得られる物理的な 3 つの音響特徴量及びそれに統計処理を行ったものを使用した．音響特徴量の抽出には Python の音楽情報処理ライブラリ librosa を用いた．対象は全てｍｐ3 形式の楽曲であり，ステレオ形式でサンプリングレートは 44100Hz である．また，librosa を用いた楽曲の読み込みを行う段階で，オーディオサンプリングレートは 22050Hz に自動的にリサンプリングされ，ステレオ形式からモノラル形式に変換されている．音響特徴量の抽出範囲は，開始直後の無音状態を考慮して楽曲の開始 1 秒の地点から 30 秒間とした．今回対象としたＢGMには，ドラゴンクエスト６，ドラゴンクエスト７，ファイナルファンタジー１，ファイナルファンタジー３，ファイナルファンタジー５の５作品に使用されているＢGM91 曲（以下市販ゲームのＢGM）と，複数のフリー音源サイトから入手したＢGM149 曲（以下フリーＢGM）を用いた．また，フリーＢGMを入手する際には特に基準を設けず，それぞれの楽曲には一意の曲番号を付与した．

３つの音響特徴量は全てスカラー値であり，１つ目はＢＰM(Beats Per Minute)を使用した．これは一分間の拍数のことであり，Librosa 内の関数 Librosa.beat.tempo を用いて抽出した．本来判明しているＢＰMに比べ値が半分になったり２倍になったりする問題が発生したが，今回は無視した．２つ目にはスペクトルセントロイドを使用し，抽出には Librosa 内の関数 Librosa.feature.spectral_centroid を用いた．３０秒間でフレーム毎に得られるスペクトルセントロイドの平均値をとり，それを特徴量とした．

３つ目はクロマグラムから得られる音名毎のパワーの分散であり，クロマグラムは Librosa の関数 Librosa.feature.chroma_stft を用いて抽出した．特徴量を作り出す手順を説明する．図 4.3.1.4-1 楽曲から得られたクロマグラムである．横軸は時間であり，縦軸は C,C#,D,D#,E,F,F#,G,G#,A,A#,B の 12 音を表している．そして色の濃さの違いは音名それぞれのパワーの強さを 0.0~１.0 の範囲で表したものである．このパワーが 30 秒間の全フレーム中で 0.5 を超えた回数を音名毎にカウントし，配列に格納した．そしてこの配列を音名を度数としたヒストグラムにしたものが表 4.3.1.4-2 であり，このヒストグラムの分散を特徴量とした．しかし，クロマは円環上になっているため C,C#,D,D#,E,F,F#,G,G#,A,A#,B の順番, C#,D,D#,E,F,F#,G,G#,A,A#,B,C の順番というように横に１音シフトして得られる 12 通り全ての分散を考慮する必要がある．よってその 12 通りのうち分散が最も小さくなるような順番のものを選択して，小数第 3 位を四捨五入して特徴量とした．

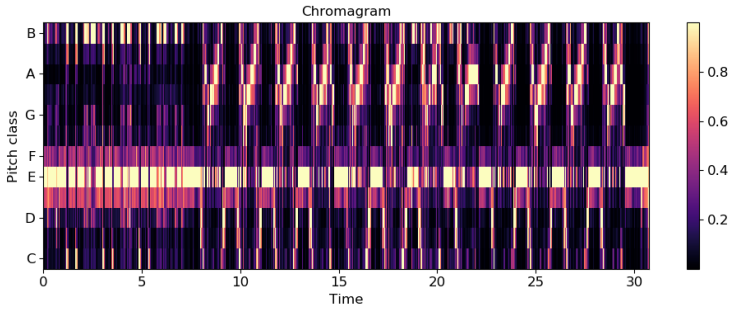
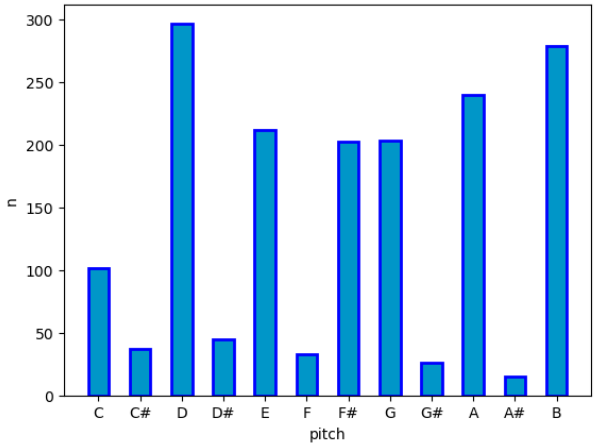


図 4.3.1.4-1　楽曲から得られたクロマグラム

表 4..3.1.4-2 クロマグラムから得られたヒストグラム



これら３つの音響特徴量と曲番号を１セットとして，市販ゲームのＢGMとフリーＢGMそれぞれにおいて CSV ファイルとして記録した.

（＊文責：山内拓真）

4.3.2 ニューラルネットワークの学習

4.3.2.1 データの調整

楽曲から得られた音響特徴量をニューラルネットワークの学習に使用する前に，python のプログラムを用いてデータの調整を行った．まずは曲ごとに得られた３つの音響特徴量を６次元の one-

hot ベクトルに変換した．One-hot ベクトルとは[0,0,0,0,1,0]のように一つのカラムだけが 1 で残りは 0 のベクトルのことである．この形式にした理由としてはニューラルネットワークの学習が容易であること，ベクトルにすることにより比較が容易になることが挙げられる．ここで，one-hot ベクトルのどこを 1 にするかという点を考える必要がある．ある数値からある数値の間に位置していた場合そこに対応するカラムを 1 にする仕組みだが，特徴量の範囲を単純に 6 分割すると，データの分布に偏りがあった場合，ある特定のカラムに 1 が集中することになる．実際に 3 つの音響特徴量それぞれにおいてヒストグラムを作成し確認してみると度数分布に偏りが見られた．それを防ぐため，変換の際，データが一様に分布するように調整した．その調整について説明する．

まずはフリーＢGM149 曲の音響特徴量のデータを昇順にソートし，今回は 6 次元のベクトルなので，先頭，16%，33%，49%，66%，82%の個数の位置に存在する値を境界に設定し，ベクトルのどこを 1 とするかを決定した．具体的にはＢＰMの場合，最小値の 99.4 から，先頭から 16%の個数即ち 24 個目の値 112.3 までに該当する場合一つ目のカラムを 1 にするといった具合である．この調整を 3 つの音響特徴量それぞれに対し同様の手順を用いて行った．

（＊文責：山内拓真）

4.3.2.2 データセット作成プログラム

ニューラルネットワークの学習に用いるデータセットを手動で作成するためには多くの時間がかかるため，データセットを作成するプログラムを python で作成した．その手順を説明する．プログラム中で用いているファイルの説明を以下に記す．

ファイルＡ：シーン分析によって定義した 8 つの感情特徴量とその時鳴っているＢGMの曲番号が記載されているファイル
ファイルＢ：市販ゲームのＢGMの曲番号と音響特徴量が記載されたファイル

プログラムの流れとしては，まず音響特徴量のあるファイルＢを一行ずつ読み込み曲番号を読み取る．そしてその曲番号と同様の曲番号をもつファイルＡの行に音響特徴量をコピーすることで，感情特徴量と音響特徴量がセットになったデータセットが完成する．学習に用いるためのデータ数は最終的に 487 個となった．

（＊文責：山内拓真）

4.3.2.3 ニューラルネットワークの実装

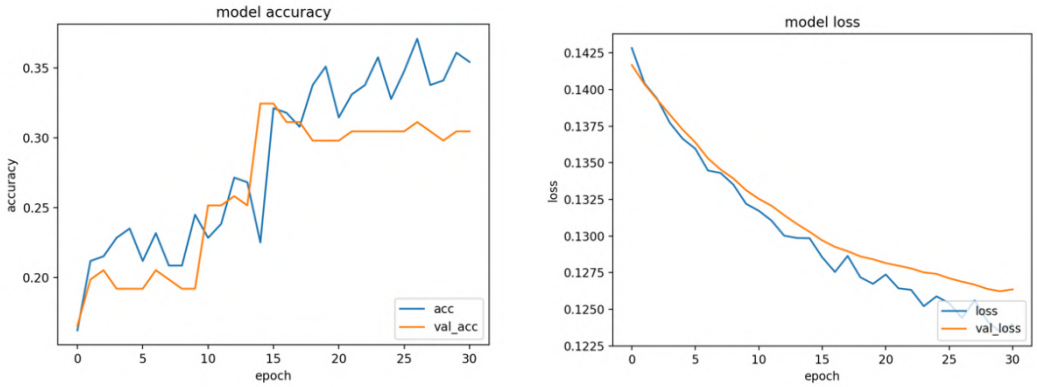
楽曲を選択するシステムを制作するにあたり,システムの一部に 3 つのニューラルネットワークを用いた.これらのニューラルネットワークは,入力層 8,中間層のユニット数が 5,出力層 6 の 3 層で構成されている中間層 1 層のもので,入力として感情特徴量を入力し,出力として音楽特徴量を出力

する.データ数は 487 である.ニューラルネットワークの学習モデル制作にあたり,ライブラリとして Keras を使用した.Keras とは，TensorFlow や Theano 上で動くニューラルネットワークライブラリの 1 つである.Keras を使用することで,ディープラーニングのベースとなっている数学的理論の部分をゼロから開発せずとも,比較的短いソースコードで実装することが可能になる.中間層の活性化関数は ReLu を用い,結果の値が高かったところに 1 を立て onehot ベクトルとした.また,出力層の活性化関数には softmax を用いた．損失関数は mean_squared_error(平均二乗誤差)を用いた.全てにおいて Epoch 数を 1,000 にし,コールバックを制作した.Keras にはデフォルトで何種類かのコールバックが用意されているが,今回は EarlyStooping というコールバックを採用した.EarlyStopping とは,学習ループに収束判定を付与することができるコールバックである.これを用いることで過学習を防ぐことができる.

