

Rationale

Vol.39



目次

顧問挨拶		01
代表挨拶		02
—活動報告—		
小幡充輝	ロボティクスコース 3 年 「ファミコン風の曲の制作」	03
村上陸	マテリアル環境コース 3 年 「Unity で玉転がしゲームの作成」	05
黒澤徹真	ロボティクスコース 3 年 「生成 AI を使用したゲーム制作」	09
	「インド風の音楽を作る」	11
編集後記		13

フラグも技巧的

ソフトウェア研究部会 顧問 北島宏之

昨号でメモリの話題を提供した。その最後に「フラグが立つ」とは？とのクイズを出していたので、今回はフラグについてお話ししたいと思う。

「フラグ」とは「flag」、つまり原義は旗である。計算機の中では、何らかの処理がなされた際に結果を示す短いデータを生成する。その生成されたデータ、もしくはそのデータを格納している領域がフラグである。場所としては、メインメモリ内や CPU 内部のレジスタ内などであるが、多くの場合、1 フラグは 1 ビット(bit, 2 進数 1 桁)のデータであって、数値的には 0 か 1 を使用する。

例えば、何かの処理を発生させる場合と発生させない場合を、1 と 0 で表すなどである。具体的には、RPG(Role-Playing Game, ロールプレイングゲーム)で、主人公が A というアイテムを集めると、祠 B の扉が開く、といった場面に利用される。この場合、A アイテムフラグの初期値を 0 としておき、主人公が A アイテムを発見して所持することにより、A アイテムフラグを 1 にセットする。その後、主人公が祠 B の前にたどり着いた際、A アイテムフラグの値を確認し、1 になっておれば祠 B の扉をゴゴゴゴッと開き、0 のままであれば祠 B の扉は無反応で開かない、といった処理である。

ここで、フラグが 1 ビットで扱えることが大きいというか、1 ビットで扱っていることこそ技巧である。これは、複数の条件の組み合わせであっても、データ数が非常に少なく済むからである。例えば、8 条件の組み合わせであっても 8 ビット=1 バイト(Byte, 2 進数 8 桁)であるし、8 ビットであれば符号なし 10 進数の 0~255 が表現できることから、プログラム上も 10 進数値の場合分けとして分岐後の様々な処理が実装できる。

話は戻るが、「フラグ」は、ゲームにおいてシナリオやステージの分岐条件の記録に用いられることから、そのまま自然にゲーム界限で用いられるようになった。さらに派生し、ゲームのみならず、小説や漫画、映画などのストーリーにおいて、お決まりのパターン通りの展開を予期させる「前振り」事象をフラグと呼ぶようになった。例えば、某有名漫画でゴブタが「自分が行ったときは門の前で絡(から)まれたっすけど」とボソッと言うと、主人公のリムルが「なんか今、盛大にフラグが立った気がする」と、その後に門の前で実際に周囲から絡まれ「はい、...絡まれました、フラグ回収です」、といった具合である。

以上、フラグの技巧でした。

代表挨拶

部会長挨拶

ロボティクスコース 3年 黒澤徹真

本部会は、コンピュータやソフトウェアに関する探究を目的として日々活動を行っております。昨年度まではプログラミングコンテストへの出場を一つの目標として取り組んでまいりましたが、今年度は手違いにより、残念ながら大会への参加は見送ることとなりました。しかし、部員一人ひとりがそれぞれの関心をもとに学びを深めており、個人の研究や開発を中心とした活動を続けています。

ソフトウェアの世界は日進月歩であり、新しい技術や考え方が次々と登場しています。そうした変化の中で、自ら課題を見つけ、学び、形にしていくことこそが本部会の目指す姿であると考えています。成果の大小にかかわらず、各自が挑戦を続けることに意義があり、その積み重ねが確かな力となっていくと信じています。

今後も部員一同、創意と探究心をもって活動を重ね、互いに刺激し合いながら成長していけるよう努めてまいります。

副部会長挨拶

ロボティクスコース 3年 小幡充輝

Rationale を手に取っていただき、ありがとうございます。私がこの部活に入り3年が経ちました。現在、ソフトウェア研究部会は個人活動に努めている部員が多く在籍しています。その一方で、先輩方がいた過去に比べ、組織としての意識が薄れているように感じます。来年度以降、これまで部員それぞれが培ってきた技術を一つの目標に向け、集約させる活動を行っていきたいと考えております。最後に、顧問の北島先生、Rationale の刊行に協力してくださった部員、そして、今Rationale をこうして手に取ってくださった方々に感謝します。どうかこれからもよろしくお願いします。

ファミコン風の曲の製作

～8bit Music をつくろう～

ロボティクスコース 3年 小幡 充輝

1. はじめに

今回私は作曲ソフト「Cubase」を使用して、ファミコン風の曲の製作を行いました。グレードは Artist で、バージョンは14.0.32です。実際に制作した曲はこちらになります(図1)。



