

Synopse - Pygame

Real-estate board game made with Pygame

Oliver Mikkell Opfermann Göth
31-05-2024

Indholdsfortegnelse

Introduktion.....	2
Problemformulering.....	2
Arbejdsspørgsmål	2
Metode.....	2
Tidsplan	3
Pygame	3
Grundlæggende komponenter	4
Grundlæggende Introduktion til Pygame	5
Installation.....	6
Essentielle Komponenter.....	6
Ikke-Essentielle Komponenter	7
Brætspil/ brætspil bane i Pygame	8
Håndtering af input for styring af spilfiguren	10
Konklusion	11
Refleksion	13
Litteratur liste	14
Bilag.....	14
Tidsplan	14
Billede af Spil.....	17

Introduktion

Velkommen til et spændende 4. semester projekt, hvor jeg vil udforske og lære gennem udviklingen af et spil i Python ved hjælp af Pygame-frameworket. I dette projekt vil jeg tage inspiration fra et brætspil der hedder "Cashflow Get Out Of The Rat Race" og skabe min egen digitale version af det.

Formålet med dette projekt er ikke kun at implementere et spil, men primært at dykke ned i bibliotekspakken Pygame, og få en betydelig stor mængde viden inden for Pygame.

Gennem dette projekt vil jeg have mulighed for at lære og anvende nye færdigheder og teknikker, samtidig med at styrker min forståelse for Python-programmering og softwarearkitektur, samt at lære Pygame. Projektet vil give mig mulighed for at udforske og eksperimentere med forskellige løsninger og tilgange, samtidig med at jeg arbejder på at skabe et spil, der er sjovt og engagerende.

Jeg vil blive opfordret til at undersøge og implementere unikke funktioner og mekanikker jeg ikke har lavet før, så det korrekt simulere det brætspil som det er baseret på.

Problemformulering

Hvordan kan man lave "Cashflow Get Out Of The Rat Race" ved hjælp af Python og Pygame-biblioteket.

Arbejdsspørgsmål

1. Hvad er Pygame, og hvilke grundlæggende komponenter består det af?
2. Hvordan kan man anvende Pygame?
3. Hvordan håndteres brugerinput i Pygame for at styre spillerfiguren?
4. Hvordan kan man lave et brætspil/ brætspil bane i Pygame?
5. Hvordan implementeres spillogikken, herunder scoring og afslutning af spillet?

Metode

Måden hvorpå jeg ønsker at finde svar på ovennævnte problemformulering, vil primært foregå igennem researcharbejde. Denne research afgrænser til bl.a. læsning af relevante dokumentation at se videovejledninger samt selv at udvikle egne eksperimenter. For at sikre sig kildernes korrekthed og relevans, vil jeg – så vidt det er muligt arbejde primært med den officielle dokumentation fra Pygame.

Jeg vil under programmeringsdelen af mit projekt gøre brug af principperne ”The Zen Of Python”, det er en gruppe af 19 vejledende principper til at skrive Python med. Det fokuserer på designet af koden, og værdisætter simplicitet og langvarighed over kompleks kode.

Tidsplan

Jeg har opdelt min arbejdsproces i 4,5 uger, begyndende fra onsdag den 1. maj og sluttende fredag den 31. maj. Hver mandag planlægger jeg ugens aktiviteter og udarbejder et skema for hele ugen. Jeg beslutter, hvad jeg skal arbejde med i den pågældende uge, baseret på de vigtigste krav for at få spillet til at fungere samt eventuelle krav til koden.

Den sidste uge har jeg reserveret til at arbejde på synopsen for at sikre, at jeg har tilstrækkelig tid samt til at rette eventuelle fejl i synopsen, som jeg muligvis har lavet.