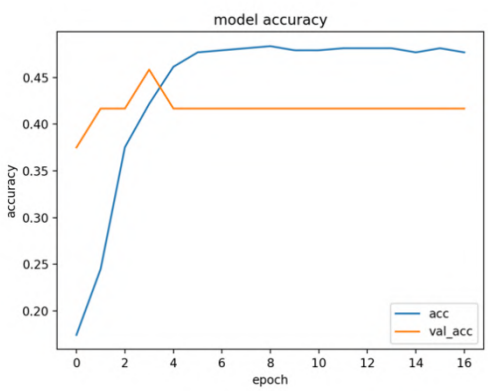
テストデータの分割法については K 分割法(交差検証)を用いた.K 分割とは,統計学において標本データを分割し，その一部をまず解析して，残る部分でその解析のテストを行い,解析自身の妥当性の検証・確認に当てる手法のことである.holdout 法という手法も存在するが,holdout 法で一部をテスト,一部をトレーニングデータに当てるとデータが少なすぎるため,公開は K 分割法を採用し全体を 3 分割した.

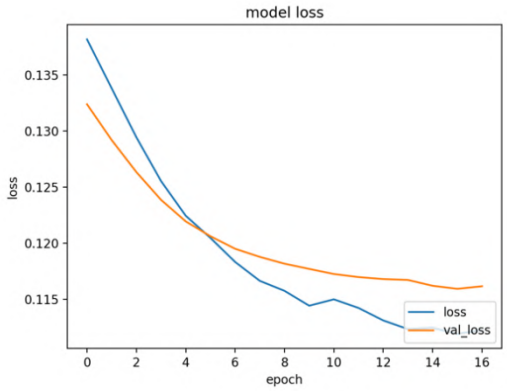
以下に各モデルの学習結果を示す.

モデル 1 (テンポ)

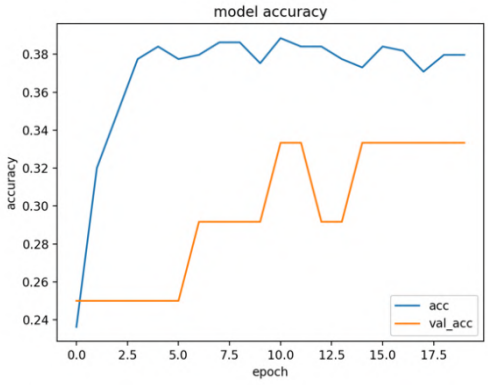


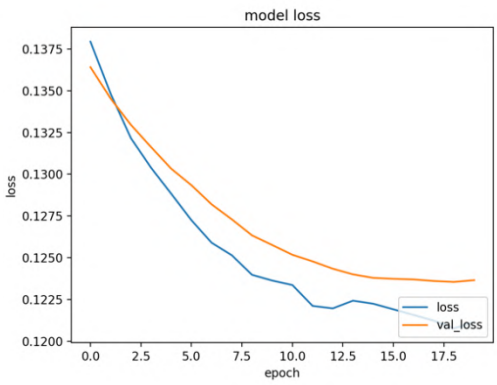
モデル 2 (スペクトルセントロイド)





モデル 3 (クロマグラム)





今回制作したニューラルネットワークには多数の問題が存在する.ひとつは,データ量が少ないためモデルの精度が低いということである.実際のゲームからデータを取得するのに時間がかかり,思うようにデータ数を増加できなかったことが原因である.ふたつめは,データ中に 0 の値が多かったということである.例えば,入力である感情を数値化した 8 データの中の一つの要素のみに 1 が入っており,それ以外の 7 つの要素に 0 が入っているデータが多数存在していたというのがあげられる.データに 0 が多すぎると学習がうまく行われない.これより,データの制作の時点で 0 が少なくなるような感情の数値化を行うべきだったと言える.

(＊文責：根本さくら)

4.3.3 選曲システム

4.3.3.1 選曲システム

イベントシーンに対しそのシーンに合った曲を選択する選曲システムの流れを解説する．物語班におけるイベント内容の自動生成システムによって生み出されたテキストファイルには表 4.3.3.1-1 のようにイベントシーンごとにイベント内容を示すタグが記載されている．中の e，[0,0,0.4,0,0,0.8,0,0]のように感情特徴量が記載されている．プログラムではまずテキスト中の感情

特徴量を全て読み取り格納しておく．次にその感情特徴量を順にそれぞれ学習済みの３つのニューラルネットワークへ入力すると，シーンに合った曲とされる未知の音響特徴量が得られる．この得られた音響特徴量と最も類似した音響特徴量を持つＢＧＭを，フリーＢＧＭの音響特徴量のデータリストの中から類似度を計算することによって選択し，そのシーンの曲番号を決定する．これを読み取った感情特徴量のぶんだけ行い，シーンそれぞれに合った曲番号をつけて表 4.3.3.1-2 のようにテキストファイルに記載する．記載する際には e から始まる行を削除し，そこに m+曲番号の形で決定した曲番号を挿入する形式であり，システム班が m から始まる行を読み取りその番号の曲を鳴らす仕組みになっている．

表 4.3.3.1-1　物語班が生成したイベント内容のテキストファイル

Event(cE1,1,OB1,1){
トーシャ,ヒロイン
ディアリオ,主人公
cC(0)
pMap1
e[0,0,0.4,0,0,0.8,0,0]
r1,dE,f(0)
r2,dW,f(1)
g1,s1@「ディアリオー」
g2,s2@「どうした？」

表 4.3.3.1-2　曲番号を付加したテキストファイル

```
Event(cE1,1,OB1,1){
トーシャ,ヒロイン
ディアリオ,主人公
cC(0)
pMap1
m24
r1,dE,f(0)
r2,dW,f(1)
g1,s1@「ディアリオー」
g2,s2@「どうした？」
```

類似度の計算の詳細においては，音響特徴量のベクトルは３つのため，完全に一致している，２つ一致している，１つ一致している，完全に一致していない，の４パターンが存在する．よって一致している数が最も多いＢＧＭを選択し，同じ類似度の曲が複数存在する場合はそのなかからランダムで一曲を選択するようにした．

（＊文責：山内拓真）

4.3.4 展望

今回の音響分析では場面に対して hevner の 8 つの印象語群を用い，音楽分析では分析項目として「BPM」「Spectral centroid」「chroma Vector」の３つを使用した．ただ，この他に音楽面では楽譜が入手できると扱えるデータが増えるので，別項目を使用した学習によってより良い結果が得られる可能性がある．さらに，イベントシーンにおける場面の感情以外に，音楽要素と強い関係がある分析項目が発見される可能性もある．今後は新たな分析項目の考察が進むことを今後の展望とする．

（＊文責：長野恭介）

4.4 視覚班

4.1.1　ロゴ

本プロジェクトを遂行するにあたって，チームのトレードマークとなるものとして，CreativeAI のロゴを制作した．クリエイティブ性と中世の世界観とコンピュータのイメージを象徴するため，筆記体風＋ゴシック体のロゴを制作した．制作の流れとしては，鉛筆で「CreativeAI」の筆記体を紙に書き，その紙をスキャナーでスキャンし，Adobe Illustrator の画像トレース機能を使ってベクターデータを起こした．本ロゴは，ゲームのタイトル画面，PV，中間発表ならびに最終発表のメインビジュアル内で使用された．