図1.曲の QR コード

2. 製作の流れ

2.1. 曲の構成を考える

最初に、曲の構成とは何でしょうか。曲にはイントロ、A メロ、B メロ、C メロ、間奏、アウトロなどの構成要素があります。この構成要素のどれを、どの順番で曲を組み立てているのか。これが曲の構成です。今回曲の構成を考えるにあたり、あまり複雑でなく、わかりやすい曲を製作したいので、イントロ→A メロ→B メロ→サビの構成で製作しました。

2.2. シンセサイザーで音を作る

ファミコンのピコピコした音は、シンセサイザーと呼ばれる機械が作っています。なので、パソコン上でシンセサイザーを鳴らせる Magical 8bit Plug2というソフトを Cubase(作曲ソフト)の中にダウンロードし、音を作りました。まず、

メロディの音を作ります。メロディの音は一番印象に残ってほしいのではっきりと聞こえる短形波で音を作りました(図2)。シンセサイザーには Attack、Decay、Sustain などのパラメータがあり、パラメータの数値を変えることで様々な音を作ることができます。メロディ用のシンセサイザーは Decay の数値をやや大きくすることで伸びやかなサウンドにしました。さらに、Sustain の数値を大きくすることで音の鳴る持続時間を長めにしました。

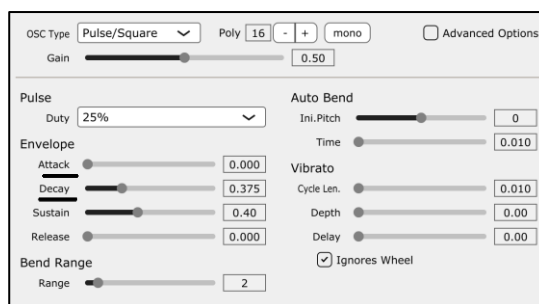


図2. メロディシンセのパラメータ

つぎにベースの音を作っていきます。今回の曲におけるベースの役割はメロディを支えることなのであまり目立たない三角波を使って作りました(図3)。Attack のパラメータを少しだけ大きくし、音の立ち上がりを少し遅らせることで自然に聞こえるようにしました。

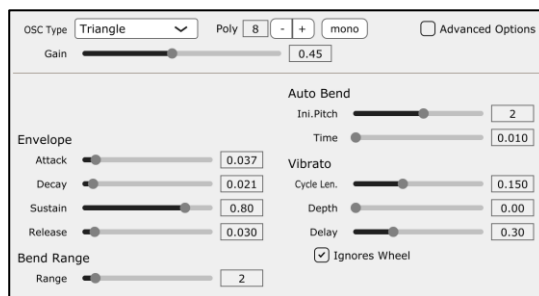


図3. ベースシンセのパラメータ

最後に、ドラムの音を作っていきます。ファミコンでは、ドラムはノイズと呼ばれる波形を使って作られています。今回制作した曲ではパラメータは同じで、音程と聞こえてくる方向を変えた2つのドラムの音をシンセサイザーで作っています(図4)。Decay の値を少しだけ小さくしてアタック感を強調し、歯切れのよい音にしています。そして Sustain の値を0にしスパッと音が消えるようにしています。

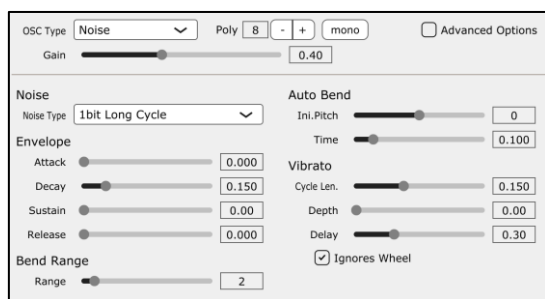


図4. ドラムシンセのパラメータ

2.3. 作った音を組み合わせる

最後の工程で、作った全てのシンセサイザーを組み合わせ、一つの楽曲にします。各パートの音量のバランスを調整し、意図していない部分が目立っている場合、目立たない音色に変更したり、音を鳴らすタイミングを合わせたりして修正します。これで曲は完成です。

3. 動画編集

この工程は曲の製作に直接関係する部分ではないのですが、製作した曲をどのように外部に公開するのか書いておきたいと思います。まず Canva という動画編集ソフトをダウンロードし、Google Gemini に生成してもらった画像をドット絵変換ツールに取り込んでドット絵に変換します。そして Canva を使って、制作した画像と曲を一つの映像にして mp4 形式に書き出

します。最後に、製作した動画を自分の YouTube チャンネルに投稿します。これで製作した曲を外部に公開することができました。

4. おわりに

はじめてすべての楽器をシンセサイザーで作ってみて、シンセサイザーに対する関心はかなり高まりました。ですが、本気でシンセサイザーについて理解しようとすると、覚えることがけっこう多く、複雑な印象を受けました。作っている間は感覚でなんとかなるので楽しいのですが、シンセサイザー自体に対する知識や音を作る際の小技はまだまだ習得できていない部分が多いので、これからもこういった曲を作っていきたいと思います。