Her er et link til min tidsplan for projektet, der vil også være billeder af tidsplanen i bilag:

<https://zealanddk.sharepoint.com/:x/s/Synopsisvejledning4.semester-F2024/Ef1Z1nW-jCxFiWmOavy9a1EBVyw-vDb2b2opqgv7BmMdzg?e=z8k2Hz6dc26c913170&wdPreviousSessionSrc=HarmonyWeb&wdPreviousSession=2e32f7c2-b0a5-4205-a999-6fcb2c2fc0be>

Pygame

Pygame er en bibliotekspakke til Python, designet med formålet at gøre det muligt at nemt udvikle spil i programmeringssproget python. Det bliver derfor primært anvendt til udvikling af spil men også multimedieprojekter. Det giver udviklere et sæt værktøjer og funktioner til at oprette interaktive applikationer, især inden for 2D-spiludvikling. Pygame er baseret på Simple DirectMedia Layer (SDL), hvilket gør det platformuafhængigt og understøtter forskellige operativsystemer som Windows, macOS og Linux. Samt at de kan udvide disse egenskaber og tilbyde en masse funktioner og klasser til nemt at håndtere de mest grundlæggende game mechanics.

Med Pygame kan udviklere oprette vinduer, håndtere brugerinput som tastatur og mus, tegne grafik og manipulere billeder, afspille lyd og musik, håndtere kollisioner mellem spilobjekter, og meget mere. Det er et alsidigt værktøjssæt, der gør det muligt for både begyndere og erfarne udviklere at skabe alt fra enkle arkadespil til mere komplekse simuleringer.

Da Pygame er baseret på Python, gør det sprogets syntaks og struktur let tilgængelig for udviklere, hvilket gør det relativt nemt at lære og bruge. Samtidig giver det også mulighed for integration af Python's omfattende økosystem af biblioteker og værktøjer, hvilket giver udviklere stor fleksibilitet og kontrol over deres projekter.

Ved at bruge et værktøj som Pygame kan programmøren lægge sin energi i at skabe deres ideer og koncepter, da Pygame er lavet til at håndtere meget af de tekniske dele ved spiludvikling. Pygame inkluderer en masse klasser som er lavet til at håndtere forskellige vigtige dele af spil udvikling heriblandt billeder, lyd, vinduer osv. Dette hjælper programmøren til at kunne implementere deres ideer til game mechanics uden at skulle bruge tid og frustrationer på at forstå de komplekse detaljer vedrørende grafikprogrammering og hardwareintegration.

Grundlæggende komponenter

Modulet `pygame.display` styrer spilvinduet og skærmen, hvor spillet vises. Det giver en række funktioner, som gør det muligt at initialisere, konfigurere og opdatere displayet. For eksempel kan man ved hjælp af `pygame.display.set_mode()` oprette et vindue med en specificeret bredde og højde. Funktionen `pygame.display.set_caption()` bruges til at sætte titlen på vinduet, mens `pygame.display.update()` eller `pygame.display.flip()` opdaterer skærmen med de nyeste ændringer. Modulet håndterer også interaktioner med brugeren, såsom ændringer i vinduestørrelse eller lukkehandlinger, der kan fanges gennem hændelseshåndtering.

```
29 screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT), pygame.RESIZABLE)
30 pygame.display.set_caption("Cashcow Game")
```

`pygame.Surface` repræsenterer billeder og kanvas i spillet. Det er et af de mest centrale objekter i Pygame, da det bruges til at tegne alt, hvad der skal vises på skærmen. Overflader kan skabes med `pygame.Surface()`, og man kan tegne på dem ved hjælp af forskellige metoder som `blit()`, som kopierer en overflade til en anden, samt tegne primitive former som linjer, cirkler og rektangler. Overflader kan også anvendes til at håndtere gennemsigtighed og billedmanipulation.

Modulet `pygame.event` håndterer alle hændelser i spillet, såsom tastaturtryk, museklik og vindueslukning. Hændelser gemmes i en kø, og man kan bruge `pygame.event.get()` til at hente dem. Man kan også oprette brugerdefinerede hændelser ved hjælp af `pygame.event.Event()`. Det er vigtigt for at skabe interaktive spil, hvor spillerens input styrer spillets handlinger og respons.

```
240 for event in pygame.event.get():
241     if event.type == pygame.QUIT:
242         pygame.quit()
243         sys.exit()
```

