










 c0a24164f5 / ProjExD\_Group11



 Code  Pull requests  Actions  Projects  Wiki  Security  Insights  Settings

ProjExD\_Group11 / README.md 

c0a24164f5 README編集

31b88f4 · 37 minutes ago



34 lines (26 loc) · 1.56 KB

Preview

Code

Blame

Raw



## 実行環境の必要条件

- python >= 3.10
- pygame >= 2.1

## ゲームの概要

- シャイニングスターというフリー音源を用いて、音ゲーを作成しました。
- 参考URL：[魔王魂](#)

## ゲームの遊び方

- A,S,D,Fキーを用いて、落ちてくるノーツにタイミングよくキーを押す。
- 

## ゲームの実装

### 共通基本機能

- 背景画像と主人公キャラクターの描画

### 分担追加機能

- 丸焼きエフェクト（担当：ふしみ）：バーナーにより豪華豚を丸焼きにするエフェクトに関するクラス
- キッチンタイマー機能（担当：ぶしみ）：制限時間以内に調理が完了しなかった場合に、豪華豚が脱走する機能
- 調理機能（担当：ぶしみ）：調理器具をキー押下により選択し、豪華豚を調理する機能

### ToDo

- ☐ ほげほげ機能
- ☐ ふがふが関数内の変数名の統一

## メモ

- クラス内の変数は、すべて、「get\_変数名」という名前のメソッドを介してアクセスするように設計してある
- すべてのクラスに関係する関数は、クラスの外で定義してある

## 分担機能 SEとエフェクトの追加

- 関数play\_soundを実装：ボタンが押されたときT.mp3をロードし、再生する。
- 関数draw\_lane\_effectを実装：判定ごとにエフェクトを追加。PERFECT!は黄色、GOOD!は青、MISS!は赤、TOO LATE!は紫。



c0a24164f5 / ProjExD\_Group11



&lt;&gt; Code

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

ProjExD\_Group11 / rhythm\_game.py



c0a24164f5 エフェクト、SE実装完了

782b706 · 40 minutes ago



195 lines (166 loc) · 7.06 KB

```
1  import threading
2  import winsound
3
4  def play_beep(freq=600, dur=100):
5      threading.Thread(target=winsound.Beep, args=(freq, dur), daemon=True).start()
6
7  import pygame
8  import csv
9  import time
10 import sys
11
12 pygame.init()
13
14 SCREEN_WIDTH = 800
15 SCREEN_HEIGHT = 600
16 screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
17 pygame.display.set_caption("My Rhythm Game")
18
19 WHITE = (255, 255, 255)
```

