公立はこだて未来大学

2024 年度

システム情報科学実習

グループ報告書

Future University Hakodate 2024 System Information Science

Practice Group Report

プロジェクト名

クリエイティブ AI

Project Name

Creative AI

グループ名

システム班

Group Name

Foreshadowing Group

プロジェクト番号/Project No.

7

プロジェクトリーダ/Project Leader

1022017 小川昂 Ko Ogawa

グループリーダ/Group Leader

1022076 河西叶大 Kanata Kawanishi

グループメンバ/Group Member

1022024 井田和樹 Kazuki Ida

1022099 小松知佳 Chika Komatsu

1022235 小林優斗 Yuto Kobayashi

指導教員

村井源 迎山和司 中田隆行

Advisor

Hajime Murai Kazushi Mukaiyama Takayuki Nakata

提出日

2024 年 1 月 15 日

Date of Submission

January. 15, 2024

概要

クリエイティブAIでは、人工知能を用いた創造性の再現を目的としてゲームを制作し、人工知能の創作方面における有用性を検討した。創作の形態をゲームとした理由は、ゲームの中にはシナリオや音楽、視覚表現、ゲームシステムなど多様な創作性を表現することができるためである。また、創作した作品を実際に体験できるため、評価を行いやすいという点も理由として上げられる。

創作における面白さは未だに定義されていない領域であり、これを人工知能を通して解明していくことが最終的な目的である。

（＊文責：井田和樹）

目次

1章：はじめに

プロジェクトメンバー全体で話し合った結果、制作するジャンルはホラーアドベンチャーゲームとなった。
このプロジェクトでは、CreativeAIの先例に倣って、ゲームを制作するにあたり、それぞれの役割に応じて、伏線班、キャラクター班、視覚班、音響班、システム班の5つの班に分かれた。
システム班は各班の成果物を統合し、ゲームを完成させることを目標に活動している。

（＊文責：井田和樹）

2章：関連研究

今年度のクリエイティブAIのシステム班では、関連研究として昨年度のクリエイティブAIの活動を参考にした[1]。
昨年度はAIを用いて、サイドビュー型の2Dアクションゲームを制作していた。そのプロジェクトのシステム班の方のお時間をいただき、開発を行う上での注意点などを教えていただくことができた。以下に、大まかにその内容を記す。

- スケジュール管理
    - 中間発表までにはゲームが最低限動き、各機能の統合が行える状態にしておくと良い。メンバーのモチベーション向上、後期で統合やデバッグに専念できるなどの利点がある。
- タスク管理
    - これを怠ると、以下の問題が起こる可能性がある。
        - 同じ部分を2人で担当してしまう。
        - 無駄な機能を作ってしまう。
        - 旧バージョンを統合してしまう。
    - 仕事がなくなるという人が極力出ないように注意する。
    - 優先度を決めて取り掛かる。
- 他班との連携
    - 画像サイズや、データのフォーマット等について早めに双方の間で合意を得た方が良い。
    - 上記の形式を決めた上で、ダミーのデータをもらっておくとシステム班が早めに作業を開始できる。
    - ゲームの各要素の仕様、期限、決定権を持つ人は決めておく。
- 開発上の注意点
    - 文字コードは統一する。

（＊文責：井田和樹）

3章：プロジェクト学習の目標

システム班は人工知能で生成された他班の成果物を統合し、ホラーアドベンチャーゲームとして完成させることを目標として開発を行った。統合時にはバグを減らすためにGitHubを活用し定期的なコードの評価とデバッグを行った。また、他班とのデータのやり取りにはJSONファイル形式を採用し、他班の要望に柔軟に対応できるようにした。

また、このプロジェクトではAIを用いてゲームを制作することが一つの目標になっているが、システム班では、アドベンチャーゲームというジャンルがAIと親和性が低いことや、前期終了時点で開発の進捗や速度から、AIを用いないことに決めた。しかし、開発中の疑問点等を生成系AIに質問するなどの一般的な使用法は取り入れている。

（＊文責：井田和樹）

4章：目的を達成するための手法、手段

4.1 Unityの学習

システム班ではメンバーのほとんどがUnityを使用した経験はなく、大学の講義等で学習したプログラミング言語の知識はあるという状態であった。そのため、Unityを使用するにあたって、操作方法やC#スクリプトの書き方等について勉強する必要が生じた。既存のチュートリアルをこなす方法では、操作方法が身に付けやすいが、コーディングを学ぶ機会が少ないことから、C#スクリプトの書き方を学び、正常に動作するプログラムを記述できるようにすることを目的に、用意したテストケースを通すプログラムを記述する方法で学習を行うことにした。

方法としてはUnityのTest Frameworkを利用してリーダーがテストケースを作成し、学習者にテストケースを全て満たすようなプログラムを記述してもらった。このとき、作成するコードについてはあらかじめスクリプトに関数を記述し、C#ドキュメントコメントを使用して説明を記述した。学習者はUnityのTest Runnerウインドウからテストを実行し、テストの成功失敗を確認し、失敗した場合は再度プログラムを書き直すという流れで行った。テストケースはEditModeテストとPlayModeテストの両方を使用して作成した。例としては

- シーン遷移
- プレイヤー入力
- UI
- プレイヤー位置
- 画像の表示
- 物理特性
- 当たり判定
- 音楽の再生

など、Unityの基本的な機能の使い方に関するテストケースを作成した。

（＊文責：河西叶大）

4.2 各機能

4.2.1 スクリプトエンジン

限られた時間でゲームを完成させるためにスクリプトエンジンを作成した。前期のシステム班は、主に伏線班の要請に合わせて、各機能を作成していた。しかし、この手法では、システム班の作業の進捗が伏線班の作業の進捗に制限されてしまっていた。また、シナリオの完成は10月を予定していたため、そこから統合作業を始めるとゲームが完成しない可能性があった。そこで、このシステムを使用することで、Unityに詳しくないメンバーであっても統合作業を行えるようにした。