今後の展望として，ブランドとなるロゴにおいては，ロゴのガイドライン，すなわちロゴの使用ルールを設けるのが好ましい．ロゴのガイドラインを設けることにより，他メンバーがロゴを使用してゲーム制作を行った際に，ロゴデザインに変化が加わり，ロゴのコンセプトやデザインを著しく損なうのを防ぐことができる．特に今回，Unity にロゴをインポートし UI に配置した際に，ロゴ使用時の禁止事項などを特に書けなかったせいで，縦横比が変わったロゴがタイトル画面として使用されるという事態があった．そのため，今後新しくロゴを制作する場合は，ロゴの禁止事項などをまとめたガイドラインを設けることが好ましいと感じた．

（＊文責：小川卓也）

4.4.2 モデル

4.4.2.1 モデリング

モデリングとは 3 次元コンピューターグラフィックスで立体的な形状の物質を形成することである．本プロジェクトでは視覚的な表現を 3 次元空間で表現する方針となったため，オブジェクトに 3D モデルを使用している．視覚班ではゲーム内で扱うモデルの規格を統一するために素体作成し，それをベースにして各モデルを作成した．

本プロジェクトでは複数リリースされているモデリングツールの中から無料で公開されている 3DCG 作成ソフトの Blender を用いた．理由については，3DCG を扱うに際に Blender を使用したことのあるメンバーが多かったことが挙げられる．Blender のバージョンについては Blender 2.79 から Blender 2.8 へと後期のプロジェクト中にアップデートした．また，モデル制作のみではなく，いくつかのモデルのモーション作成にも用いた．以下にモデル作成の手順を記す．

（＊文責：蓬畑旺周）

4.4.2.1.1 素体の作成

初めに資料として，インターネット上で無料配布されている三面図の入手と，公開されている素体モデルの観察を行った．それらをもとに服や髪のついていないベースとなる素体モデルを作成した．作成した素体モデルは男性用，女性用，ゲーム内で動作する 2 頭身モデルの 3 種類である．2 頭身モデルには男性用，女性用の区別はなく単一のモデルを使用した．また，単一のモデルを使用することでモーションをそのまま流用することもできた．

（＊文責：蓬畑旺周）

4.4.2.1.2 キャラクターのラフモデリング

作成した素体をベースに，一部のメインキャラクターはキャラクターデザインとしてアウトプットされたイラストから，髪型や服装等を大まかに設計した．キャラクターデザインの製作者と小まめに連絡を取りながら作業をすることで，製作者のイメージから大きく逸れることがないよう心掛けた．また，今回制作したゲーム内で用いるキャラクターモデルの種類は主人公モデル 1 種，ヒロインモデル 1 種，用心棒モデル 1 種，教会のシスターモデル 1 種，ゲームのガイド役モデル 1 種，モブモデル 3 種，敵モデ

ル３種となった.

（＊文責：蓬畑旺周）

4.4.2.2　キャラクターモデル

制作した各キャラクターモデルについての詳細を以下に記す.

（＊文責：蓬畑旺周）

4.4.2.2.1 主人公モデル

主人公は 2 頭身モデルを前期に制作した．前期の段階で，主人公モデルはキャラクターデザインとしてアウトプットされたイラストとをもとにモデル化した．人型モデルを作る際には T ポーズと A ポーズがあるが，今回は T ポーズを採用した．T ポーズの利点はモデリング，リギングがしやすいといったことが挙げられる．前期の時点では主人公にモーショキャプチャを用いてモーションを実装する方向性であったため，腕が肩以上の高さに上がる場合を考慮していた．腕が肩以上に上がることのないモデルは基本的に A ポーズを採用している．

初めに 2 頭身キャラクターの素体をベースに，キャラクターデザインと比較しながら髪型から制作した. 髪の部分は半球をもとに, 分け目の部分を新しい面として変形し張ることで髪の全体の形を整えた. 大まかな形が決まったあとに髪の厚み付けをすることで立体感と奥行きをもたせた．髪の裏側は見えないメシュを削除することでポリゴン数を削減した．次に素体をもとにキャラクターデザインと比較しながら衣類の制作をした．主人公モデルはジャケットを羽織るデザインであったため，素体を変形させた体部分の上に覆いかぶさるように面を張り変形させて形を整えた. 2 頭身モデルの△ポリゴン数は 10,692 となった.

（＊文責：蓬畑旺周）

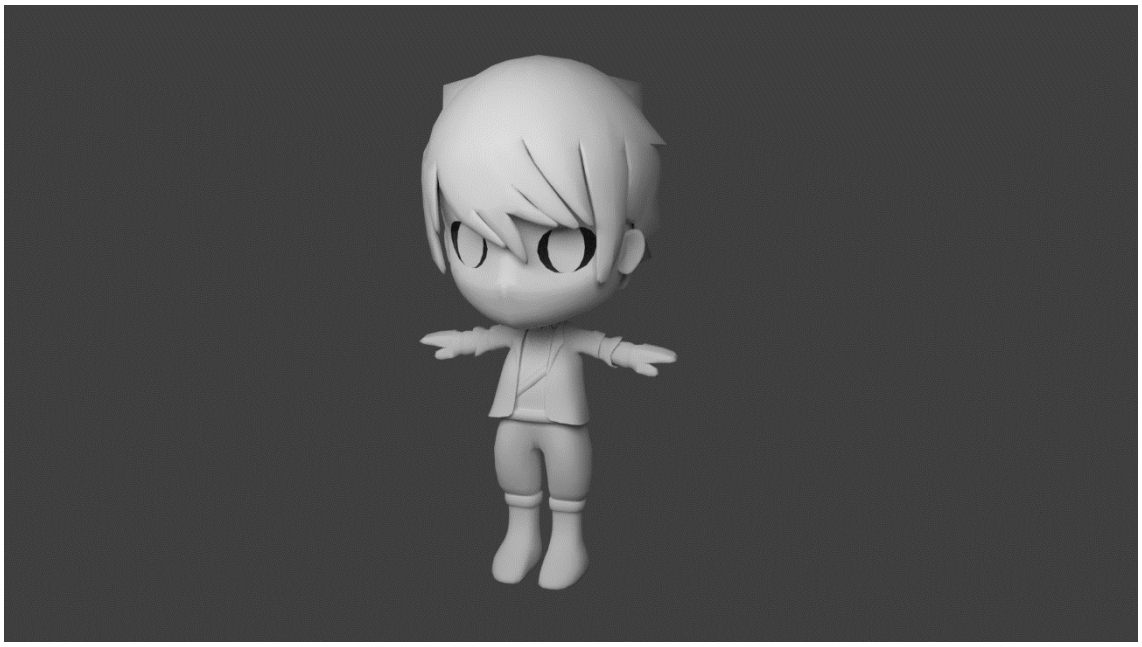


図 4.4.2.2.1-1

4.4.2.2.2　ヒロインモデル

ヒロインのモデルは等身大モデルと 2 頭身モデルの 2 種を前期に制作した．ヒロインモデルはキャラクターデザインとしてアウトプットされたイラストとをもとにモデル化した．人型モデルを作る際には T ポーズと A ポーズがあるが，等身大モデルは A ポーズを採用した．モーションキャプチャを用いてのモーション実装をする方向性であったが，キャラクターデザインでは A ポーズであり，A ポーズであったほうがモデル化しやすかったためである．2 頭身モデルは主人公モデル同様に T ポーズを採用した．

初めに等身大キャラクターは女性用の素体をもとに，キャラクターデザインと比較しながら髪型から制作した．髪の部分は半球をもとに，分け目の部分を新しい面として変形し張ることで髪の全体の形を整えた．大まかな形が決まったあとに髪の厚み付けをすることで立体感と奥行きをもたせた．髪の裏側は見えないメシュを削除することでポリゴン数を削減した．次に素体をもとにキャラクターデザインと比較しながら衣類の制作をした．ヒロインモデルは主人公モデルと同様にジャケットを羽織るデザインであったため，素体を変形させた体部分の上に覆いかぶさるように面を張り変形させて形を整えた．

2 頭身モデルでは，2 頭身キャラクターの素体をベースに，髪，小物，衣類を等身大モデルと同じものを使用することで作業工程数を削減した．等身大モデルと 2 頭身モデルではそのままのオブジェクトを使用するとサイズ感に違いが出てくるため，2 頭身モデルに適当なサイズに調整した．等身大モデルの△ポリゴン数は 21,170，2 頭身モデルの△ポリゴン数は 12,854 となった．