参考文献

8bit(ファミコン風)ゲーム音楽の作り方。仕様を理解し正しく打ち込みましょう。

<https://wingless-seraph.net/wp/>

Magical8bitPlug でドラムを作ろう！

<https://ele-create.com/>

Unity で玉転がしゲームの作成

マテリアル環境コース 3年 村上 陸

1. はじめに

今回私は Unity2022 と Visual StudioCode を使用し、簡易的な玉転がしゲームを作成しました。

このゲームは球状の Player を操作し、ステージ内に設置されたアイテムをすべて回収するというシンプルなゲーム性であり、Unity を始めたばかりの私でも一週間程で作成することができたため、Unity を始めてみたいと思っている人やこれから始めようと思っている人には Unity を始める初めの一歩としておすすめです。

2. 使用アプリケーション

2.1. Unity

Unity とは、あらゆる OS に対応できるゲーム開発プラットフォームであり、個人から企業まで様々な場所で使用されています。基本的に無料で使用することができ、独自のシステムで専門知識やスキルがなくてもゲーム開発を始めることができます。

2.2. Visual StudioCode

Visual StudioCode とはマイクロソフトの提供するコードエディタで、だれでも無償で利用でき、その機能性の広さからプログラム開発だけでなく Web 制作、テキスト原稿の執筆・編集にも利用されています。このゲームの作成では C# を用いて「スタートボタン」などに機能を与えるプログラムを書くためなどに利用しました。

3. ゲーム作成

3.1. 全体の流れ

今回はゲームを一から作るために以下の手順でゲーム作成を行いました。

1. ステージ作成



2. プレイヤーの追加



3. 物理演算機能の追加



4. カメラの設定



5. アイテムの設定



6. UI 作成



7. 障害物の作成



8. タイトル、ステージ選択画面の作成



9. シーン間の移動方法の作成

図 1. 作成手順

3.2. コンポーネントとグラフ

コンポーネントとはゲームオブジェクトに機能を追加することができる要素であり、例えば物理演算機能や当たり判定などがあります。グラフとは Visual Scripting の基本的な構成要素であり、プログラミングを使わずゲームの仕組みを視覚的に表すことができます。今回はゲームのリセットや UI 表示などに使用しました。

3.3. ステージ作成

ステージは Unity 上で作成できる簡易的な 3D オブジェクトを任意の大きさに調整し、基本となる土台や壁などを作成し色付けをしました。またこのゲームではステージを 2 つ作成したため別のステージを作成するためシーン (Unity においてゲーム、ステージを表現するための基本単位) を追加しました。

3.4. Player の追加

Player は Unity 上にある球状の 3D オブジェクトを追加しステージの大きさに合うよう調整しました。(図 2)

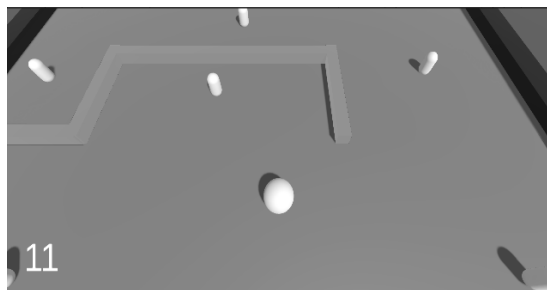


図 2. ステージに配置した Player

3.5. 物理演算機能の追加

追加したプレイヤーを自由に動かすことができるようにするため、物理演算機能を

追加します。

コンポーネントを用いて物理演算の機能を追加し Player がステージや壁をすり抜けないようにしました。次にグラフを用いてキーボードの矢印キーで Player を上下左右操作できるようにしたほか、動く速度を自由に変更できるようにしました。(図 3)

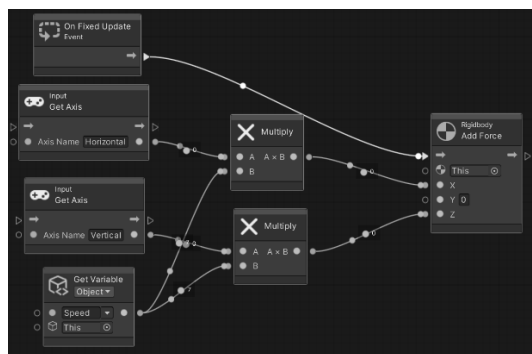


図 3. Player を動かす際のグラフ

3.6. カメラの追加

動き回る Player をカメラが追跡できるよう、グラフを用いてカメラに Player の位置を毎フレームごとに取得し座標を合わせる機能を追加しました。(図 4)