ここでのスクリプトエンジンとは、JSONファイルに内容を記述し、それらの内容をデータクラスに変換して、ゲーム内のオブジェクト等に反映させるものを指す。スクリプトエンジンを構成するエンジンについて、以下のものが挙げられる。

- オブジェクトエンジン
- マップエンジン
- 会話エンジン

各エンジンはMessagePack for C#[2]を使用して、開発ではStreamingAssetsフォルダで、成果物はResourcesフォルダを使用してJSONファイルを読み込んだ。成果物にResourcesフォルダを使用した理由は、プラットフォームの制限により、StreamingAssetsフォルダが使用できないためである。

（＊文責：河西叶大）

4.2.1.1 オブジェクトエンジン

オブジェクトデータはマップ名と発動する場所の座標、イベントの発動条件と発動するイベントの種類を指定することで、イベントを呼び出すことができるようにした。このとき、発動条件はフラグ条件と規定のイベント発動条件を組み合わせて記述することができる。イベントの発動条件とイベントの種類は以下の通りである。

イベントの発動条件

- オブジェクトと座標が一致した時
- オブジェクトの周囲1マス（4方向）にいて、オブジェクトの方向を向いている時、決定キーを押す
- オブジェクトの周囲1マス（4方向）または、オブジェクトと座標が一致していて、オブジェクトの方向を向いている時、決定キーを押す
- オブジェクトと座標が一致していて、決定キーを押す

イベントの種類

- ChangeScene ”マップ名,x座標,y座標”
  対応したシーンの指定した場所に移動
- PlayerMove ”x座標,y座標”
  同マップの指定した場所に瞬間移動
- Conversation 文字列引数
  文字列引数に渡したJSONファイルの会話を開始する
- GetItem 文字列引数
  文字列引数に対応したアイテムを取得する
- PaperGame
  重ね合わせゲームを開始する
- SearchGame
  サーチゲームを開始する
- TimingGame
  タイミングゲームを開始する
- TileModify ”マップ名 [StylesFront | StylesMiddle | StylesMiddle | Tiles] x,y タイル記号”
  タイルを指定した種類のものに変更する
- KillCharacters
  登録されているそのシーン内のキャラチップを全て非表示にする

オブジェクトデータのjsonファイルの作成は伏線班に依頼した。

（＊文責：河西叶大）

4.2.1.2 マップエンジン

マップエンジンとはJSONファイルに記述されたマップデータをシーン読み込み時に読み込んで各シーンのTilemapオブジェクトにタイルを配置するシステムである。
マップデータは当たり判定情報を決める部分と、3層の表示タイルの部分で構成されており、表示タイルは画面のタイルの前面に表示するタイルを記述する部分と、背面に表示するタイルを記述する部分と、その間に表示するタイルを記述する部分で記述することができる。

マップデータとタイルの対応については、1文字の記号とタイル画像のファイル名の対応の決定を視覚班に依頼し、それをScriptableObjectとして実装することで柔軟にタイルの種類の追加ができるようにした。このとき、視覚班が確認して作業しやすいように、StreamingAssetsフォルダを利用して、試しにJSONファイルを読み込んで配置できるようなツールも作成した。

（＊文責：河西叶大）

4.2.1.3 会話エンジン

会話エンジンに使用されるデータクラスTalkDataの構成は以下の通りである。

- Content
    - Speaker
    - Text
    - ChangeImage
        - ImageName
        - SpriteName
    - QuestionData
        - Answer
        - NextFlag
        - NextTalkData
    - BGM
    - SE

Contentはデータクラスの内容をまとめるために用意された、Contentクラスのオブジェクトである。Speakerは発話者を指定するための変数である。Textは内容を記述するための変数である。ChangeImageは画像指定のためのオブジェクトで、内部に画像を表示する領域を指定するImageNameと、画像を名称で指定するSpriteNameを持つ。QuestionDataは画像指定のためのオブジェクトで、選択肢の回答を（はい,いいえなど）記述するAnswerと、変更するフラグの名前と変更後のフラグの値（true, false）を指定するNextFlag、選択肢によって変更される次のテキストJSONを指定するNextTalkDataを持つ。BGM、SEは音声ファイルの名前（拡張子なし）を指定する変数である。