ProjExD\_Group11 / rhythm\_game.py

↑ Top

Code

Blame

Raw



```
25 font = pygame.font.Font(None, 48)
26 small_font = pygame.font.Font(None, 36)
27
28 LANE_COUNT = 4
29 LANE_WIDTH = 100
30 LANE_SPACING = (SCREEN_WIDTH - LANE_COUNT * LANE_WIDTH) // (LANE_COUNT + 1)
31 NOTE_SPEED = 5
32 NOTE_HEIGHT = 20
33 JUDGEMENT_LINE_Y = SCREEN_HEIGHT - 100
34 JUDGEMENT_WINDOW = 30
35
36 lane_keys = {
37     pygame.K_a: {"lane_idx": 0, "color": (255, 100, 100)},
38     pygame.K_s: {"lane_idx": 1, "color": (100, 255, 100)},
39     pygame.K_d: {"lane_idx": 2, "color": (100, 100, 255)},
40     pygame.K_f: {"lane_idx": 3, "color": (255, 255, 100)}
41 }
42 key_to_lane_idx = {key: data["lane_idx"] for key, data in lane_keys.items()}
```

```
43 lane_idx_to_key_char = {0: 'A', 1: 'S', 2: 'D', 3: 'F'}
44
45 # 効果音の再生
46 ✓ def play_sound(path):
47     try:
48         sound = pygame.mixer.Sound(path)
49         sound.play()
50     except pygame.error:
51         print(f"効果音ファイル '{path}' を読み込めませんでした。")
52
53 # レーンごとの円形エフェクトを描画
54 def draw_lane_effect(screen, x_center, color, alpha=100, radius=50):
55     s = pygame.Surface((SCREEN_WIDTH, SCREEN_HEIGHT), pygame.SRCALPHA)
56     pygame.draw.circle(s, color + (alpha,), (x_center, JUDGEMENT_LINE_Y), radius)
57     screen.blit(s, (0, 0))
58
59 BEATMAP_FILE = 'beatmap.csv'
60 BEATMAP = []
61 try:
62     with open(BEATMAP_FILE, 'r') as f:
63         reader = csv.reader(f)
64         BEATMAP = [[int(row[0]), int(row[1])] for row in reader]
65 except FileNotFoundError:
66     print(f"エラー: '{BEATMAP_FILE}' が見つかりません。")
67     pygame.quit()
68     sys.exit()
69
70 beatmap_index = 0
71 notes = []
72 score = 0
73 combo = 0
74 max_combo = 0
75 judgement_effect_timer = 0
76 judgement_message = ""
77 judgement_color = WHITE
78
79 # レーンごとのエフェクト情報
80 lane_effects = [None] * LANE_COUNT
81 lane_effect_timers = [0] * LANE_COUNT
82
83 try:
84     pygame.mixer.music.load('ex5/maou_short_14_shining_star.mp3')
85 except pygame.error:
86     print("音楽ファイルをロードできませんでした。")
87
88 running = True
89 clock = pygame.time.Clock()
90 game_started = False
91 game_start_time = 0
92
93 while running:
94     if not game_started and BEATMAP:
95         pygame.mixer.music.play()
96         game_start_time = time.time()
97         game_started = True
98
99     for event in pygame.event.get():
100         if event.type == pygame.QUIT:
```

```
101     running = False
102     if event.type == pygame.KEYDOWN:
103         if event.key in lane_keys:
104             play_sound("ex5/T.mp3")
105
106         pressed_lane_idx = key_to_lane_idx[event.key]
107         hit_found = False
108         for note in notes[:]:
109             if note['lane'] == pressed_lane_idx and not note['hit']:
110                 if abs(note['rect'].centery - JUDGEMENT_LINE_Y) < JUDGEMENT_WINDOW:
111                     score += 100
112                     combo += 1
113                     max_combo = max(max_combo, combo)
114                     notes.remove(note)
115                     if abs(note['rect'].centery - JUDGEMENT_LINE_Y) < JUDGEMENT_WINDOW / 2:
116                         judgement_message = "PERFECT!"
117                         lane_effects[pressed_lane_idx] = (255, 255, 0) # 黄色
118                     else:
119                         judgement_message = "GOOD!"
120                         lane_effects[pressed_lane_idx] = (0, 128, 255) # 青
121                         judgement_color = GREEN
122                         lane_effect_timers[pressed_lane_idx] = 30
123                         judgement_effect_timer = 30