（＊文責：蓬畑旺周）

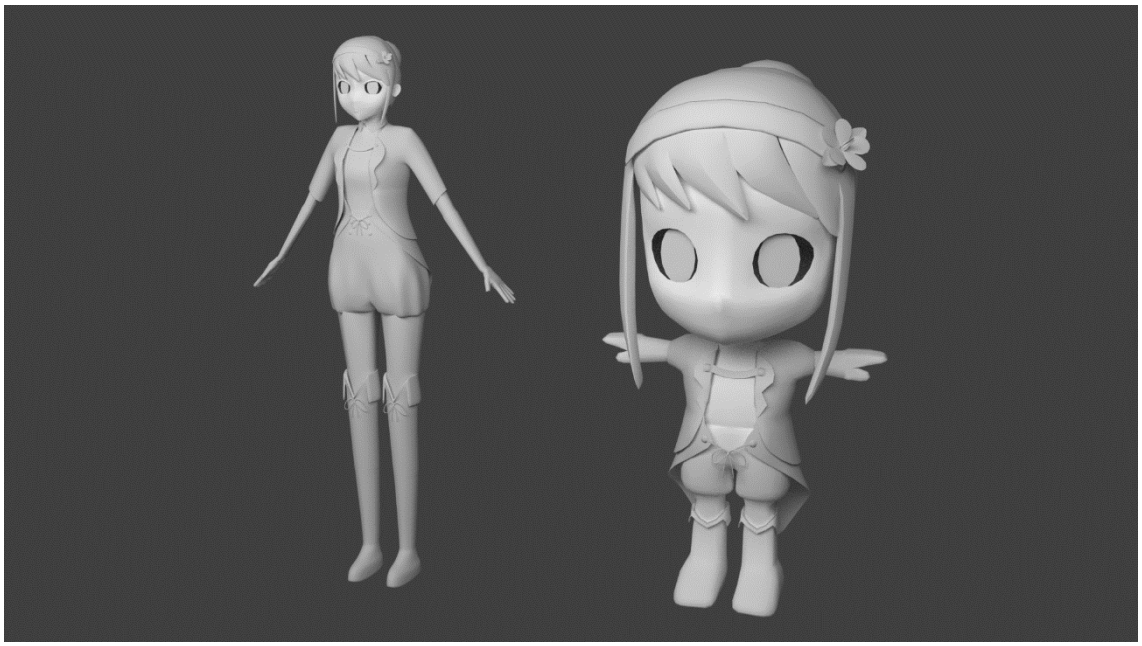


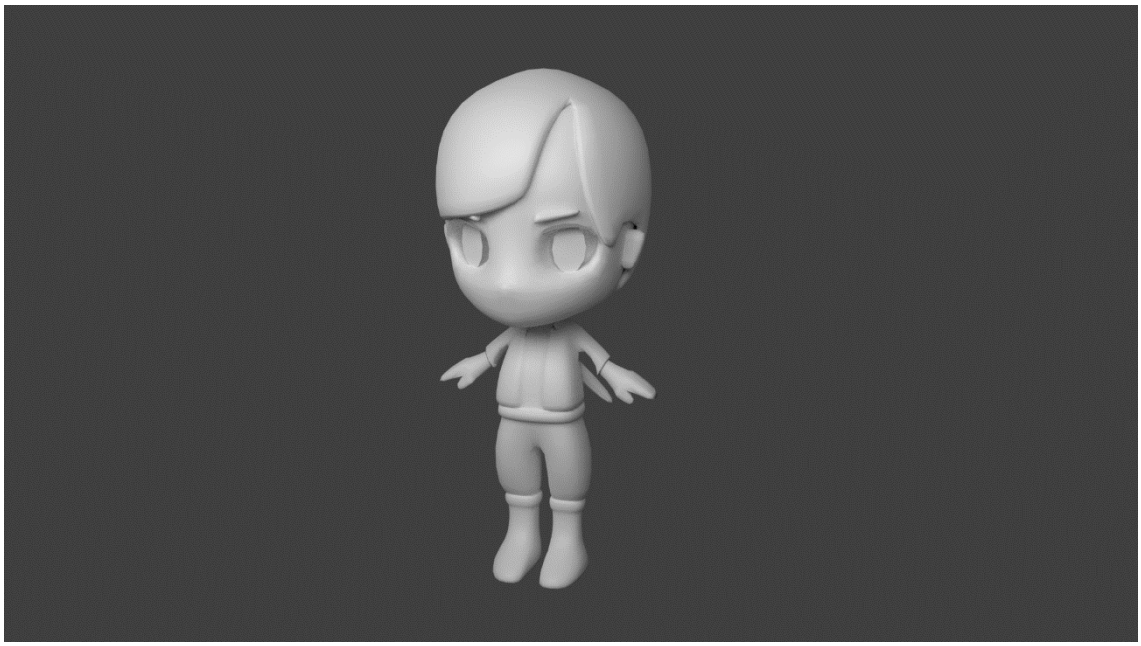
図 4.4.2.2.2-1

4.4.2.2.3　用心棒モデル

用心棒モデルは 2 頭身モデルを後期に制作した．用心棒モデルにはキャラクターデザインはなく，用心棒にふさわしい人物像を参考にモデル化した．人型モデルを作る際には T ポーズと A ポーズがあるが，今回は A ポーズを採用した．主人公モデルとヒロインモデルとは違い，モーションキャプチャでのモーション実装の予定はなかったためである．また，A ポーズの利点としては T ポーズと違い，自然な形になるためである．歩行モーションを設定する際も前期に手掛けた主人公モデルとヒロインモデルと比べて自然な歩き方になった．

初めに 2 頭身キャラクターの素体をベースに，参考となる写真，イラストを参考にしつつ髪型から制作した．髪の部分は半球をもとに，分け目の部分を新しい面として変形し張ることで髪の全体の形を整えた．大まかな形が決まったあとに髪の厚み付けをすることで立体感と奥行きをもたせた．髪の裏側は見えないメシュを削除することでポリゴン数を削減した．次に素体をもとに参考となる写真，イラストを参考にしながら衣類の制作をした．用心棒モデルは主人公モデルと同じくジャケットを羽織るデザインであったが，構造が単純であったため素体を変形することで十分であった．用心棒モデルの△ポリゴン数は 8,320 となった．

（＊文責：蓬畑旺周）



図

4.4.2.2.4　シスターモデル

シスターモデルは前期に制作された 2 頭身モデルを素体にして後期に制作をした．人型モデルを作る際にはＴポーズとＡポーズがあるが，今回はＡポーズを採用した．主人公モ，デルとヒロインモデルとは違い，モーションキャプチャでのモーション実装の予定はなかったためである．A ポーズの利点としてはＴポーズと違い，自然 な形になるためである．歩行モーションを設定する際も前期に手掛けた主人公モデルとヒロインモデルと比べて自然な歩き方になった．また，モデルのモーションは前期のヒロイン

モデルのモーションを流用した．シスターにキャラクターデザインはなく，Web などの検索により出てくるシスターを参考にした．モデリングする際にシスターの特徴として抽出したものは，白いベールと黒い服装である．白いベールで髪型がほとんど隠れるため，髪型は簡素なものとした．シスターモデルの白いベールを再現するために，Blender に標準に搭載されているクロスシミュレーションを板状のモデルに紐づけた．これにより，頭にかけられた布の形をシミュレートすることで大まかな布のしわをモデルで再現した．これに調整を加えることで自然なベールを再現することができた．

（＊文責：友広純々野）

4.4.2.2.5　モブモデル

モブモデルは 2 頭身モデル 3 種を後期に制作した．モブモデルにはキャラクターデザインはなく，ほかのキャラクターと比べて情報量が控えめな方向性でモデル化した．主人公モデルとヒロインモデルとは違い，モーションキャプチャでのモーション実装の予定はなかったためである．また，A ポーズの利点としては T ポーズと違い，自然な形になるためである．歩行モーションを設定する際は，用心棒モデルからのモーションを流用した．

初めに 2 頭身キャラクターの素体をベースに，参考となる写真，イラストを参考にしつつ髪型から制作した．今回はモブキャラクターモデルを 3 種類にしたかったため，帽子のあるモブモデル，帽子のないモブモデル，髪の長い女性のモブモデルを用意した．それぞれモデル A，モデル B，モデル C とする．モデル A の帽子は円を変形，面の押出をすることで形を整えた．次に素体をもとに参考となる写真，イラストを参考にしながら衣類の制作をした．モデル C の髪の部分は半球をもとに，分け目の部分を新しい面として変形し張ることで髪の全体の形を整えた．モデル C の髪型は用心棒モデルから流用した．大まかな形が決まったあとに髪の厚み付けをすることで立体感と奥行きをもたせた．髪の裏側は見えないメシュを削除することでポリゴン数を削減した．モブモデルは衣類構造を単純にしたため，素体を変形することで十分であった．モデル A は△ポリゴン数 6,682，モデル B は△ポリゴン数 6,568，モデル C は△ポリゴン数 7,548 となった．