（＊文責：小林優斗）

4.2.2 ミニゲーム

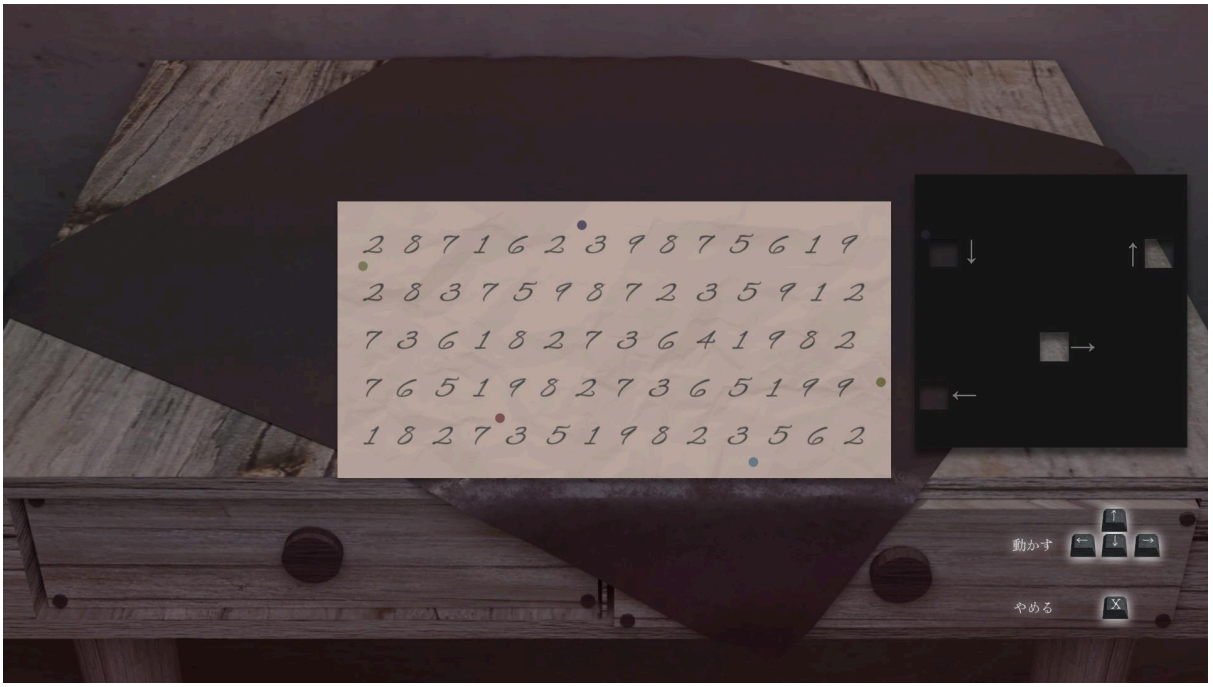
4.2.2.1 サーチゲーム

マップ上の特定の場所を調べることで開始するミニゲームである。マウスカーソル移動や方向入力で赤い点のカーソルを移動でき、画面中のキラキラに重ねるとカーソルが虫眼鏡に変化する。この状態の時にマウスの左クリックやZキー入力を行うと、アイテムを入手することができる。

（＊文責：井田和樹）

4.2.2.2 重ね合わせゲーム

謎の数字が書いてある一枚の動かないイラストと、動かせる一枚の穴が開いたイラストが表示される。背景イラストと穴が開いたイラストを正しく重ね合わせると、意味のある文章が浮き上がり、ストーリーが進行できるようになる。



（＊文責：小松知佳）

4.2.2.3 タイミングゲーム

概要

タイミングゲームとは、決定キーを押しスライダー上端を上昇させ、離して上昇を止め、基準となるタイミングを示す横線（以下、タイミングバー）に合わせるゲームである。ゲームのシナリオにおいて、薬の調合を行う場面があり、薬の調合をゲームとして表現するために作成した。また、タイミングゲームを繰り返し行うことで、薬の調合のうち、材料の計量を表現することに焦点を絞った。

画面の構成

現状のタイミングゲームの画面には、タイミングバーと、上に伸びていくスライダーがある。タイミングバーはスライダーと紐付けされており、スライダーの上端と下端を超えない範囲で自由に配置できる。また、タイミングバーのy座標はスライダーの下端から計算される。スライダーが上に伸びていく理由について、薬の調合を表現するために、スライダーを試験管に、スライダー上端の上昇を注ぐことに見立てている。

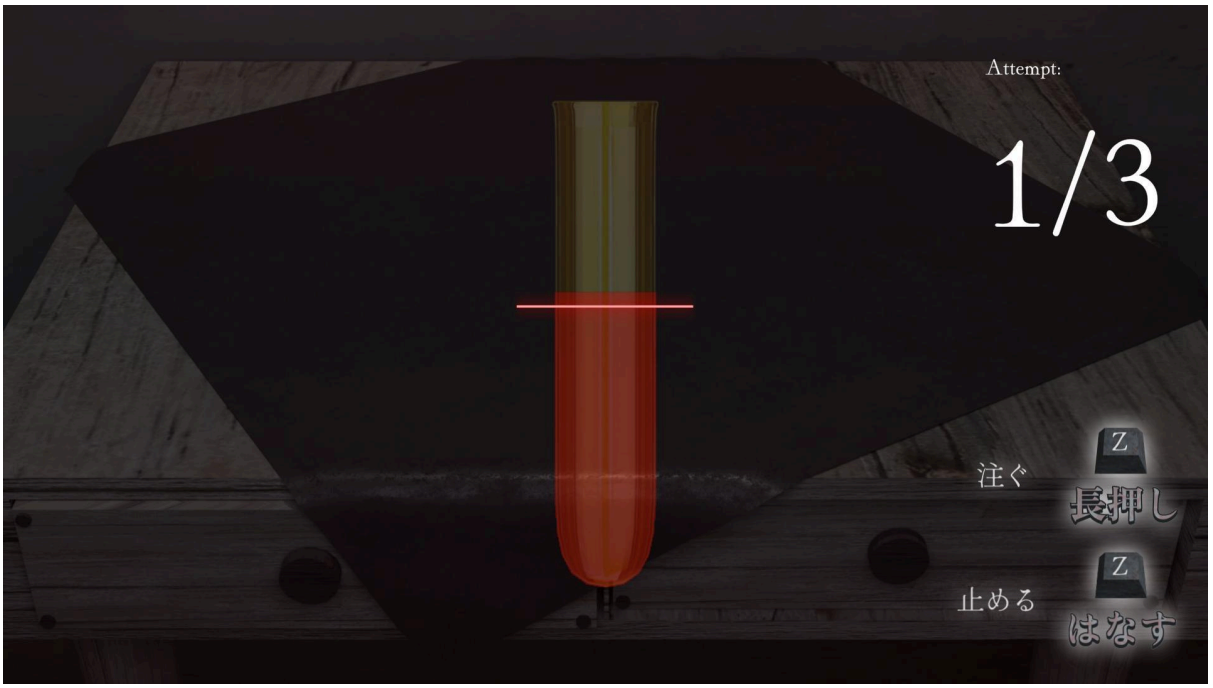
作成したクラス

タイミングゲームを成立させるため、今回作成したクラスに「TimingGameManager」「TimingGame」「TimingSlider」「TimingGameJudge」「TimingGameItem」がある。TimingGameManagerには、ゲームの進行上必要なフラグ操作を行う機能がある。TimingGameには、タイミングゲームの初期化や、タイミングゲームの終了処理、タイミングゲームのスコアを保存する機能があり、タイミングゲームを繰り返す回数やタイミングの判定幅を決める変数がある。TimingSliderには、スライダーを上に伸ばす処理や止める処理があり、スライダーの上昇速度を決める変数がある。TimingGameJudgeには、タイミングの判定をする機能がある。TimingGameItemには、ゲームの進行上必要なアイテムの変更が含まれている。