124                         hit_found = True
125                         break
126                 if not hit_found:
127                     combo = 0
128                     judgement_message = "MISS!"
129                     lane_effects[pressed_lane_idx] = (255, 0, 0) # 赤
130                     lane_effect_timers[pressed_lane_idx] = 30
131                     judgement_color = RED
132                     judgement_effect_timer = 30
133
134     if game_started:
135         current_game_time_ms = (time.time() - game_start_time) * 1000
136         while beatmap_index < len(BEATMAP) and current_game_time_ms >= BEATMAP[beatmap_index][0]:
137             note_data = BEATMAP[beatmap_index]
138             target_time_ms = note_data[0]
139             target_lane = note_data[1]
140             lane_x_start = LANE_SPACING + target_lane * (LANE_WIDTH + LANE_SPACING)
141             new_note_rect = pygame.Rect(lane_x_start, -NOTE_HEIGHT, LANE_WIDTH, NOTE_HEIGHT)
142             frames_to_travel = (JUDGEMENT_LINE_Y + NOTE_HEIGHT) / NOTE_SPEED
143             new_note_rect.y -= frames_to_travel * NOTE_SPEED
144             notes.append({'rect': new_note_rect, 'lane': target_lane, 'hit': False})
145             beatmap_index += 1
146
147     for note in notes[:]:
148         note['rect'].y += NOTE_SPEED
149         if note['rect'].top > JUDGEMENT_LINE_Y + JUDGEMENT_WINDOW and not note['hit']:
150             notes.remove(note)
151             combo = 0
152             judgement_message = "TOO LATE!"
153             lane = note['lane']
154             lane_effects[lane] = (128, 0, 128) # 紫
155             lane_effect_timers[lane] = 30
156             judgement_color = RED
157             judgement_effect_timer = 30
158
```

```
159     if judgement_effect_timer > 0:
160         judgement_effect_timer -= 1
161
162     screen.fill(BLACK)
163     for i in range(LANE_COUNT):
164         lane_x_start = LANE_SPACING + i * (LANE_WIDTH + LANE_SPACING)
165         pygame.draw.rect(screen, GRAY, (lane_x_start, 0, LANE_WIDTH, SCREEN_HEIGHT), 2)
166         key_char_text = small_font.render(lane_idx_to_key_char[i], True, WHITE)
167         screen.blit(key_char_text, (lane_x_start + (LANE_WIDTH - key_char_text.get_width()) // 2, JU
168
169     pygame.draw.rect(screen, GRAY, (0, JUDGEMENT_LINE_Y, SCREEN_WIDTH, NOTE_HEIGHT), 0)
170     pygame.draw.line(screen, WHITE, (0, JUDGEMENT_LINE_Y), (SCREEN_WIDTH, JUDGEMENT_LINE_Y), 3)
171
172     for note in notes:
173         pygame.draw.rect(screen, lane_keys[list(lane_keys.keys())[note['lane']]]['color'], note['rec
174
175     score_text = font.render(f"Score: {score}", True, WHITE)
176     combo_text = font.render(f"Combo: {combo}", True, WHITE)
177     screen.blit(score_text, (10, 10))
178     screen.blit(combo_text, (10, 50))
179
180     if judgement_effect_timer > 0:
181         judgement_display = font.render(judgement_message, True, judgement_color)
182         judgement_rect = judgement_display.get_rect(center=(SCREEN_WIDTH // 2, JUDGEMENT_LINE_Y - 50
183         screen.blit(judgement_display, judgement_rect)
184
185     # 各レーンのエフェクトを描画
186     for i in range(LANE_COUNT):
187         if lane_effect_timers[i] > 0 and lane_effects[i]:
188             x_center = LANE_SPACING + i * (LANE_WIDTH + LANE_SPACING) + LANE_WIDTH // 2
189             draw_lane_effect(screen, x_center, lane_effects[i])
190             lane_effect_timers[i] -= 1
191
192     pygame.display.flip()
193     clock.tick(60)
194
195     pygame.quit()
```