Modulet `pygame.image` bruges til at indlæse og gemme billeder. Funktionen `pygame.image.load()` gør det muligt at indlæse billeder fra filer i forskellige formater (såsom PNG, JPEG), som derefter kan bruges som `Surface`-objekter. Dette modul tilbyder også

`pygame.image.save()`, der gemmer en overflade til en fil. Billedhåndtering er essentiel for at kunne inkludere grafik og sprites i spillet.

```
18 dice_images = [pygame.image.load(f'Graphics/Animation/Dice/d{i}.png').convert_alpha() for i in range(1, 7)]
19 dice_images = [pygame.transform.scale(image, (100, 100)) for image in dice_images]
```

`pygame.sprite` tilbyder en række klasser og metoder til at arbejde med grupper af objekter, der bevæger sig rundt i spillet. Den centrale klasse er `Sprite`, som repræsenterer en individuel sprite med egenskaber som billede og rektangel til kollision. Man kan samle flere sprites i grupper ved hjælp af `Group`-klassen, hvilket gør det lettere at opdatere og tegne mange sprites på én gang. Dette modul forenkler håndteringen af animationer og kollisioner mellem forskellige spilobjekter.

```
51 dice1_sprite = Dice(dice_images[dice1 - 1], center_x, center_y)
```

Modulet `pygame.mixer` håndterer lyd i spillet, herunder afspilning af musik og lydeffekter. Man kan bruge `pygame.mixer.Sound()` til at indlæse lydeffekter og `pygame.mixer.music` til at afspille musikfiler. Dette modul understøtter grundlæggende lydfunktioner såsom afspilning, pause og stop, og man kan justere lydstyrken individuelt for lydeffekter og musik. Lyd er en vigtig del af spiloplevelsen og hjælper med at skabe en mere engagerende atmosfære.

`pygame.font` modulet håndterer tekstgengivelse i spillet. Med `pygame.font.Font()` kan man vælge skrifttyper og størrelser til teksten. Metoden `render()` bruges til at tegne teksten som en overflade, der kan blit'es til skærmen. Tekst er afgørende for at vise score, beskeder og andre vigtige oplysninger til spilleren.

```
10 text_font = pygame.font.SysFont("Arial", 30)
```

Modulet `pygame.time` giver funktioner til tidsstyring og timing i spillet. Funktionen `pygame.time.Clock()` bruges til at styre spillets framerate ved hjælp af metoden `tick()`, som sikrer, at spillet kører med en konstant hastighed. Man kan også bruge `pygame.time.get_ticks()` til at få tiden, der er gået, siden Pygame blev initialiseret, hvilket er nyttigt til timing og animationer. Tidsstyring er vigtig for at sikre et jævnt og stabilt spilforløb.

```
46 pygame.time.wait(200)
```

Grundlæggende Introduktion til Pygame

Pygame er et kraftfuldt bibliotek til at lave spil og multimedieapplikationer i Python. Her er en grundlæggende vejledning til at komme i gang med animation og lyd i Pygame.

Installation

Først skal du installere Pygame. Dette kan gøres med pip:

```
pip install pygame
```

Essentielle Komponenter

Pygame applikationer har typisk en hovedsløjfe, der håndterer begivenheder, opdaterer spillets tilstand og tegner alt på skærmen.

Initialisering: For at starte et Pygame-program skal biblioteket initialiseres ved at kalde `pygame.init()`, som opsætter alle nødvendige moduler. Initialiseringen sikrer, at alle komponenter er korrekt opsat og klar til brug.

```
6 pygame.init()
```

Skærmindstillinger: Dette er display, og som sagt styrer det alt om spilvinduet og skærmen, det er derfor essentielt at bruge det for et spil, ellers vil der ikke være nogen skræm. Det indeholder `pygame.display.Set_mode`, som vist før, hvilket kræver at man sætter screen height og width, som således:

```
8 SCREEN_WIDTH, SCREEN_HEIGHT = 1500, 1000
```