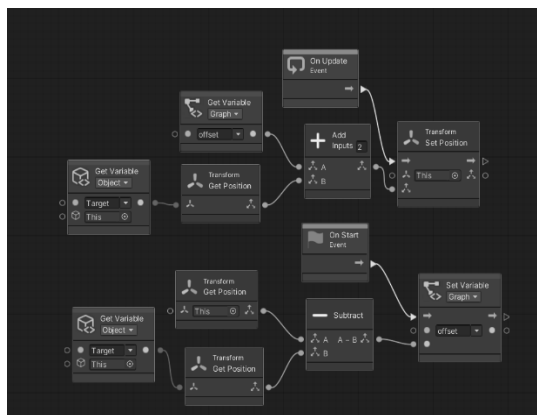


図 4. Player を追跡するためのグラフ

3.7. アイテムの設定

このゲームはステージ上のアイテムをすべて回収することでクリア判定となるため、グラフを用いて Player との接触判定を設定し、そのことを分かりやすくするため接触したアイテムが消えるようにしました。アイテムの形は Unity 上の 3D オブジェクトを変形させ作成しました。(図 5)

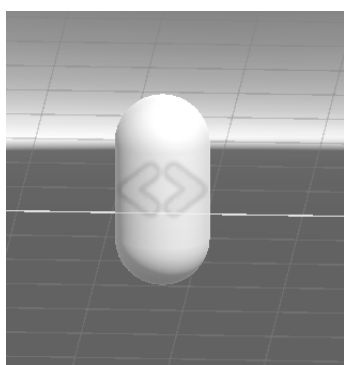


図 5. アイテム

3.8. UI の設定

残りのアイテム数を表示する UI を画面左下に配置し、アイテムが接触によって消えるたびに UI が残りのアイテム数に変化するようにしました。また UI が「0」になった時に画面中央に「YOU WIN」と表示されるようになっています。(図 6)

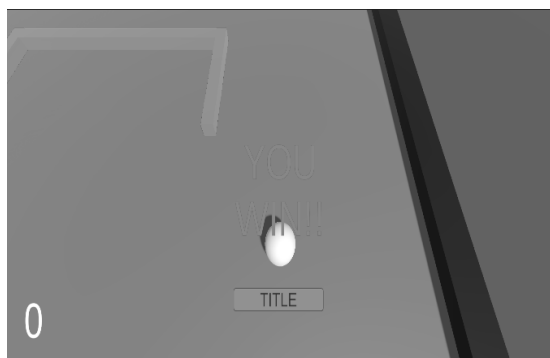


図 6. ゲームクリア時の UI

3.9. 障害物の作成

障害物に当たるとゲームがリセットされスタート時の状態に戻ります。今回は Player と障害物に衝突判定が発生したときにゲームを再度読み込みするようグラフを書きました。また一目で接触してはいけないと分かるよう障害物は赤色で作成しました。

3.10. タイトル、ステージ選択画面の作成

よりゲームらしさをだすためタイトル画面とステージ選択画面を作成しました。どちらも初めに背景を追加しその背景にテキストとシーン間を移動するためのボタンを追加し色付けをしました。(図 7,8) ボタンについては後述します。

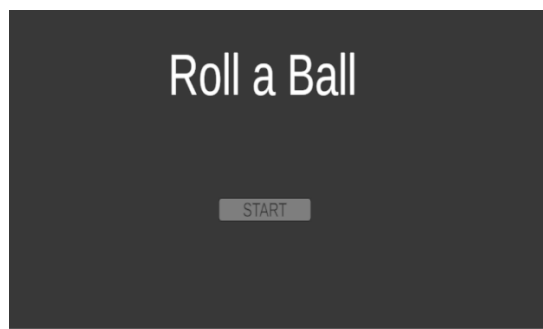


図 7. タイトル画面

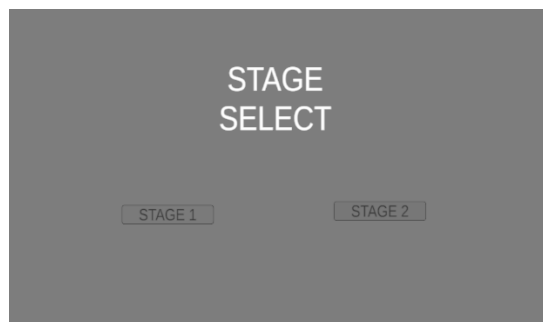


図 8. ステージ選択画面

3.11. シーン間の移動方法の作成

このゲームではシーン間の移動にボタンを用いており、ボタンをクリックすることで現在いるシーンから特定のシーンに移動できるようにしました。

Unity 上で作成したボタンはそのままでは何も機能がないため、Visual StudioCode を用いてシーン間の移動に関するプログラムを作成しました。(リスト 1) このプログラムでは 5 行目に現在のシーン (TITLE) を書き、10 行目に移動したいシーン (stage select) を書くことで、ボタンを押した際にシーンがロードされ移動することができます。また別のシーン間の移動の際はリスト 1 のプログラムの 5, 10 行目のシーン名の部分だけを変更し残りの部分はそのまま使用しています。