c0c2406216 / ProjExD\_Group11

<> CodeIssuesPull requestsActionsProjectsSecurityInsights

# Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#) or [learn more about diff comparisons](#).

base repository: c0c2406216/ProjExD\_Group11

base: main

←

...

head repository: c0a24164f5/ProjExD\_Group11

compare: C0A24164/エフェクト、SE

Discuss and review the changes in this comparison with others. [Learn about pull requests](#)

Create pull request

2 commits

2 files changed

1 contributor

Commits on Jul 15, 2025

- エフェクト、SE実装完了

c0a24164f5 committed 37 minutes ago

782b706

<>
- README編集

c0a24164f5 committed 35 minutes ago

31b88f4

<>

Showing 2 changed files with 53 additions and 43 deletions.

SplitUnified

6 README.md

@@ -1,4 +1,4 @@

1 - # 丸焼き豪華豚

1 + #

2 2

3 3

4 4

28 28

29 29

30 30

31 +

32 + ### 分擔機能 SEとエフェクトの追加

33 + \* 関数play\_soundを実装：ボタンが押されたときT.mp3をロードし、再生する。

34 + \* 関数draw\_lane\_effectを実装：判定ごとにエフェクトを追加。PERFECT!は黄色、GOOD!は青、MISS!は赤、TOO LATE!は紫。

90 rhythm\_game.py

@@ -1,29 +1,30 @@

1 + import threading

	2	+ import winsound
	3	+
	4	+ def play_beep(freq=600, dur=100):
	5	+     threading.Thread(target=winsound.Beep, args=(freq, dur), daemon=True).start()
	6	+
1	7	import pygame
2		- import csv # 譜面ファイルの読み込み用
3		- import time # ゲーム開始時間の記録用
	8	+ import csv
	9	+ import time
4	10	import sys
5	11	
6		- # Pygameの初期化
7	12	pygame.init()
8	13	
9		- # 画面設定
10	14	SCREEN_WIDTH = 800
11	15	SCREEN_HEIGHT = 600
12	16	screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
13	17	pygame.display.set_caption("My Rhythm Game")
14	18	
15		- # 色の定義
16	19	WHITE = (255, 255, 255)
17	20	BLACK = (0, 0, 0)
18	21	RED = (255, 0, 0)
19	22	GREEN = (0, 255, 0)
20	23	GRAY = (100, 100, 100)
21	24	
22		- # フォントの設定
23	25	font = pygame.font.Font(None, 48)
24	26	small_font = pygame.font.Font(None, 36)
25	27	
26		- # ゲーム設定
27	28	LANE_COUNT = 4
28	29	LANE_WIDTH = 100
29	30	LANE_SPACING = (SCREEN_WIDTH - LANE_COUNT * LANE_WIDTH) // (LANE_COUNT + 1)
32	33	JUDGEMENT_LINE_Y = SCREEN_HEIGHT - 100
33	34	JUDGEMENT_WINDOW = 30
34	35	
35		- # 各レーンに対応するキーと色、表示用の文字
36	36	lane_keys = {
37	37	pygame.K_a: {"lane_idx": 0, "color": (255, 100, 100)},
38	38	pygame.K_s: {"lane_idx": 1, "color": (100, 255, 100)},
42	42	key_to_lane_idx = {key: data["lane_idx"] for key, data in lane_keys.items()}
43	43	lane_idx_to_key_char = {0: 'A', 1: 'S', 2: 'D', 3: 'F'}
44	44	
45		- # --- ▼▼▼ ここからが大きな変更点 ▼▼▼ ---
	45	+ # 効果音の再生
	46	+ def play_sound(path):
	47	+     try:
	48	+         sound = pygame.mixer.Sound(path)
	49	+         sound.play()
	50	+     except pygame.error:
	51	+         print(f"効果音ファイル '{path}' を読み込めませんでした。")
	52	+
	53	+ # レーンごとの円形エフェクトを描画
	54	+ def draw_lane_effect(screen, x_center, color, alpha=100, radius=50):
	55	+     s = pygame.Surface((SCREEN_WIDTH, SCREEN_HEIGHT), pygame.SRCALPHA)