タイミングの判定
　タイミングバーのy座標を、スライダーの上昇スピードで割ることで、基準となるタイミングにスライダーの上端が到達する時間を、基準となるタイミングとしている。また、スライダーを止めた際に「Great」「Good」「Bad」の三種類の精度にかかわる判定を行う。現状「Great」「Good」間の判定幅は50ミリ秒、「Good」「Bad」間の判定幅は150ミリ秒である。

タイミングゲームの終了処理
　タイミングゲームの試行回数が、繰り返し回数を定めた変数未満であるとき、タイミングゲームの初期化を行う。一方、タイミングゲームの試行回数が、繰り返し回数を定めた変数以上になったとき、タイミングゲームのスコアの計算と保存を行い、他のシーンに遷移する。

タイミングゲームのスコア
　タイミングゲームの出来によって、シナリオに変化をもたらしたいという要望があったため実装した。現状、具体的な数値はプレイヤーに示されない。スコアは基準となるタイミングとのずれの絶対値を試行ごとに記録し、それらの記録の平均値をスコアとする。このようにした理由は、タイミングゲームの繰り返し回数の変更に強い点とゲーム性の向上が挙げられる。繰り返し回数の変更に強いとする理由は、繰り返し回数が未定であることと、単に失敗回数を記録すると繰り返し回数を変更するたびに、その都度ソースコードを書き換える必要があるためである。ゲーム性の向上の理由は、単に失敗回数を記録すると、大きい失敗も惜しい失敗も一つの失敗としてカウントされてしまい、薬の調合の表現とかみ合わず、ゲームの説得力が薄れる可能性があるためである。



（＊文責：小林優斗）

4.2.3　セーブ機能

　セーブ機能はゲームの状態を保存する機能である。このゲームで保存する情報は以下の通りである。

- フラグ情報：ゲームの進行によって変化するフラグの状態。
- アイテム所持情報：アイテムの獲得状況。
- シーン情報：前回のプレイで最後に居た場所。

私たちはセーブデータの保存に二つの方法を使った。一つはUnityで用意されているPlayerPrefsである。PlayerPrefsは手軽に使用できるため、主にシーンの情報を保存するのに使用した。もう一つはMessagePack for C#[2]である。これによってファイルに大量のデータをクラスの情報として復元可能な形式で保存できるため、フラグ情報とシーン情報の保存に使用した。保存したフラグの種類は以下の通りである。

- 進行度フラグ：ストーリーがどのくらい進んだかを表すフラグである。ゲーム中での第1章、第2章などで大きくストーリーを分割する。後述の経験フラグ等に同じ役割を持たせることもできるが、開発者のわかりやすさ向上や、論理演算の回数削減のために設ける。また、ストーリー分岐したときにはそれぞれの分岐先でフラグ別のものとする。
- ミニゲーム内フラグ：ミニゲーム内の情報管理に利用するフラグである。専用の画面に遷移するミニゲーム以外にも、ギミックに作用するマップ上のスイッチのON/OFFの状態の表現にもこのフラグを用いる。
- アイテムフラグ：アイテムを現在所持しているかのフラグ。アイテムの所持状況によって会話内容が変化し、ストーリーが進行することが考えられるので必要である。アイテムリストを参照することで、アイテムフラグの値を確定する。
- 会話回数フラグ：主人公が話しかけた回数をキャラクターごとに記録するフラグである。話しかけた回数によって会話内容を変化させることを想定しているため、必要である。
- 経験フラグ：ある部屋に入ったこと、ある家具を調べたこと、ある人の話を聞いたこと、あるミニゲームをクリアしたことなど、主人公が経験したことを表現するフラグである。この特性上、値がtrueからfalseになることは想定していない。
- 好感度フラグ：あるキャラクターの好感度を整数として表現するフラグである。このフラグが必要になる、ストーリー上で好感度が変化するキャラクターは1人のみである。主人公が好感度が上がる行動を取れば数値が増え、好感度が下がる行動を取れば数値が減る。

（＊文責：河西叶大）

4.2.4 音楽・効果音の機能

SoundManagerの機能

本報告では、SoundManagerの機能と音量調整に関する詳細について説明する。SoundManagerは、ゲーム内のバックグラウンドミュージック（BGM）およびサウンドエフェクト（SE）の管理を行う重要なコンポーネントである。

AudioMixerの利用とグループ分け

音量をBGMやSEなどにグループ分けして操作するためには、UnityのAudioMixer機能を利用する必要がある。AudioMixerを作成し、Master、BGM、SEという3つのグループを定義した。Masterグループは、BGMとSEを総括した全体の音量バランスを管理するためのグループである。この設定により、全体の音量バランスを一括で調整することが可能となる。

各グループのVolumeパラメータをScript側に公開することで、スクリプトから音量を操作できるようにしている。具体的には、AudioSourceにMixerの各グループを設定し、ユーザーインターフェース（UI）のSliderの値をMixerに渡すことで、各グループごとに音量を調整できるようにしている。

音量調整パネルの設計と実装
　PanelとAudioMixer、Sliderを使用して、音量調整バーを含む音量調整パネルを作成した。音量調整バーは、Master、BGM、SEの3種類が存在する。音量調整バーは、プレイヤーが直感的に音量を変更できる装置であり、操作性を向上させるための重要な要素である。バーのつまみを右に引っ張ると音量が最大になり、左に引っ張ると音量が最小になる仕組みである。
　視覚班と連携し、音量調整バーの視覚的な設計にも工夫を凝らした。