（＊文責：蓬畑旺周）

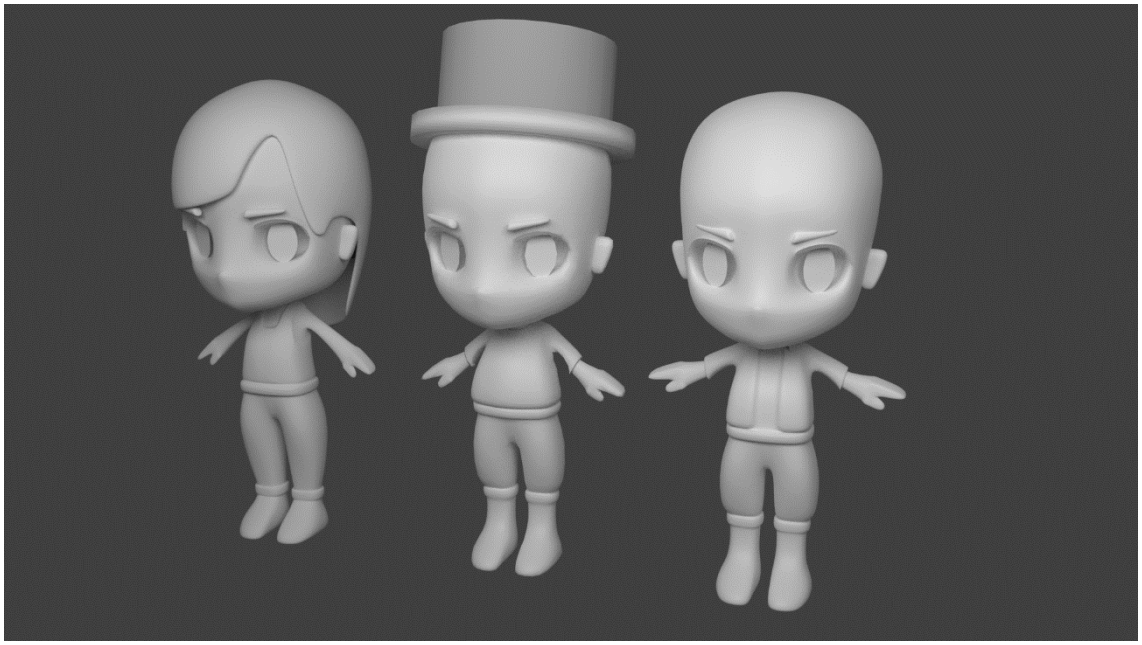


図 4.4.2.2.5-1

4.4.2.3　建築モデル

3D を用いたゲームを実装する際に，ゲームの舞台となるマップ上に町の構成要素である建築物と，ある街を象徴するランドマークを制作した．町は全 3 種あり，風車，灯台，小船，城壁の要素を取り入れ，各町のランドマークとした．ランドマークのほかに制作した建築モデルは 7 種ある．建築モデルの一部は一つのモデルを左右対称とするミラーリングや，オブジェクトのつけたしをすることで作業工程数を減らしつつ見た目を変えている．

（＊文責：蓬畑旺周）

図 4.4.2.3-1

4.4.2.4　エネミーモデル

モデルはゲーム内のシンボルエンカウントとして使用される．シンボルエンカウント[8] とは，主にコンピュータ RPG において，フィールド上で敵キャラクターのシンボルが見えており，プレイヤーキャラクターがそれに接触することで戦闘パートに移行するゲームシステムをいう．また，モデルを画像として出力ゲーム内戦闘パートにうつった時，敵の立ち絵として使用される．

（＊文責：友広純々野）

4.4.2.4.1　ナメクジ

ナメクジのモデルはカエルのモデルを流用し，後期に制作した ．ナメクジにデザイン等はなく，不気味な感じを出すために．ナメクジとクラゲを合わせたようなデザインを心掛けた．

（＊文責：友広純々野）

4.4.2.5.2　カエル

カエルのモデルは前期に既に制作してあったものを流用し，後期に制作した．カエルに前期にカエルのデザインを制作したためそれを基に制作した．

（＊文責：友広純々野）

-その他メモ-

・Blender v2.79 → v2.8 へとバージョンアップした際に，v2.8 で制作したモデルを v.2.79 で読み込んで編

集することができなかった.

・メッシュの表裏の確認 をする．Unity・Substance Painter で扱う際にテクスチャが正常に反映されない.

・Blender で細分化（subdivision surface）とミラーを適応する順番を変えると，境界部分の表示が変わる場合がある

（＊文責：蓬畑旺周）

4.4.3 ステージの制作

制作されたステージデザインを基に Blender と Unity を用いてステージの制作をおこなった．大まかなステージ製作法は，Blender で制作したモデルを fbx 形式で出力し，この fbx 形式のデータを Unity の Scene に反映させるという流れである．Unity の Scene とは，ゲームの環境とメニューが含まれているものである．各 Scene ファイルは一意のレベルであるため，構築する環境やモデルを Scene 毎に設定，配置をすることが可能である．シーンごとに各 Scene ファイルを制作することで，ステージ分けを容易におこなうことができる．なぜ Unity に反映させる作業をおこなったかというと，システム班への共有をする際にデータの崩れを抑えることができるからである．例えば，すべて Blender 側で調整をし，Unity へ反映させると，データが崩れてしまう部分が出てきてしまった．そこで，Blender でモデルを製作し，Unity 上でライティングなどの調整をおこなう事で，データの崩れを防ぎ，無駄な手間を省くことができた．中間発表までのステージの成果物は，テストシナリオの一番はじめのシーンで使用する，最初のステージ，2 番目のシーンで使用する孤児院内部のステージ，3 番目のシーンで使用する孤児院地下室のステージ，最終シーンであるダンジョン内の道に使用するモデルである．最終発表までのステージの成果物は，アルトリーヨ，トフー，オズの 3 つのステージとダンジョンである．

（＊文責：友広純々野）

4.4.3.1 ステージの詳細について

前期でも主人公が探索するためのステージがあったが，後期でも主人公が探索するためのステージを制作した．その数は前述した通り 3 種類である．それぞれのステージに名前がついており，アルトリーヨ，トフー，オズという名前である．2 つの町を互いに行き来することができるが，最初の町であるアルトリーヨは前期で制作した孤児院とつながっている．行き来ができるステージはアルトリーヨと孤児院，アルトリーヨとトフー，トフーとオズ，オズとアルトリーヨである．アルトリーヨと孤児院，それぞれの町とダンジョンのゲーム内移動方法は徒歩で設定されている．また，アルトリーヨとトフーのゲーム内移動方法は船で設定されている．最後に，トフーとオズ，オズとアルトリーヨのゲーム内移動方法は馬車で設定されている．

さらに，各ステージにはそれぞれコンセプトがあり，それは別で詳細に説明する．また，町にはそれぞれその町の特徴を示した建物が存在している．それらをランドマークと呼ぶ．このランドマークは，物語分析班が制作したアルゴリズムに従って生成されたクエストが発生するための場所であり，生成されるクエスト数は可変のためランドマークも各ステージに最低 3 つ存在する．

（＊文責：友広純々野）

4.4.3.1.1　アルトリーヨ

この町のコンセプトは，緑が少な目で，港通り等のテーマがある．クエストが発生したときのランドマークとして，灯台，船といったものがある．参考にした資料は中世ヨーロッパの海沿いにある町である．石造りの建築物が多く建物と建物の立地に高低差があるためである．

また，トフーへと続く道を作るため，土色の道以外に海が存在する．地形は Unity で標準に使用することができる Terrain で制作した．また，海に使用した水面や地形のテクスチャはフリーで使用可能な NatureStarterKit2 Assets と Standard Assets の中にある Environment Assets を使用している．

この町の課題としては，実際にゲームをプレイしてみると，町の外に通じる出口が３つ（ダンジョンへ行く出口，トフーへ行く出口，オズへ行く出口）あるにも関わらず，看板等の情報がないため３つの出口の区別が付かない点である．またこの出口とクエストが発生する場所の区別が付きづらい点もある．そのため，町自体は狭いがプレイヤーが他の町へ行こうとする際に迷ってしまう可能性が高い．

（＊文責：友広純々野）

図 4.4.3.1.1-1　アルトリーヨの全景

4.4.3.1.2　トフー

この町のコンセプトは水辺ののどかな農村といったテーマがある．クエストが発生したときのランドマークとして，畑，風車といったものがある．アルトリーヨへと続く道を作るため，土色の道以外に山と川が存在する．山と地形は Unity で標準に搭載されている Terrain で制作した．また，川に使用した水面や緑のテクスチャはフリーで使用可能な NatureStarterKit2 Assets と Standard Assets の中にある Environment Assets を使用している．トフーのランドマークは風車，畑と，部屋の内部に入ることができる建物が一軒存在する．プレイヤーが迷わないようにクエストが発生する可能性のある場所にのみ道を制作している．また，移動できる範囲をなるべく制限し，その制限がプレイヤーに伝わるようにしている．この移動可能範囲の制限をつくるために町全体を柵で囲んである．この町の課題としては，実際にゲームをプレイしてみると，町の外に通じる出口が３つ（ダンジョンへ行く出口，アルトリーヨへ行く出口，オズへ行く出口）あるにも関わらず，看板等の情報がないため 3 つの出口の区別が付かない点であ