	56	+	pygame.draw.circle(s, color + (alpha,), (x_center, JUDGEMENT_LINE_Y), radius)
	57	+	screen.blit(s, (0, 0))
46	58		
47		-	# 譜面ファイルの読み込み
48	59		BEATMAP_FILE = 'beatmap.csv'
49	60		BEATMAP = []
50	61		try:
51	62		with open(BEATMAP_FILE, 'r') as f:
52	63		reader = csv.reader(f)
53		-	# [ヒットすべき時間(ms), レーン番号] のリストを作成
54	64		BEATMAP = [[int(row[0]), int(row[1])] for row in reader]
55	65		except FileNotFoundError:
56	66		print(f"エラー: '{BEATMAP_FILE}' が見つかりません。")
57		-	print("先に 'create_beatmap.py' を実行して譜面ファイルを作成してください。")
58	67		pygame.quit()
59	68		sys.exit()
60	69		
61		-	beatmap_index = 0 # 次に生成すべきノーツのインデックス
62		-	
63		-	# ノーツのリスト
	70	+	beatmap_index = 0
64	71		notes = []
65		-	
66		-	# スコアとコンボ
67	72		score = 0
68	73		combo = 0
69	74		max_combo = 0
70		-	
71		-	# 判定エフェクトの表示設定
72	75		judgement_effect_timer = 0
73	76		judgement_message = ""
74	77		judgement_color = WHITE
75	78		
76		-	# 音楽のロードと再生
	79	+	# レーンごとのエフェクト情報
	80	+	lane_effects = [None] * LANE_COUNT
	81	+	lane_effect_timers = [0] * LANE_COUNT
	82	+	
77	83		try:
78		-	pygame.mixer.music.load('maou_short_14_shining_star.mp3')
	84	+	pygame.mixer.music.load('ex5/maou_short_14_shining_star.mp3')
79	85		except pygame.error:
80		-	print("警告: 音楽ファイルをロードできませんでした。")
	86	+	print("音楽ファイルをロードできませんでした。")
81	87		
82		-	# ゲームループのフラグとクロック
83	88		running = True
84	89		clock = pygame.time.Clock()
85	90		game_started = False
86	91		game_start_time = 0
87	92		
88		-	# ----- メインのゲームループ -----
89	93		while running:
90	94		if not game_started and BEATMAP:
91		-	# 最初のノーツがあればゲームを開始
92	95		pygame.mixer.music.play()
93	96		game_start_time = time.time()
94	97		game_started = True

95	98	
96		- # --- 1. イベント処理 ---
97	99	for event in pygame.event.get():
98	100	if event.type == pygame.QUIT:
99	101	running = False
100	102	if event.type == pygame.KEYDOWN:
101	103	if event.key in lane_keys:
	104	+ play_sound("ex5/T.mp3")
	105	+
102	106	pressed_lane_idx = key_to_lane_idx[event.key]
103	107	hit_found = False
104	108	for note in notes[:]:
110	114	notes.remove(note)
111	115	if abs(note['rect'].centery - JUDGEMENT_LINE_Y) <
		JUDGEMENT_WINDOW / 2:
112	116	judgement_message = "PERFECT!"
	117	+ lane_effects[pressed_lane_idx] = (255, 255, 0) # 黄色
113	118	else:
114	119	judgement_message = "GOOD!"
	120	+ lane_effects[pressed_lane_idx] = (0, 128, 255) # 青
115	121	judgement_color = GREEN
	122	+ lane_effect_timers[pressed_lane_idx] = 30
116	123	judgement_effect_timer = 30
117	124	hit_found = True
118	125	break
119	126	if not hit_found:
120	127	combo = 0
121	128	judgement_message = "MISS!"
	129	+ lane_effects[pressed_lane_idx] = (255, 0, 0) # 赤
	130	+ lane_effect_timers[pressed_lane_idx] = 30
122	131	judgement_color = RED
123		- judgement_effect_timer = 30
124		-
125		- # --- 2. ゲームの状態更新 ---
	132	+ judgement_effect_timer
126	133	
127		- # 【変更点】ランダム生成の代わりに譜面からノートを生成
128	134	if game_started:
129	135	current_game_time_ms = (time.time() - game_start_time) * 1000
130		-
131		- # 譜面の最後までチェック
132	136	while beatmap_index < len(BEATMAP) and current_game_time_ms >=
		BEATMAP[beatmap_index][0]:
133	137	note_data = BEATMAP[beatmap_index]
134	138	target_time_ms = note_data[0]
135	139	target_lane = note_data[1]
136		-
137		- # 新しいノートを作成
138	140	lane_x_start = LANE_SPACING + target_lane * (LANE_WIDTH + LANE_SPACING)
139	141	new_note_rect = pygame.Rect(lane_x_start, -NOTE_HEIGHT, LANE_WIDTH,
		NOTE_HEIGHT)
140		- # y座標を調整: 判定ラインから逆算して、正しいタイミングでラインに到達するようにする
141		- # (移動フレーム数 = 距離 / 速度)
142	142	frames_to_travel = (JUDGEMENT_LINE_Y + NOTE_HEIGHT) / NOTE_SPEED
143		- # 予めそのフレーム数分だけ上に配置しておく
144	143	new_note_rect.y -= frames_to_travel * NOTE_SPEED
145		-
146	144	notes.append({'rect': new_note_rect, 'lane': target_lane, 'hit': False})