リスト 1. シーン間の移動の仕組み

```
1.using System.Collections;
2.using System.Collections.Generic;
3.using UnityEngine;
4.using UnityEngine.SceneManagement;
5.public class TITLE :
6.MonoBehaviour
7.{
8.public void change_button()
9.{
10.SceneManager.LoadScene("stage
11.select");
12.}
13.}
```

4. おわりに

普段はゲームをプレイする側なので今回初めてゲームを作成してみて、今まで当たり前にあると思っていた機能を自分で作成するという体験を通して、ゲームを作ることの大変さを学ぶことができました。また今回はゲームに必要な最低限の機能しか作成することができなかったため来年度以降は、もっとプレイする人を楽しませることができるよう、Unity やプログラミングの勉強に励んでいきたいです。

参考文献

Unity 入門チュートリアル「玉転がし」

<https://learn.unity.com/project/bolt-roll-a-ball-tutorial>

初心者向け！Unity でシーンを切り替える簡単な方法

<https://cbagames.jp/2022/09/10/unity-scene-switch-guide/>

Visual StudioCode

<https://code.visualstudio.com/download>

生成 AI を使用したゲーム制作

ロボティクスコース 3年 黒澤 徹真

1. はじめに

今回は生成 AI を使用してゲームを制作しました。この記事では、その方法と制作したゲームを紹介します。

2. ソフトウェアのダウンロード

まずは生成 AI の代表ともいえる対話型生成 AI サービス「ChatGPT」をダウンロードします。ChatGPT は <https://chatgpt.com> からブラウザで使用またはダウンロードでき、基本プランであれば誰でも無料で利用できます。

また、今回はプログラム言語として「Python」、プログラムの実行・編集用に「Visual Studio Code」も使用します。それぞれ python.org と code.visualstudio.com から無料でダウンロードすることができます。

3. プログラムの作成

各ソフトウェアのセットアップが完了したら、Chat GPT を開き、メッセージを送信します。「Python で動くテトリスを作ってください」というメッセージを送信すると、ChatGPT から400行ほどのプログラムを含む返答がありました(図1)。プログラムのフローチャートは図2のとおりです。



Pythonで動くテトリスを作ってください。

Python Tetris - python コピーする 編集する ダウンロードする **コードを実行する**

```
"""
Simple Tetris implementation in Python using pygame.

Controls:
Left/Right: move
Up: rotate
Z: rotate counter-clockwise
Down: soft drop
Space: hard drop
C: hold
P: pause
Esc: quit

Requirements:
pip install pygame

Save as python_tetris.py and run: python python_tetris.py
"""
import pygame
pygame.init()
screen = pygame.display.set_mode((1000, 1000))
pygame.display.set_caption("Tetris")
clock = pygame.time.Clock()
font = pygame.font.SysFont("monospace", 20)

# Colors
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
RED = (255, 0, 0)
GREEN = (0, 255, 0)
BLUE = (0, 0, 255)
YELLOW = (255, 255, 0)
PURPLE = (255, 0, 255)
BROWN = (139, 69, 19)
GRAY = (128, 128, 128)

# Board
board = [[0] * 10]
next_block = None
current_block = None
current_x = 0
current_y = 0
score = 0
level = 1
paused = False
game_over = False

def draw_board():
    for y in range(20):
        for x in range(10):
            if board[y][x] != 0:
                color = board[y][x]
                pygame.draw.rect(screen, color, (x * 40 + 40, y * 40 + 40, 40, 40))

def draw_next_block():
    if next_block is not None:
        color = next_block
        pygame.draw.rect(screen, color, (500, 100, 40, 40))

def rotate(block):
    if block == 0:
        return 0
    elif block == 1:
        return 2
    elif block == 2:
        return 3
    elif block == 3:
        return 1
    elif block == 4:
        return 5
    elif block == 5:
        return 6
    elif block == 6:
        return 7
    elif block == 7:
        return 4

def move(block, x, y):
    if 0 <= x < 10 and 0 <= y < 20:
        return x, y
    return None, None

def drop(block, x, y):
    while True:
        x, y = move(block, x, y + 1)
        if x is None or y is None:
            break
    return x, y

def hard_drop(block, x, y):
    x, y = drop(block, x, y)
    board[y][x] = block
    return x, y

def soft_drop(block, x, y):
    x, y = drop(block, x, y)
    board[y][x] = block
    return x, y

def rotate_block(block, x, y):
    new_block = rotate(block)
    new_x, new_y = move(new_block, x, y)
    if new_x is not None and new_y is not None:
        return new_block, new_x, new_y
    return block, x, y

def clear_lines():
    lines = []
    for y in range(20):
        if all(board[y][x] != 0 for x in range(10)):
            lines.append(y)
    for y in lines:
        board[y] = [0] * 10
    return len(lines)

def get_block():
    global next_block
    next_block = None
    for i in range(4):
        for j in range(4):
            if i != j:
                next_block = i * 4 + j
                break
        if next_block is not None:
            break
    return next_block

def game_loop():
    global next_block, current_block, current_x, current_y, score, level, paused, game_over
    next_block = get_block()
    current_block = next_block
    current_x = 5
    current_y = 0
    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                game_over = True
            elif event.type == pygame.KEYDOWN:
                if event.key == pygame.K_LEFT:
                    current_x -= 1
                    current_x = max(0, current_x)
                elif event.key == pygame.K_RIGHT:
                    current_x += 1
                    current_x = min(9, current_x)
                elif event.key == pygame.K_UP:
                    rotate_block(current_block, current_x, current_y)
                elif event.key == pygame.K_Z:
                    rotate_block(current_block, current_x, current_y)
                elif event.key == pygame.K_DOWN:
                    soft_drop(current_block, current_x, current_y)
                elif event.key == pygame.K_SPACE:
                    hard_drop(current_block, current_x, current_y)
                elif event.key == pygame.K_C:
                    hold_block = current_block
                    current_block = get_block()
                    current_x = 5
                    current_y = 0
                elif event.key == pygame.K_P:
                    paused = not paused
                elif event.key == pygame.K_ESCAPE:
                    game_over = True
        if not paused:
            current_x, current_y = drop(current_block, current_x, current_y)
            board[current_y][current_x] = current_block
            lines = clear_lines()
            score += lines * level
            level = min(20, level + lines)
            current_block = get_block()
            current_x = 5
            current_y = 0
        pygame.draw.rect(screen, GRAY, (0, 0, 1000, 1000))
        draw_board()
        draw_next_block()
        pygame.display.flip()
        clock.tick(60)
    pygame.quit()

if __name__ == "__main__":
    game_loop()
```