Main Loop: Main loop er central i et Pygame-program og styrer spillets kørsel. Den kan omfatte flere kritiske funktioner: begivenhedshåndtering, hvor brugerinput som tastetryk og musebevægelser behandles; opdatering af spillets tilstand, herunder figurers bevægelse og kollisionskontrol; rendering, hvor alle grafiske elementer tegnes på skærmen; og synkronisering ved hjælp af `pygame.time.Clock` for at sikre en konstant opdateringshastighed. Løkken fortsætter, indtil spillet afsluttes, typisk når en afslutningsbegivenhed registreres.

```

232 def main():
233     global player_track
234
235     clock = pygame.time.Clock()
236
237     dice_value, dice1 = dice_roll()
238
239     while True:
240         for event in pygame.event.get():
241             if event.type == pygame.QUIT:
242                 pygame.quit()
243                 sys.exit()
244             elif event.type == pygame.KEYDOWN:
245                 if event.key == pygame.K_r:
246                     rolling_audio.play()
247                     dice_value, dice1 = dice_roll()
248                     move_player(dice_value)
249                     rolling_stop_audio.play()
250                 if event.key == pygame.K_t:
251                     if player_track == 'rectangular':
252                         player_pos = 0
253                         player_track = 'circular'
254                     elif player_track == 'circular':
255                         player_pos = 0
256                         player_track = 'rectangular'
257
258                 screen.fill(BOARD_COLOR)
259                 draw_board()
260                 draw_player()
261                 display_dice(SCREEN_WIDTH // 2, SCREEN_HEIGHT // 2)
262
263                 pygame.display.flip()
264                 clock.tick(FPS)

```

Ikke-Essentielle Komponenter

Basiske komponenter som ikke er nødvendige, med øger brugeroplevelsen er komponenter så som:

Farver: I Pygame repræsenteres farver typisk som RGB-værdier, hver farve har en værdi mellem 0 og 255. Farver bruges til at tegne forskellige elementer på skærmen, såsom baggrunde, figurer og tekst. Eksempelvis kan man definere farver som WHITE = (255, 255, 255) og BLACK = (0, 0, 0), og derefter anvende dem i tegneoperationer som screen.fill(WHITE) for at gøre skærmen hvid.

```

14 WHITE = (255, 255, 255)
15 BLACK = (0, 0, 0)
16 RED = (255, 0, 0)
17 GREEN = (0, 255, 0)
18 BLUE = (0, 0, 255)
19 YELLOW = (255, 215, 0)
20 PURPLE = (128, 0, 128)
21 ORANGE = (255, 165, 0)

```


Lyd: Pygame understøtter afspilning af lydfile i forskellige formater som WAV, MP3 og OGG. Du kan afspille lyde ved hjælp af pygame.mixer-modulet. Her er nogle grundlæggende trin til at afspille en lyd:

```
23 rolling_audio = pygame.mixer.Sound('audio/roll_audio.mp3')
24 rolling_stop_audio = pygame.mixer.Sound('audio/roll_stop_audio.mp3')

245 if event.key == pygame.K_r:
246     rolling_audio.play()
247     dice_value, dice1 = dice_roll()
248     move_player(dice_value)
249     rolling_stop_audio.play()
```

Dette vil afspille lydfile roll_audio.mp3 og roll_audio_stop.mp3, når jeg kaster en terning.

Animation: Pygame gør det nemt at lave animationer ved hjælp af at blitse billeder på skærmen med en bestemt frekvens. For eksempel kan du lave animerede figurer, baggrunde eller specialeffekter. Dette er ikke noget jeg har implementeret i den aktuelle version af min kode. Jeg har dog planer om at implementere det inden min præsentation af projektet.

Brætspil/ brætspil bane i Pygame

På mit brætspil har jeg lavet 2 baner, da det er en del af det rigtige spil, og spiller en vigtig del i game mechanics.