る．またこの出口とクエストが発生する場所の区別が付きづらい点もある．そのため，町自体は狭いがプレイヤーが他の町へ行こうとする際に迷ってしまう可能性が高い．

（＊文責：友広純々野）

図 4.4.3.1.2-1　トフーの全景

4.4.3.1.3 ダンジョン

今回制作したロールプレイングゲームではダンジョンを自動生成したものとしている．そこで用いるモデルは生成されたダンジョンの数列に合わせて変更可能でないとならない．そこで，複数の壁のモデルを用意し，処理をした数列に合わせて配置することでプレイ可能なダンジョンとした．通路に接しておらず上面だけのもの，１面のみ通路に接しているもの，二面が通路に接しており角になっているもの，二面が通路に接しており隅になっているもの，三面を通路に接しているもの，八近傍で斜めに通路と接しているものを制作し，それぞれを回転させることでプレイ可能なダンジョンにするための 19 個のモデルを用意した．

（＊文責：白石智誠）

4.4.4 マテリアル

前期の活動に引き続きモデルのマテリアル制作は Substance Pinter を用いて行った．Substance Pinter を用いるメリットは豊富なリソースがデフォルトで利用可能なことである．ドラッグアンドドロップだけでも多くのマテリアルを表現可能で，マスクやブラシ，Procedurals などの使い方を学ぶことで，新たなリソースを追加することなく，基本的なマテリアルの制作をひとつのツールで制作可能である．本プロジェクトでは，これらを用いて必要なモデルに対して十分なクオリティのマテリアルを制作した．

（＊文責：白石智誠）

4.4.4.1 モブキャラクターのマテリアル

肌の色は smart material の Skin Face を用いて表現した．Skin Face では鋭い頂点ほど赤くなるようになっているため，目の周辺や頬は赤くなる．これを用いることで表現の難しい肌の制作を短時間かつキャラクターのサイズに対して十分なクオリティとした．

目や髪などは用意されているものでは表現しにくいため，ブラシ機能を用いてペイントした．
服は Fabric Rough, Fabric Rough Aligned, などを用いて表現した．部分的に金属表現を入れることで世界観とキャラクターデザインに沿ったものにした．また，本プロジェクトで制作したロールプレイングゲームのシステムではキャラクターにインタラクションすることでクエストが開始されるため，クエストに関係するキャラクターの見分けがつかなければならない．そこで，三種類のモデルに対してそれぞれ5種類のマテリアルを制作することで，キャラクターの識別を可能にした．
これらのマテリアルの制作にあたって，substance painter での編集はベースカラーのみにし，特殊な柄なども substance painter の Grunges や Procedurals などのリソースを用いることにより表現した．このようにベースカラーのみを変更するようにすると，各マテリアルでメタリックマップとディフューズマップを共用できるようにし，unity 上での読み込みと動作及びファイルサイズの軽減を試みた．

（＊文責：白石智誠）

4.4.4.2 メインキャラクターのマテリアル

メインキャラクターはモブキャラクターと同様に Skin Face を用いて肌を制作し，服も Fabric Rough, Fabric Soft denim, Fabric Rough Aligned, Fabric Suit Vintage,などをもとに光沢や密度などを調整していった．また，目や髪はブラシ機能を用いたペイントでの表現とした．

（＊文責：白石智誠）

4.4.4.2.1 主人公のマテリアル

シャツは Fabric Suit Vintage をもとに白くするとともにディティールを弱くすることで古さが出ないようにした．ベストは Fabric Knit Sweater を用いて粗目の素材感を表現した．ジャケットも Fabric Suit Vintage だが，シャツよりも目が粗くなるように調整した．ズボンには Fabric Rough の色を調整して用いた．靴などの革の部分は Leather bag のラフネスを下げて光沢感を出した．また，靴の装飾はアルファ素材をハイトに用いたブラシを使用して表現した．

（＊文責：白石智誠）

4.4.4.2.2 ヒロインのマテリアル

シャツは主人公と同様に Fabric Suit Vintage を用いたが，より古さを抑えて清潔感を出した．ジャケットは Fabric soft denim を用いて主人公のものよりも厚手に見えるようにした．ズボンは Fabric denim base を用いた．燕尾の部分には Fabric Suit Vintage を目が粗くなるようにして用いた．各署の金属部分や紐などはディティールが出ないように単色で制作した．

（＊文責：白石智誠）

4.4.4.2.3 旧友のマテリアル

ローブは Fabric Suit Vintage の目を粗くすることで質感を表現した．ローブの模様はアルファ素材をハイトに用いメタリックの値を上げたブラシを使用した．杖や頭などの宝石の装飾はプラスチックに近いマテリアルを用い，反射光を焼きこむことで表現した．

（＊文責：白石智誠）

4.4.4.3 敵キャラクターモデル

敵として用いるヘビ，カエル，ナメクジのモデルもキャラクターのモデルと同様にそれぞれに５種類のマテリアルを用意した．敵キャラクターのモデルはゲーム内の戦闘で同じ敵ばかりでは単調になるため，多くの種類を登場させるためにこの方法を取った．ゲーム内で使用するのは制作したモデルを blender 上でレンダリングした画像を用いるため，キャラクターのようなデータサイズの削減が不要のため，ラフネスやメタリックなどの他のパラメータも変更しマテリアルごとの違いを大きくした．マテリアルと戦闘で使用するステータスが決まっているため，ステータスが高いものには暗い色や毒々しい色を適用し，柄やハイト，ラフネスなどに substance painter の Grunges や Procedurals よりノイズを使用し密度を出した．

（＊文責：白石智誠）

4.4.4.4 町モデル

今回制作したロールプレイングゲームでは３つの町を扱うものとしたため，これらに用いる町の建物のマテリアルを制作した．すべての町が同じ雰囲気になるのを避けるため，町ごとで屋根の色を変更した．これはキャラクターモデルと同様にベースカラーのみの変更とし，処理の軽減とファイルサイズの削減を行った．Wood American Chery，Wood Rough を用いて建物の外壁を表現した．また建物の屋根は Wood American Chery からラフネスやノーマルを用いて，ベースカラーのみを任意の色にした．

またランドマークの風車，城壁，灯台のモデルはそれぞれにマテリアルを制作した．風車はほかの建物と同様のリソースを用いて制作した．城壁と灯台は Concrete Bare で石の質感を表現し，Procedurals より Bricks generator をハイトマップに用いてレンガの凹凸を表現した．灯台の上部にレンガの凹凸を入れないために Bricks generator のレイヤーにマスクを用いた．

（＊文責：白石智誠）

4.4.4.5 室内のモデル

建物のモデルと同様に Wood American Chery，Wood Rough を用いて床や壁，机などを表現した．建物とは縮尺が異なり，ノーマルマップやハイトマップを適応するとディティールが増えすぎるため，ノーマルチャンネルやハイトチャンネルを無効化するなどして違和感をなくした．暖炉や時計には金属のマテリアルを適応し，木材と同様にディティールが強くなりすぎないように調整した．

（＊文責：白石智誠）

4.4.4.6 ダンジョンモデルのマテリアル

各町に隣接して一つずつダンジョンがあるという設定とシステムに合わせて，黒い岩と白い岩とツタが絡んだ岩の三種類のマテリアルを制作したプリミティブのコンクリートに対してハイトマップで凹凸を出すことによって岩のような見た目にした．この質感にベースカラーを黒よりの灰色と白よりの灰色と中間の色のものを制作した．ツタが絡んだ状態は substance painter のツールより Ivy Branch を用いてペイントした．また，上面のみのモデルは繰り返しになりやすいため，目立たないように UV を調整し繰り返しが分かりにくいようなテクスチャにした

（＊文責：白石智誠）

4.4.5 UI（ユーザーインターフェース）

本ゲームは，前期の時点においてデバック用としての最低限の UI しか設計しておらず，操作についての説明などは一切明記していなかった．この状態では，評価実験のゲーム操作で被験者ないしユーザーに混乱を招き，ユーザビリティが低下する可能性があることが題視された．そこで後期では，本ゲームの UI デザインを見直し，ユーザーの混乱を防ぎ，操作性を高めることを目的に，新たに UI をデザインすることとなった．制作ツールには，ベクターデータの図形が簡単に素早く作れるのと，修正などに柔軟に対応できるのが特徴の Adobe Illustrator を採用した．