+ 質問してみましょう

図1.ChatGPT とのチャット

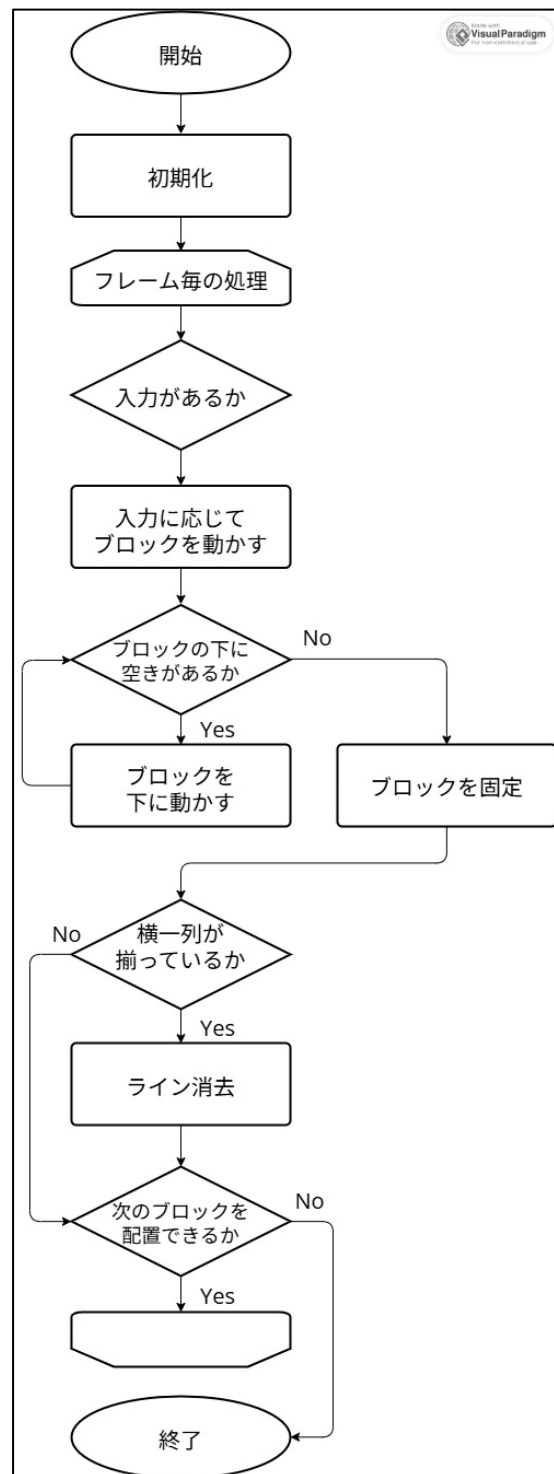


図2.プログラムのフローチャート

4. プログラムの実行

出力されたプログラムをダウンロードし、ファイルを Visual Studio Code で開きます。そのまま画面右上の▶マークをクリックすると、ゲームが別ウィンドウで開きます(図3)。テトリスとして問題なくプレイすることができました。