Den ene bane er en firkantet bane 13x9, banen består af positioner der er arrangeret i en firkantet, der dækker kanterne af spillepladen. Jeg starter med at definere positionerne relativt til cellens bredde og højde. Dette gøres ved hjælp af en liste af tuples, hvor hver tuple repræsenterer en celle's x- og y-koordinater i forhold til et gitter.

```
80 cell_positions = [
81     (0, 0), (1, 0), (2, 0), (3, 0), (4, 0), (5, 0), (6, 0), (7, 0), (8, 0), (9, 0), (10, 0), (11, 0), (12, 0),
82     (12, 1),
83     (12, 2),
84     (12, 3),
85     (12, 4),
86     (12, 5),
87     (12, 6),
88     (12, 7),
89     (12, 8), (11, 8), (10, 8), (9, 8), (8, 8), (7, 8), (6, 8), (5, 8), (4, 8), (3, 8), (2, 8), (1, 8), (0, 8),
90     (0, 7),
91     (0, 6),
92     (0, 5),
93     (0, 4),
94     (0, 3),
95     (0, 2),
96     (0, 1)
97 ]
```

Hver tuple i `cell_positions` angiver en celle's position på spillepladen i form af (x, y) koordinater. For eksempel repræsenterer (0, 0) den første celle i det øverste venstre hjørne, og (12, 0) repræsenterer den sidste celle i den øverste række.

Når koden når den sidste tuple i `cell_position`, bruger jeg modulo operation (%), som er en del af python. Det er en standard aritmetisk operation i Python, der bruges til at finde resten, når et tal divideres med et andet. I min kode bruger jeg modulo operation til at sikre, at spillerens position går tilbage til begyndelsen af listen, når spilleren bevæger sig forbi den sidste celle. Dette gør jeg i metoden `move_player`, som også checker hvilken bane spilleren befinder sig på.

```
221 def move_player(steps):
222     global player_pos, player_track
223
224     if player_track == 'circular':
225         player_pos = (player_pos + steps) % len(circular_positions)
226     else:
227         player_pos = (player_pos + steps) % len(rectangular_positions)
```

For at konvertere disse relative positioner til absolutte skærmkoordinater, beregner vi startpunkterne for spillepladen (`start_x` og `start_y`) og tilføjer forskydninger for hver celle:

```
99 board_width = 13 * CELL_WIDTH
100 board_height = 9 * CELL_HEIGHT
101 start_x = (SCREEN_WIDTH - board_width) // 2
102 start_y = (SCREEN_HEIGHT - board_height) // 2
103
104 rectangular_positions = [(start_x + x * CELL_WIDTH, start_y + y * CELL_HEIGHT) for x, y in cell_positions]
```

- `board_width` og `board_height` beregner den samlede bredde og højde af spillepladen baseret på antallet af celler og cellernes dimensioner.
- `start_x` og `start_y` beregner de øvre venstre startkoordinater for spillepladen, så den bliver centreret på skærmen.
- `rectangular_positions` er en liste af tuples, der indeholder de absolutte (x, y) skærmkoordinater for hver celle.

Som sagt er der også en rund bane inde i midten af den firkantede bane, den runde bane er arrangeret i en cirkel omkring midten af skærmen. Jeg har været nødt til at lave denne bane på en alternativ måde da det ikke er muligt at lave den via den samme metode jeg brugte til at lave den firkantede bane. Jeg har derfor taget brug af `math` modulet, som er en standard del af Python.

Til oprettelsen af den runde bane definerer jeg en funktion, `generate_circular_positions`, til at beregne disse positioner:

```

106 def generate_circular_positions(center_x, center_y, radius, num_cells):
107     angle_step = 2 * math.pi / num_cells
108     return [
109         (
110             int(center_x + radius * math.cos(i * angle_step) - CELL_WIDTH // 2),
111             int(center_y + radius * math.sin(i * angle_step) - CELL_HEIGHT // 2)
112         )
113         for i in range(num_cells)
114     ]

```

- center_x og center_y er koordinaterne for cirkelns centrum.
- radius er cirkelns radius.
- num_cells er antallet af celler, der skal placeres rundt om cirklen.
- angle_step beregner vinklen mellem hver celle i radianer.

For hver celle beregnes dens (x, y) position ved at bruge trigonometri (cosinus og sinus funktioner) til at placere cellen korrekt på cirklen. Resultatet justeres derefter for at centrere cellen korrekt.