デシベルから音圧への変換
　音量調整を滑らかに行うためには、デシベル（dB）を音圧に変換する設定が必要である。UnityのAudioMixer機能から取得できるVolumeは、デシベルとして表現されており、これは音圧を常用対数で変換した値である。このままでは、Volumeの値が大きい領域では音量が急激に変化し、値が小さい領域ではつまみを操作しても音量の変化が感じられないという問題が生じる。これを解決するために、デシベルを音圧に変換する設定を行った。これにより、音量調整がより滑らかで直感的に行えるようになった。

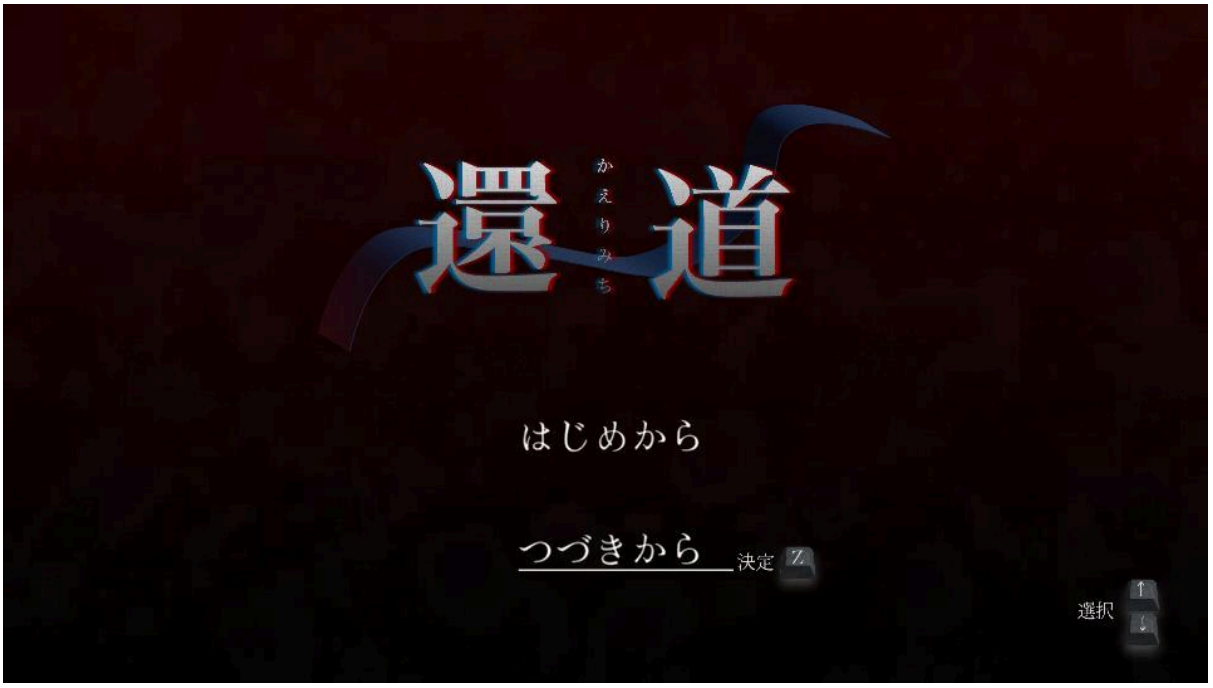
各シーンでの活用
　各シーンでSoundManagerの機能を用いてBGM、SEを再生した。また、ゲームプレイ中のいかなる場面でも音量調節パネルを開いて音量調節を行うことができる。

（＊文責：小松知佳）

4.2.5　タイトル画面

　タイトル画面とはゲームの最初に表示される画面である。タイトル画面には、「はじめから」と「つづきから」の項目と項目を選択するためのカーソルが存在する。「はじめから」を選択すると、セーブ機能によって保存された情報をすべて消去し、初期値を代入して、最初のマップがあるシーンに遷移する。「つづきから」を選択すると、セーブ機能によって保存された情報を読み込み、適切なアイテムの所持とシーンの遷移を行う。セーブデータが存在しないときは「つづきから」

が表示されず、セーブデータが存在するときは、「つづきから」にカーソルがあっている状態でタイトル画面が表示される。



（＊文責：小林優斗）

4.2.6 キャラクター操作

キャラクター移動機能はキャラクターをプレイヤーのキー入力によって操作し、画面上の場所を移動する機能である。まず、キャラクター移動機能の作り方は二つの方法を考えた。
一つ目は、キャラクターの位置を入力された方向のベクトルを加算したものに一定の速度で更新する。というものである。この方法では、UnityのTransformのpositionを変更して移動を行う。二つ目はキャラクターに物理特性を設定し、入力された方向ベクトルの速度を設定する。というものである。この方法ではUnityのRigidbodyのvelocityの値を入力があったときに変更することで実装した。

最初は、毎フレーム位置を変更する処理をRigidbodyに委譲することができるという理由から二つ目の方法を採用してキャラクター移動機能を実装した。ところが、メニュー画面を表示させているときにキャラクターの移動機能を無効にしなければならなくなり、メニュー画面が表示されたときに速度を0にする実装を追加しなければならなかったことから、一つ目の方法で再度実装した。

当たり判定はキャラクターに追加したUnityの機能であるBoxColliderとUnityのイベント関数であるOnCollisionEnterを使用して衝突を検出し、特定の種類の障害物が移動先に存在した場合は移動前の場所に戻すようにした。通常は障害物の内部にキャラクターのコライダーが入ったときにOnCollisionに関するイベント関数が呼ばれるが、BoxColliderのoffsetを1フレームで移動する距離だけ入力された方向のベクトルに設定することで、障害物の手前で衝突を検出し、元の場所に戻ることができるようにした。
また、画面外に出るバグが多かったため、対策として、キャラクターの進行方向にあるオブジェクトが画面外であったときに移動先の座標を画面内に収めるように実装している。