本ツールで，UI の素材制作からレイアウト設計までを一貫して行い，完成した素材をシステム班に共有し，Unity にて UI のシステムを実装する制作形態をとった．反省点としては，UI 実装後での一部ゲームシーンのデバッグにて，フォントサイズが比較的小さいためか文章やパラメータの確認し辛かったことである．UI 画面のスペースにも余裕があったため，今後の UI 設計は，今回の設計よりもフォントサイズを少し大きめに設計するのが好ましい．

（＊文責：小川卓也）

4.4.5.1 トーン&マナー

まず初めに，UI デザインに一貫性を持たせるため，UI のトーン＆マナーを大まかに決めることにした．「中世という時代設定」と「AI を活用したゲーム」の 2 つの要素から，「中世っぽさと堅実さのあるシンプルなデザイン」というキーワードを UI デザインの指針とした．今回は作業期間を考慮して，アニメーションによる遷移表現ではなく，色による遷移表現を条件に UI デザインを行った．

（＊文責：小川卓也）

4.4.5.2 制作時のルール

UI デザインを行う前に，以下のようにいくつかのルールを設けた．これらのルールに則って UI デザインを行うことで，Adobe Illustrator 内の UI 素材の配置を Unity 内でも忠実に再現できると考えた．

- UI 素材の幅，高さ，位置座標，マージ，フォントサイズ，行間の指定は整数値
    - Unity で UI 素材のサイズを指定しやすくするため
- カラーモードは RGB
    - Unity で扱いやすいカラーコードで指定できるため
- UI 素材のファイル名は 1 バイト文字
    - 2 バイト文字による読み込みエラーのリスクを避ける
- 命名規則は UI_[どのシーンの]_[何の UI 素材か].xxx(ファイル拡張子)
    - サムネイルのみだとどの UI 素材かの判別が難しいため
- UI 素材は等倍，2 倍の 2 種類のデータでそれぞれ書き出す
    - Unity 内で UI のサイズの修正が必要になった時，等倍のデータの拡大によって UI 素

材が劣化してしまうのを避けるため

- （＊文責：小川卓也）

4.4.5.3 制作の流れ

まず，UI 基本となるタイトル，イベント CG，マップ中会話，マップ移動，メニュー，戦闘画面，ゲームオーバーシーンを想定して UI 素材の元となる図形の作成を行った．中世という設定から，基本テーマカラーはベージュとした．更に，メニューの非選択時と選択時の表現は，薄いベージュと濃いベージュによって，UI 遷移を表現した．

使用するフォント選別においては，windows10 と Mac 両方で使用できるフリーフォントの中でも，幅広い文字コードをサポートし，かつ可読性の優れるものを選ぶ必要があった．メインフォントには，マップ中のデフォルメされたキャラクターや建物の雰囲気を考慮して，丸みを帯びた MotoyaLMaru W3 mono を採用した．しかし，数値において，文体が細くて可読性に欠けるという観点から，HP, MP 等の数値には Noto Sans JP を採用した．作成した UI の元となる図形を，実際のゲーム画面を背景として，Unity で実装した際にどのようなレイアウトになるのかを決め，既存ゲーム作品の UI を参考に，図形を配置していった．次に，配置した図形や文字を，Unity で実装可能な形で共有した．出力した画像データは，UI 素材として集約し，完成したデータは等倍透過 PNG，2 倍透過 PNG の 2 種の形式で出力し，システム設計メンバーが必要に応じて自由に使ってもらえるように共有した．

（＊文責：小川卓也）

4.4.6 エフェクト

ゲームの戦闘シーンにおいて，敵への魔法攻撃を表現するエフェクトを制作した．制作したエフェクトは火属性，風属性，水属性の 3 種類である．これらのエフェクトは，モーショングラフィックスやビジュアルエフェクトが簡単に作れる Adobe AfterEffects と，AfterEffects 専用のパーティクルプラグイン Trapcode Particular 4 で制作した．コンポジション設定は，解像度を 1000x10000，フレームレートを 20fps，デュレーションを 40 フレームとした．初めに AfterEffects 内の Particular 4 で，エフェクトの大元となるパーティクルをシミュレーションし，火属性，風属性，水属性で形と色を変え，3 種類のエフェクトを制作した．制作したエフェクトを連番形式の透過 PNG で書き出し，Unity 上で連続再生してアニメーションさせることで魔法攻撃を表現した．

（＊文責：小川卓也）

4.4.7 メインビジュアル

メインビジュアルは，中間発表時の聴衆寄せを目的に制作した．映画風のレイアウトにした理由としては，遠くから見ても通行人の興味を引きやすく，ある程度真似しやすいレイアウトだったからである．メインとなるイメージには，別メンバーが手掛けたイベント CG のラフ画を使用し，本プロジェクトのクリエイティブな側面や開発段階である様子を表現した．後期の最終発表では，キャラクターのイラストを含む新メインビジュアルの制作を考えていたが，ポスター制作などの他に優先すべき作業があったた

め，制作を見送った．反省点としては，背景のイラスト部分を作業に余裕のあるメンバーに外注するべきだったと思ったが，そもそも，人物画が描けるメンバーがいないと実現は厳しいため，本プロジェクトでは，メインビジュアル制作の優先順位は低いものだと考えられる．しかし，中間発表ならびに最終発表では，メインビジュアルの聴衆寄せ効果は大きいものであったため，メインビジュアル制作によって，本プロジェクトの宣伝に大きく貢献することがわかった．

（＊文責：小川卓也）

4.4.8　まとめ

4.4.8.1　後期の活動の流れ

4.4.8.1.1　他班との連携，制作数決定

主に物語分析班とシステム班との相談を行った．物語分析班とは，前期から相談を行っていた，RPGのストーリーを進める上で必要なモブのキャラクター数，敵として出現するエネミー数やステージ数の相談を最初に行った．その他に，ステージの中でダンジョンの属性や，ステージが有する属性について検討を行っていた．

システム班とは，ステージやステージの中でもダンジョンの自動生成を検討した．また，UI 等は早期から話し合いをし，UI の中でも必要な項目をシステム班側と早期から話し合いを行った．また，ゲームの解像度の決定などを行った．

（＊文責：友広純々野）

4.4.8.1.2　目的

後期では前期で立てた目的とは別に以下 2 つの目的を立てた．

・生成されたクエストやダンジョンに対して矛盾のない視覚表現を検討する

・プレイヤーがゲームの世界観に入り込める見せ方を検討する

まず，一つ目の目的に対しては

本プロジェクトではロールプレイングゲームを題材とし，シナリオ，ステージ，BGM などゲームシステムの種々の要素に対して複数の自動生成アルゴリズムを適用したアプリケーションである．そのうちシナリオ，ステージに関しては，視覚班が担当する視覚的要素が発生する．シナリオは物語分析班が担当し，ステージはシステム班がそれぞれ担当している．

まず，シナリオについて説明をする．物語分析班の制作したロールプレイングゲームのシナリオを生成するアルゴリズムでは，ロールプレイングゲームにおけるクエストの連続性，マルコフ連鎖，遷移確立モデルを用いることで，およそ 1800 個程度のクエストを生成することができる．それに伴い，これほど多くのクエスト数に対応するように視覚的要素であるモデルすべてを用意するのは，限られた人数かつ制期間作するのは難しく生成されるクエストに対して，少ないモデル数で何度も使用すると，ゲームのプレイヤーは，登場するキャラクターや，ゲーム内で移動するステージに対して，見分けがつかずゲーム内で，混乱しまうと考えた．そこで，グループのモデル制作の負担を少なくし，多くの種類を制作するために，一つのモデルに対して，複数のマテリアルを用意することで生成される多くのクエスト数に対応す

る形をとった．なぜ，このような手法を取ったというと，モデリングとマテリアルを用意する労力を比較すると，マテリアルを用意するための期間のほうが短いからである．

（＊文責：友広純々野）

4.4.8.2　成果物について

最終発表までに制作したモデル数やマテリアル数は以下の表に示す．

表 4.4.8.2　制作したモデルの数，マテリアルの数

	モデル数	マテリアル数
建物	12	3
ダンジョン	16	3
キャラクター	6	20
エネミー	3	15

ここでは最終的にどのくらいの量ができたかを記述する．建物については，これらの建物のモデルを利用して移動や，クエストが発生する 3 ステージを制作した．ダンジョンについては，これらのモデル数と，森，岩（白，黒）の 3 種類の属性を用意し，自動的に生成されるダンジョンの形に対応をした．キャラクター，エネミーについては複数のマテリアルを用意することで，20,15 種類のモブキャラクターを制作した．