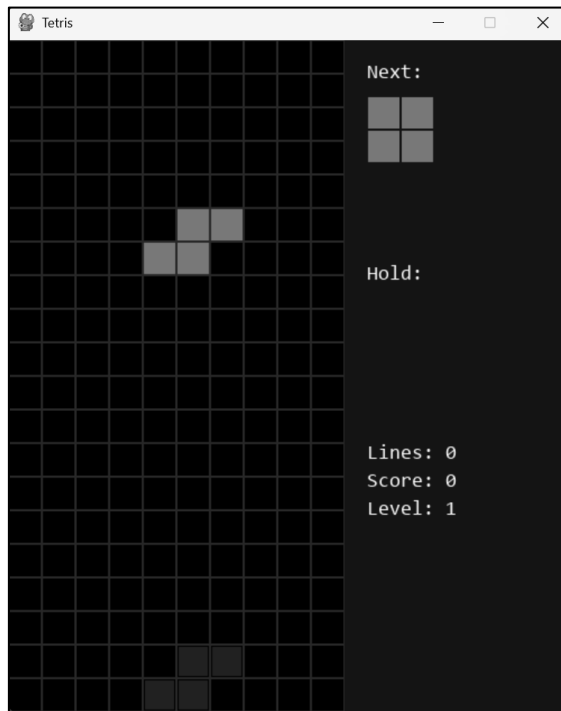


図3.ゲームのプレイ画面

5. 調整

動作が確認できたので、次はゲームを改善していきます。今回は、背景を黒から白、文字を白から黒に変えます。さらに、テトリミノ(ブロック)が完全ランダムではなく、公式の「テトリス」同様7種1巡で出現するようにします。どちらも、ChatGPT にその旨を伝えることで改善された状態のプログラムを出力することができます。出力されたプログラムを Visual Studio Code で実行すると、要望どおりのテトリスをプレイすることができました(図4)。

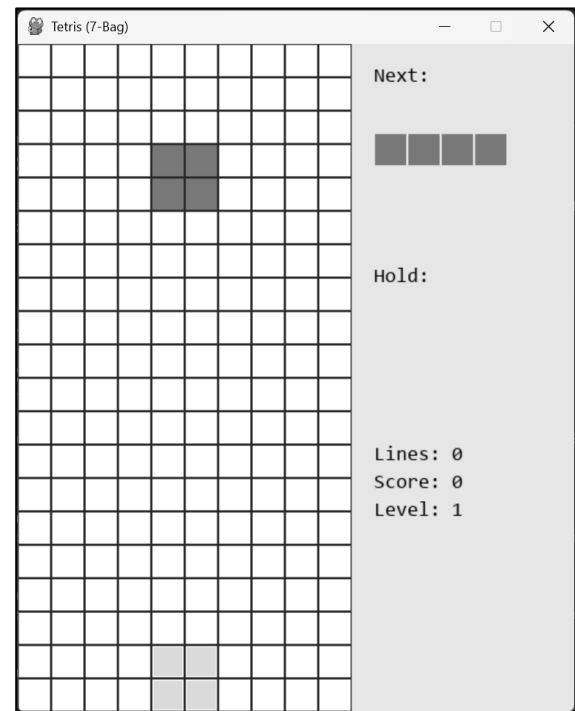


図4.改善後のプレイ画面

6. おわりに

数年前と比較してAIの性能は格段に向上しており、プログラムや設計という点においてもその変化が見られることに驚きました。私は今年からプログラミングの授業を受けはじめましたが、やはり一筋縄ではいかず、今回のようにAIに頼ることもあります。しかし、学習を重ね、自分でコードを書いたり、出力されたコードを理解して添削したりできるようになりたいです。

使用した Web サイト

ChatGPT

<https://chatgpt.com>

Python

<https://python.org>

Visual Studio Code

<https://code.visualstudio.com>

インド風の音楽を作る

ロボティクスコース 3年 黒澤 徹真

1. はじめに

今回、私はインド風の音楽を作ることに挑戦しました。ここでは、その過程と完成した作品を紹介します。

2. スケールについて

ドからシまでの12音の中から7音を抜き出して並べたものを「スケール」といいます。例えば、図1のようにピアノの鍵盤をイメージした時、白鍵だけ(ドレミファソラシド)を叩く場合、その音の組み合わせを「メジャースケール」といい、聞き手に明るい印象を与えます。また、ミ、ラ、シの3音を半音下げて弾くと(ドレミ \flat ファソラ \flat シ \flat ド)「マイナースケール」になり、暗い印象を与えます。この2つは最も広く使用されているスケールで、世の中にある殆どの音楽がこのスケールを使用しているといっても過言ではないでしょう。

さて、今回は「インド風の曲」を作るため、スケールもそれに合わせて変える必要があります。そこで使うのが「ラーガ・バイラブ・スケール」とよばれるものです。このスケールは図2に示したように、レとラを半音下げた「ドレ \flat ミファソラ \flat シド」で構成されています。このスケールを使うだけで、誰でも簡単にインド風の音楽が作れるのです。