最終的なキャラクター移動の流れ

まず、キー入力を毎フレーム受け取る。入力を受け取ったら入力に応じた方向ベクトルを記録し、移動先のタイルにキャラクターが移動するまでキャラクターの座標に記録した方向ベクトルを加算し続ける。このとき、移動が終わるまで入力を受け取らないようにする。また、移動中に障害物に衝突したらキャラクターを移動前の位置に戻しキー入力待ちに戻る。そして、移動先のタイルにキャラクターが移動したらキー入力待ちに戻る。

操作のキー割り当て
移動：上下左右キー,（WASDキー）
　　マップ上のキャラクターの移動や、メニュー画面でのカーソルの移動などを行う
決定：Zキー,（Enterキー）
　　ボタンを押したり、マップ上のオブジェクトを使用する
キャンセル：Xキー,（Escキー）
　　1つ前のメニューウィンドウに戻る。ホームウィンドウで押すとマップ移動画面に戻る
メニュー開閉：Cキー,（Spaceキー）
　　マップ移動中に押すと、メニューが開く。メニュー画面で押すとすぐマップ移動画面に戻るため、Xキーでホームウィンドウまで1つずつ戻る必要がない。

割り当て理由
　このキー割り当ては、基本的には右手で移動、左手で決定、キャンセル、メニュー開閉ができるように配置している。また、用いるキーを左右それぞれで隣接したキーにすることで、プレイ中に手を大きく動かさずに操作できるようにしている。また、「青鬼」や「魔女の家」などの、本ゲームと同じ見下ろしホラーアドベンチャーゲームでもほぼ同じキー割り当てを採用していることも、この割り当てにした理由の1つである。以下は「魔女の家」の操作方法である。[4]

キャラクターの移動　　：　カーソルキー（↑／←／→／↓）
決定　　：　Zキー、Space、Enter
キャンセル、メニュー表示　：　Xキー、Esc、Num 0
ダッシュ　　：　Zキー、shiftキー

　それとは別に、PCゲームでのメジャーな移動方法であるWASDキーを用いた操作にも対応している。

（＊文責：河西叶大）

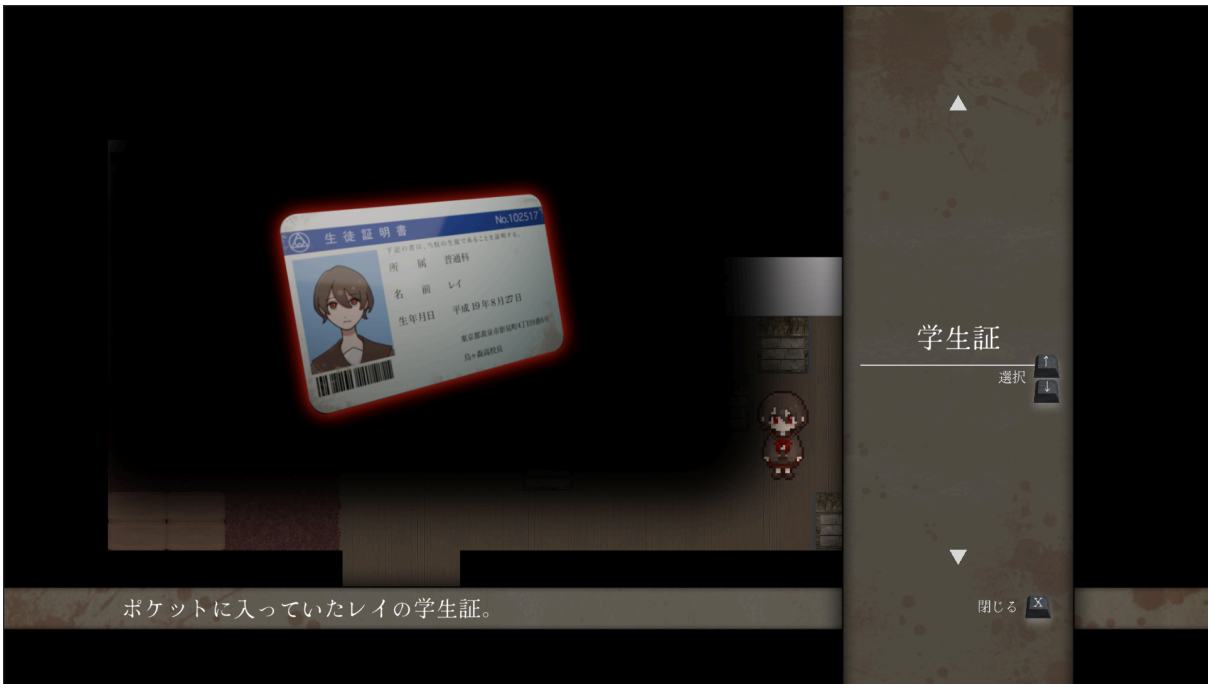
4.2.7　メニューUI

　メニューUIとはゲーム中に起動できる、所持しているアイテムを一覧表示するUIである（下部の画像参照）。これはゲーム中にCキーかスペースキーで起動でき、Cキー、スペースキー、Xキーのいずれかを押すと閉じることができる。次に、メニューUIの機能について説明する。画面右側の縦長のエリアには、現在所持しているアイテムを一覧表示できる。アイテムを複数持っているときは、上下入力でカーソルを移動できる。5つ以上のアイテムを所持している場合は、方向入力を続けることでアイテムの一覧をスクロールすることができる。また、画面下部の横長のエリアに、現在カーソルを当てているアイテムの簡単な説明文を表示している。また、画面左上の大きな黒い範囲には、現在カーソルを当てているアイテムの見た目を表す画像を表示する。また、本などの一部のアイテムは、カーソルを当てているときにZキーかEnterキーを押すことでその本に書かれている内容をテキストウィンドウで表示することができる。