（＊文責：友広純々野）

4.5 「HAKODATE *アカデミックリンク* 2019」における活動

2019 年 11 月 9 日に北海道教育大学函館校にて，「HAKODATE *アカデミックリンク* 2019」，はこだて高等教育機関合同研究発表会が行われた．「HAKODATE *アカデミックリンク* 2019」とは，函館市内にある 8 つの高等学校教育機関の学生同士が，研究している内容や取組，成果などをポスターセッションやステージセッションで披露しあい，各々の研究テーマの連携や協力の可能性を模索する合同の研究発表会のことである．また，今回も函館市内にある高等学校や，道南圏と青森県内の大学による特別参加，企業･団体ブースの登場等があった．

「HAKODATE *アカデミックリンク* 2019」では，本プロジェクトはブースセッションに参加した．発表形式は，参加した 65 のチームが各自割り当てられたブースにて，パネル展示の前でデモンストレーションや解説を行うというものだった．研究成果を発表するために A1 サイズのポスター

2 枚とスライド，中間発表会で使用したデモ動画を用意し，発表を行った．また，発表会後に行われた交流会にも参加し，他校の学生・教員と交流を深めた．

（＊文責：斉藤勇璃）

4.6 成果発表会

2019 年 12 月 7 日に公立はこだて未来大学にて成果発表会が行われた．成果発表会を行うにあたって，中間発表と同じくプロジェクトの活動内容を説明するための A1 サイズのポスター 5 枚とスライド，制作したゲームの 1 分間のデモ動画を用意し，20 分のプレゼンテーションを 6 回行った．また, プレゼンテーションの聴講者に発表技術と発表内容についてのアンケートに回答してもらい，評価を受けた．

アンケートは発表技術と発表内容の項目に分け，10 段階評価とコメントを回答してもらった．

中間発表で 80 人から受けた評価は表 3.4.1 に示す結果となった．

評価点	発表内容	発表技術
1	0	0
2	0	0
3	0	0
4	1	1
5	3	2
6	5	2
7	13	10
8	33	26
9	11	13

10	10	19
無回答	4	7
平均	7.934211	8.369863

表 3.4.1 最終での評価

発表技術についてのコメントの一例を示す.

- 説明のない単語もあり，少しわかりづらかった.
- 用語についての解説をもう少しいただけるとありがたい.
- スライドはわかりやすくてよかった．十分な写真の量．声量が足りない.
- デモの活用が良かった.
- 話すテンポが少々はやいのかと思った．身振り手振りがあるのでわかりやすいと感じた.
- 減:観客整理

発表内容についてのコメントの一例を示す.

- たぶんいいものだと思います．我々の目が肥えている.
- 班別に順に説明されていて理解しやすかった.
- レベリングの機能がとても面白いと感じた．マップとクエストに関するクエストが発生するところが実際のゲームと似ていてよかった.
- AI によるクエスト生成，BGM セレクトはとても面白いと思った．自分の目的の 1 つに，アドバイス等が可能なナビゲーター風の AI を作って組み込むというものがあり，これもぜひ一度検討してみてほしい.

発表技術については, 中間発表であった発表者が発表原稿を見ているのが気になるという意見や, スライドの文字が小さいという意見がなくなっていたことから，課題でもあった発表準備がしっかり出来ていたと考える．しかし，今回のアンケートの結果では単語の説明が不十分であるという意見が多かった．もし今後機会があれば，そういった点にも配慮した発表準備を行いたい.

発表内容については，とても面白いといった意見が多かった．また，前期の課題であった，RPG を題材にしたことや 3 DCG を用いることについてなど，決定した事項についてなぜそのようにしたのかという理由の説明をきちんと行うことができた.

今後は，2020 年 2 月 16 日に行われる *プロジェクト学習*成果発表会&企業交流会 in 東京やその他の学会で，今回の反省を生かしていきたい.

（＊文責：斉藤勇璃）

4.7 第 2 期 enPiT における活動

第 2 期 enPiT とは，文部科学省事業である「成長分野を支える情報技術人材の育成拠点の形成」の愛称であり，ビッグデータ・AI 分野，セキュリティ分野，組込みシステム分野，ビジネスシステムデザイン分野の 4 つの分野における高度 IT 人材の育成を目指すプロジェクトである．今回は，ICT や IoT といった最新鋭の技術について理解を深め，ビジネスイノベーションを創造する人材の育成を目指す「ビジネスシステムデザイン分野」での人材育成を中心に取り組んでいた．

本プロジェクトでは代表者 1 名が，2019 年 12 月 8 日（土）に公立はこだて未来大学で開催された「函館高専・公立はこだて未来大学合同 PBL 発表会」に参加し，ポスターによる成果発表を行った．前日に公立はこだて未来大学で開催された成果発表会でも使用した A1 サイズのメインポスター1 枚と，視覚班が作成したビジュアルポスター1 枚，作成したゲームの 1 分間のデモ動画を用いて，発表を行った．発表形式は，参加した 14 つのグループを半分に分け，前半では片方が発表し，もう片方は発表を聞き，後半では逆に前半で発表したグループが聞き手に回り，前半で発表を聞いていたグループが発表をするという形式を取っていた．本プロジェクトは前半に他の発表を聞き，後半は発表を行った．

発表内容は，最初にプロジェクトの目標と成果，次に物語分析班，音響班，視覚班，システム班の主な活動内容と班ごとの成果，最後に今後の予定と展望を話した．発表後の質疑応答では，「このプロジェクトはどのような開発手法を使っていたのか」，「今後行う評価実験ではどのぐらいの規模でどのようなことを行うのか」，「このプロジェクトの成果はどのようなことに活用，または応用できるのか」等の質問があった．一部の質問は答えることができなかったものの，ほとんどの質問には適切な解答をすることができたと思われる．

今後同じような発表の場に恵まれたのであれば，すべての質問に答えられるよう，自分の班以外が行っている活動について理解し，プロジェクト全体で行ったことを把握しておけるように努力したい．また，発表が長すぎるとの意見もあったので，内容をもう少し簡潔に話せるように，事前に発表内容をまとめるなどの準備を怠らないようにしたいと思う．

（＊文責：斉藤勇璃）

参考文献

[1]松原仁, 佐藤理史, 赤川美奈, 角薫, 迎山和司, 中島秀之, 瀬名秀明, 村井源, 大塚裕子(2013) コンピュータに星新一のようなショートショートを創作させる試み. An attempt at automatic composition of Shin'ichi Hoshi-like short short stories. 第 27 回人工知能学会全国大会
[2]三宅陽一郎(2012) 次世代デジタルゲームにおける人工知能の研究課題について. Recent Research Topics Summaries for Next-generation Digital Game AI.
[3]三宅陽一郎(2015) ディジタルゲームにおける人工知能技術の応用の現在. Current Status of Applying Artificial Intelligence for Digital Games. 人工知能, 30(1), pp.45-64.

[4] 川野陽慈, 山野辺一記, 栗原聡(2018) シナリオ創発に向けたプロット生成に関する研究. Proposition of automatic plot generation framework for scenario building.
[5]小野淳平, 小方孝(2016) TRPG 方式に基づく物語自動生成ゲームにおける場面連鎖拡張機構の試作. A Prototype Systemof a SceneSequence ExpansionMechanismin an Automatic Narrative Generation Game based on TRPGMethod. 第 30 回人工知能学会全国大会.
[6] 豊澤修平, 工藤はるか, 石田晃大, 遠藤史央里, 川瀬稜人, 菊池亮太, 工藤健太郎, 栗原将風, 櫻井健太郎, 佐藤好高, 玉置秀基, 根本裕基, 原科充快, 久野露羽, 平田郁織, 村井源, 椿本弥生, 角薫, 松原仁(2017) 推理小説プロットを自動生成し映像化する統合的インタラクティブシステムの開発と評価. 研究報告人文科学とコンピュータ, Vol. 13, pp.1-5.
[7] 鈴木諒輔, 佐々木奨之, 袴田翔, 田中瑞穂, 三浦隆太郎, 城田晃希, 高橋翔太, 南部太雅, 山田康貴, 吉田拓海, 松浦史佳, 松原千里, 寺島啓悟, 津沢慎吾, 渡邊広基, 村井源, 迎山和司, 田柳恵美子, 平田圭二, 角薫, 松原仁(2018) 物語と情景描写を自動生成する統合的システムの検討と開発. 情報処理学会研究報告, Vol. 50, pp.1-8.
[8]はてなブログタグ（最終閲覧：2020 年 1 月 14 日）
https://d.hatena.ne.jp/keyword/%E3%82%B7%E3%83%B3%E3%83%9C%E3%83%AB%E3%82%A8%E3%83%B3%E3%82%AB%E3%82%A6%E3%83%B3%E3%83%88

（＊文責：根本さくら）