147		-
148		- beatmap_index += 1 # 次のノートへ
	145	+ beatmap_index += 1
149	146	
150		- # ノーツの移動と判定外れチェック
151	147	for note in notes[:]:
152	148	note['rect'].y += NOTE_SPEED
153	149	if note['rect'].top > JUDGEMENT_LINE_Y + JUDGEMENT_WINDOW and not note['hit']:
154	150	notes.remove(note)
155	151	combo = 0
156	152	judgement_message = "TOO LATE!"
	153	+ lane = note['lane']
	154	+ lane_effects[lane] = (128, 0, 128) # 紫
	155	+ lane_effect_timers[lane] = 30
157	156	judgement_color = RED
158	157	judgement_effect_timer = 30
159	158	
160	159	if judgement_effect_timer > 0:
161	160	judgement_effect_timer -= 1
162	161	
163		- # --- 3. 描画 ---
164	162	screen.fill(BLACK)
165	163	for i in range(LANE_COUNT):
166	164	lane_x_start = LANE_SPACING + i * (LANE_WIDTH + LANE_SPACING)
167	165	pygame.draw.rect(screen, GRAY, (lane_x_start, 0, LANE_WIDTH, SCREEN_HEIGHT), 2)
168	166	key_char_text = small_font.render(lane_idx_to_key_char[i], True, WHITE)
169	167	screen.blit(key_char_text, (lane_x_start + (LANE_WIDTH - key_char_text.get_width()) // 2, JUDGEMENT_LINE_Y + 50))
	168	+
170	169	pygame.draw.rect(screen, GRAY, (0, JUDGEMENT_LINE_Y, SCREEN_WIDTH, NOTE_HEIGHT), 0)
171	170	pygame.draw.line(screen, WHITE, (0, JUDGEMENT_LINE_Y), (SCREEN_WIDTH, JUDGEMENT_LINE_Y), 3)
172	171	
183	182	judgement_rect = judgement_display.get_rect(center=(SCREEN_WIDTH // 2, JUDGEMENT_LINE_Y - 50))
184	183	screen.blit(judgement_display, judgement_rect)
185	184	
	185	+ # 各レーンのエフェクトを描画
	186	+ for i in range(LANE_COUNT):
	187	+ if lane_effect_timers[i] > 0 and lane_effects[i]:
	188	+ x_center = LANE_SPACING + i * (LANE_WIDTH + LANE_SPACING) + LANE_WIDTH // 2
	189	+ draw_lane_effect(screen, x_center, lane_effects[i])
	190	+ lane_effect_timers[i] -= 1
	191	+
186	192	pygame.display.flip()
187	193	clock.tick(60)
188	194	