アイテムはゲームプレイ中に拾ったり、会話中に入手することができる。アイテムは、以下の情報を持つ。

- 名前
- 説明文
- 画像
- （任意）アイテム本文のテキストJSONの保存場所

また、一部のアイテムは2つを組み合わせて1つの新しいアイテムに合成することができる。この合成は、材料となるアイテムを両方入手したタイミングで自動的に行われる。また、ゲーム内部ではアイテムとは別にアイテム合成時のレシピ情報を持っており、それを参照して合成が行われる。



（＊文責：井田和樹）

4.2.8 テキストウィンドウ

概要

変数で指定されたテキストファイル（.json）または文字列とそのリストを読み込み、文章を一文字ずつ表示する。テキストファイルで指定された際は、発話者欄、背景画像、選択肢、音声出力等も行う。決定キーで次の行に移行することができ、行のすべての文章が表示されていないとき、決定キーで行の最後まで飛ばすことができる。テキストを表示するオブジェクトのTextMeshProの機能として文章を装飾できる。例えば、太字、斜体、下線、文字色変更が挙げられる。また、ルビを振る機能として、オープンソースのもの[5]を使用した。加えて、テキストウィンドウのシステムについてはすでにメンバーが作っていたものがあり、それを再構築する形で開発した。

目的

アドベンチャーゲームにおいて文章を表示する機能が必須であるからである。

画面構成

現状のテキストウィンドウの画面には、テキストの本文を表示する部分と発話者を表示する部分、テキストの行が変更可能であることを示すアイコンがある。また、目視できない要素として、背景やキャラクターの立ち絵を表示する領域がある。

作成したクラス

テキストウィンドウを成立させるため、今回作成したクラスに「ConversationTextManager」「MainTextDrawer」「NameTextDrawer」「ChangeBackground」「Question」がある。ConversationTextManagerはテキストウィンドウの制御をする役割があり、初期化や行数の操作、テキストファイルで指定された内容に合わせて各種機能を呼び出す機能がある。MainTextDrawerは本文を表示する役割があり、本文を一文字ずつ表示する機能、行の最後まで飛ばす機能、ルビを振る機能を呼び出す機能、テキストの行が変更可能であることを示すアイコンを制御する機能がある。NameTextDrawerは発話者を表示する役割があり、発話者を表示する機能と発話者を表示する部分を制御する機能がある。ChangeBackgroundは背景やキャラクターの立ち絵を変更する役割があり、画像を変更する機能がある。Questionは選択肢の表示や制御を行う役割があり、選択肢の表示、カーソル移動、カーソル位置出力の機能がある。

行数の操作

入力されたものがテキストファイルのとき、その内容を会話エンジンで規定されたデータクラスであるTalkDataに変換しTalkDataの要素であるContentの配列として保存する。一方、入力されたものが文字列とそのリストのとき、Contentの配列であることは同じであるが、Contentの要素であるTextに代入される。行数の操作はこの配列のインデックスを増減させることで行う。具体的には、決定キーが押されたときに配列のインデックスを増やす。また、リストのインデックスが範囲外になることを防ぐため、最後の行のとき配列のインデックスは増やせないようになっている。

本文を一文字ずつ表示する機能

TextMeshPro内に表示できる文字数を指定するmaxVisibleCharactersという変数があり、この変数を毎フレーム増加させることで、一文字ずつ表示する機能を実現している。また、ConversationTextManagerで渡された配列のうち、最後の要素を表示する。一方、行の最後まで飛ばす機能は、行の文字数をmaxVisibleCharactersへ代入することで実現している。

アイコンを制御する機能

本文がすべて表示されていないときは、アイコンを表示しない。一方、本文がすべて表示されたときは、本文の最後の文字の位置を取得し、アイコンをその位置に表示する。

各種機能を呼び出す機能

TalkDataの要素であるContentに含まれる要素として、「Text」「Speaker」「ChangeImage」「QuestionData」「BGM」「SE」がある。TextはMainTextDrawer、SpeakerはNameTextDrawer、ChangeImageはChangeBackground、QuestionDataはQuestionに渡される。BGMとSEはSoundManagerを呼び出す。

NameTextDrawerが持つ機能

ConversationTextManagerから渡されたSpeakerの要素を表示する。Speakerがnullのときは、発話者を表示する部分を非表示にする。

ChangeBackgroundが持つ機能

ConversationTextManagerで渡されたChangeImageを元に画像を変更する。ChangeImageがnullのときは、画像の変更を行わない。

Questionが持つ機能
　ConversationTextManagerで渡されたQuestionDataを元に選択肢を表示する。ChangeImageがnullのときは、選択肢を表示しない。カーソル移動、カーソル位置出力については、Questionクラスの要素として存在するcursorPlaceを増減や出力することで実現している。

（＊文責：小林優斗）

4.3　成果物の統合

開発の流れ
　システム班では、リーダーが各班員ごとに作業を割り当て、機能毎にブランチを作成して、各メンバーの成果物はメンバーの担当機能毎に新しくシーンを制作し、そのシーン上に実装することで機能を制作した。機能が完成したらメンバーはPull Requestを作成し、リーダーがそれを確認してコードの修正や、レビューを行うというやりとりを繰り返して本流のブランチに統合しながら開発を進めた。プロジェクトの終盤では機能開発だけでなく、既存機能のバグ修正も同様の流れで行った。

Githubの活用
　このとき、シーンやPrefabのオブジェクトのスクリプトの参照がMissingになっているものが含まれて、統合した後にWarningが発生して各メンバーが修正してしまい、conflictが発生してしまうという出来事が度々発生していたため、game-ciのtest runner[3]を使用してスクリプトの参照がMissingになっているものを検出し、ブランチを保護する仕組みを構築した。