図1.ピアノの鍵盤

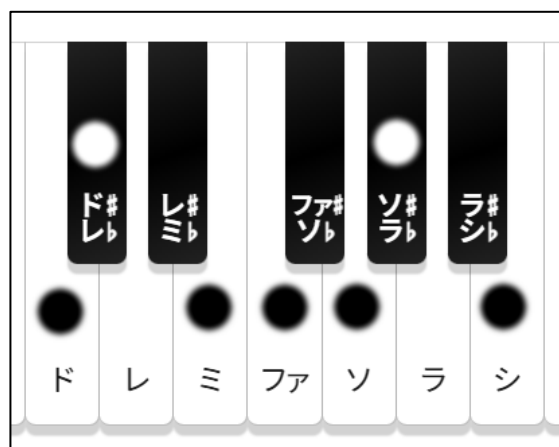


図2.ラーガ・バイラブ・スケール

3. ソフトウェアのダウンロード

今回は作曲に「Cakewalk by Bandlab」を使用するため、<https://www.cakewalk.com/>からダウンロードします。アカウントの作成は必要ですが、無料で使用できます。

4. 準備

ダウンロードとアカウントの作成が完了したら、早速「Cakewalk by Bandlab」を起動します。新規プロジェクトから Basic.cwt を選択し、今回使わないオーディオトラックを右クリックで削除します。そうしたら、トラックの上にある＋マークを押して

「インストゥルメント」を選択し、その中にある「インストゥルメント」を「Cakewalk TTS-1」に設定します。これで MIDI トラックに自動で音源が設定されます。個別に音源を設定するには、インストゥルメントトラックの「Cakewalk TTS-1」の左にあるアイコンをクリックしてメニュー(図3)を開きます。メニューからは音源を16個まで設定できるので、「EDIT」から音源名を左クリックして音源を変更することができます。

「表示」タブから「ピアノロールビュー」を選択すると、ピアノが表示されます(図4)。この画面をマウスでクリックすることで、音を直接書き込めるようになりました。



図3.メニュー

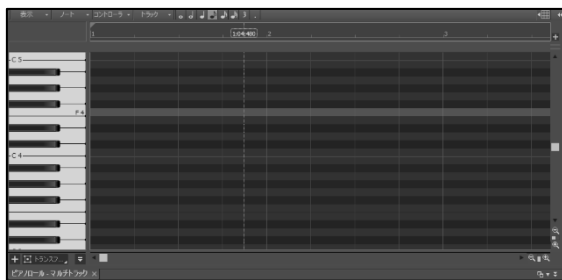


図4.ピアノロールビュー

5. 作曲

準備が完了したので、作曲に入ります。コード(3音の重なり)から作る方法が一般的ですが、今回は特徴的なスケールでメロディが作りやすいので、メロディラインから作ります。メロディはラーガ・バイラブ・スケールを意識して音を繋げていき、コードとベースはメロディに合うよう作っていきます。すべてのパートを合わせて違和感がなければ完成とします。

6. おわりに

完成した曲を Youtube にアップロードしました。図5の QR コードからページを開くことができるので、ぜひお聴きください。最後まで読んでいただきありがとうございました。

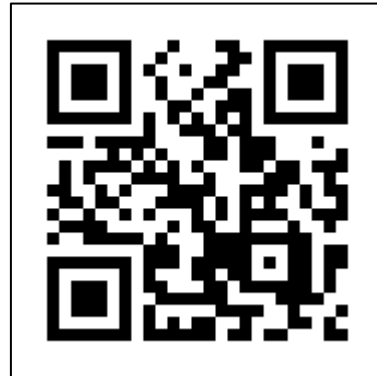


図5.動画の QR コード

参考文献

さあ、DTM をはじめよう！

<https://www.newdtm-rain.com/>

ギタリストによるラーガ・バイラブ・スケール
(インド音階)のすすめ

<https://thepocketguitar.com/archives/4744>

編集後記

今年も Rationale を無事発行できました。下の QR コードでは、この Rationale に加えて過去の Rationale の PDF が閲覧できます。興味ございましたら、ぜひご覧ください。

今年はプログラミングコンテスト関係の記事がない分、去年よりもページ数が少なくなっていました。その分個人の活動報告の記事が増えたことがとても嬉しいです。来年度の Rationale は、今年より分厚くなっていることを期待しています。

最後に、記事を執筆してくださった部員のみなさん、お忙しい中寄稿してくださった顧問の北島先生、そして最後まで Rationale を読んでくださったみなさんに感謝いたします。ありがとうございました。

編集 黒澤徹真

Rationale Volume39

発行日	10月25日
発行団体	仙台高専名取キャンパス ソフトウェア研究部会
発行所	仙台高専名取キャンパス

STAFF

代表	黒澤徹真
校正	黒澤徹真
執筆	ソフトウェア研究部員
製本	ソフトウェア研究部員



過去の Rationale はこちらからご覧いただけます