発生した問題
　他の機能と組み合わせて利用することが想定されていなかったために、問題が生じた。例えば、Unityのシーン開始時に呼び出されるStart関数から機能が呼ばれるように設計されており、外部から機能を呼び出すことができなくなっていた。また、機能を呼び出すキーが競合し、別のシステムが呼び出されるという問題が発生した。また、UIの解像度が統一されておらず、解像度を変更した際に表示が崩れる問題が発生した。

解決策

キーが競合する問題についてはPause機能を実装することで解決した。Pause機能では、Pauseコンポーネントに機能を停止させたいBehaviourを継承したスクリプトを設定することでまとめて機能を停止、再開させることができる。この機能を使ってスクリプトを制御し、キーの競合問題を解決した。外部から機能を呼び出すことができなくなっていた問題については、機能を作成したメンバーと相談しながらスクリプトを修正することで解決した。UIの解像度が統一されていない問題については、Canvasをそのまま利用し、Canvasのオブジェクト本体のアクティブ・非アクティブを切り替えることで解決した。他のシーンで作成したオブジェクトはPrefabという他のシーンでも利用可能な形式に変換し、移動して利用した。

（＊文責：河西叶大）

4.4 他班との連携

システム班では主に、開発物の仕様に誤解がないか、こまめに視覚班や伏線班と直接会話して確認を行った。11月下旬からは、各システム班員がミニゲームやUIなどの各分野の担当者となることで、情報がまとまるようにした。

（＊文責：井田和樹）

5章：結果

システム班は人工知能で生成された他班の成果物を統合し、ホラーアドベンチャーゲームとして完成させることを目標として開発を行った。そのために、Unityを用いて開発を行った。本プロジェクト学習で得られた成果は、一つのホラーアドベンチャーゲームである。このゲームの中には前述したミニゲームが含まれている。現時点ではゲーム進行に関わるバグが残っているため、今後もデバッグが必要である。

（＊文責：小松知佳）

6章：考察

Unityの学習

前述したように、Unityの学習を早期に始めたことは前期の活動において有益であった。特にテストコードの作成は、技術力・品質向上に大いに役立った。このテストコードの利用は、来年度のチームメンバーの技術力向上にも役立つと考えられる。来年度のチームメンバーの技術力に沿ったテストデータの更新を行うことで、更なる技術力・品質向上が期待できる。

班内進捗会の効果

班内進捗会を定期的に開催したことは制作に役立った。成果物作成中は、各メンバーが1人で作業に没頭しがちであり、これは班内のコミュニケーション不足につながるリスクがある。これを防止するために、週に2回、進捗報告の時間を確保した。班内進捗会の効果として、以下の点が挙げられる。

- メンバーが問題を抱え込まず、早期に共有できた。

- 班全体の課題を迅速に発見し、対策を講じることができた。
- メンバー間のコミュニケーションが活性化し、作業効率が向上した。

他班とのコミュニケーション不足とその改善策

一方で、他班とのコミュニケーション不足は深刻な問題であった。他班との連携が不足することで、以下のような問題点が生じた。

- ゲームの全体像を把握しづらくなった。
- タスクを効率的にこなすことが難しくなった。
- スケジュールが不透明になり、計画の作成が困難であった。

これらの問題を改善するために、システム班から以下の二つの提案を行った。

- ゲームプランナーという役職を立てること。
- リーダー会議を定期的に開催すること。

ゲームプランナーについては人手不足のため見送られたが、リーダー会議は6 月19 日より実施することとなった。リーダー会議の実施により、以下の改善が見られた

- 他班とのコミュニケーションが活発になり、情報共有がスムーズに行われるようになった。
- ゲームの全体像を把握しやすくなり,タスクの効率的な遂行が可能になった。
- スケジュールの透明性が向上し、計画の遅延が減少した。

しかし、計画の遅延を完全に防ぐことは不可能であった。これは、以下の原因が挙げられる。

- ゲームプランナーをプロジェクト発足時に立てられなかったこと
- 他班への締切設定をシステム班が担うことが遅れること
- デバッグ作業を行う人員が確保できなかったこと

以上の経験から、ゲーム開発プロジェクトにおける適切な進め方を考察する。計画の遅延を防ぐためには、プロジェクト発足時、リーダーとは別にプロジェクトを統括するゲームプランナーを立てることを推奨する。ゲームプランナーを立てることが困難な場合は、システム班主導でプロジェクトを牽引する必要がある。そうすることで、デバッグ人員を確保することにもつながる。

（＊文責：小松知佳）

参考文献

[1] 大柳 裕士, 田中 良磨, 森 正樹, 中村 慶琉, 村岡 広海, 松下 瑚南, 山崎 雄太, 関谷 天希, 佐藤 玲, 時永 空侑, 西口 優太朗, 尾﨑 陽彦, 鈴木 壱, 長瀬 悠太, 山本 拓摩, 迎山 和司, 田柳 恵美子, 平田 圭二, 吉田 博則, 村井 源, 複数のAI自動生成技術を組み合わせたループ型2Dアドベンチャーゲームのシステム開発, 第38回人工知能学会全国大会, 1I4-OS-31a-03, 2024.
[2] GitHub - MessagePack-CSharp/MessagePack-CSharp: Extremely Fast MessagePack Serializer for C#(.NET, .NET Core, Unity, Xamarin). / msgpack.org[C#] https://github.com/MessagePack-CSharp/MessagePack-CSharp (2025/01/10アクセス)
[3] Unity - Test runner · Actions · GitHub Marketplace · GitHub https://github.com/marketplace/actions/unity-test-runner (2025/01/10アクセス)
[4] 魔女の家 (ver1.08)「ビデオゲーム」(2012) . ふみー
[5] GitHub - ina-amagami/TextMeshProRuby: TextMeshProで漢字テキストにルビを振るコンポーネント https://github.com/ina-amagami/TextMeshProRuby (2025/01/10アクセス)