Jeg kalder derefter denne funktion for at generere de cirkulære positioner:

```

116 center_x, center_y = SCREEN_WIDTH // 2, SCREEN_HEIGHT // 2
117 radius = 3 * CELL_WIDTH
118
119 circular_positions = generate_circular_positions(center_x, center_y, radius, 20)

```

- center_x og center_y sætter midten af cirklen til midten af skærmen.
- radius sætter cirkelns radius til tre gange cellens bredde.
- circular_positions indeholder de absolutte (x, y) skærmkoordinater for hver celle i cirklen.

Håndtering af input for styring af spilfiguren

Håndteringen af input kræver en spilfigur, som er en klasse eller et objekt, der indeholder dens position, bevægelseshastighed og metode til at opdatere spilfigurens position baseret på input.

Når en tast er trykket ned, genereres en pygame.KEYDOWN-begivenhed. Vi kan identificere hvilken tast, der blev trykket, ved at kontrollere event.key attributten. I vores kode håndterer vi to taster:

- **K_r**: Ruller terningerne og flytter spilfiguren.
- **K_t**: Skifter mellem den rektangulære og den cirkulære bane.

```

244         elif event.type == pygame.KEYDOWN:
245             if event.key == pygame.K_r:
246                 rolling_audio.play()
247                 dice_value, dice1 = dice_roll()
248                 move_player(dice_value)
249                 rolling_stop_audio.play()
250             if event.key == pygame.K_t:
251                 if player_track == 'rectangular':
252                     player_pos = 0
253                     player_track = 'circular'
254                 elif player_track == 'circular':
255                     player_pos = 0
256                     player_track = 'rectangular'

```

- **pygame.K_r**: Når r-tasten trykkes, ruller vi terningerne ved at kalde `dice_roll()` og flytter spilfiguren ved at kalde `move_player(dice_value)`.
- **pygame.K_t**: Når t-tasten trykkes, skifter vi mellem banerne ved at nulstille `player_pos` til 0 og ændre `player_track` mellem 'rectangular' og 'circular'.

Jeg har også funktionen `move_player` som er ansvarlig for at opdatere spillerens position baseret på antallet af skridt:

```

221 def move_player(steps):
222     global player_pos, player_track
223
224     if player_track == 'circular':
225         player_pos = (player_pos + steps) % len(circular_positions)
226     else:
227         player_pos = (player_pos + steps) % len(rectangular_positions)

```

- **player_track**: Kontrollerer, hvilken bane spilleren er på.
- **player_pos**: Opdateres baseret på antallet af skridt og banens længde.

Konklusion

I dette projekt har jeg undersøgt og udviklet et digitalt brætspil ved hjælp af Pygame-biblioteket, inspireret af "Cashflow Get Out Of The Rat Race". Projektets hovedmål var at opnå en dybere forståelse af Pygame og implementere et funktionelt spil.

Arbejdsspørgsmål 1: Hvad er Pygame og hvilke grundlæggende komponenter består det af? Pygame er et bibliotek til Python, designet til at gøre spiludvikling nemmere. De grundlæggende komponenter inkluderer display, surface, event, image, sprite, mixer, font, og time, som alle spiller en vigtig rolle i håndteringen af spilvinduer, grafik, brugerinput, lyd, tekst og timing.

Arbejdsspørgsmål 2: Hvordan kan man anvende Pygame? Pygame anvendes ved at initialisere biblioteket, oprette et display-vindue, håndtere begivenheder, opdatere spilstilstande og tegne grafik på skærmen. Den centrale del af en Pygame-applikation er hovedsløjfen, som styrer spillets kørsel og sørger for, at alt fungerer korrekt og opdateres jævnlige.

Arbejdsspørgsmål 3: Hvordan håndteres brugerinput i Pygame for at styre spillerfiguren? Brugerinput håndteres gennem pygame.event-modulet, som registrerer tastaturtryk, museklik og andre begivenheder. Specifikke input kan knyttes til handlinger, såsom at rulle terninger eller flytte spilfiguren, ved at definere funktioner, der udføres når bestemte taster trykkes.

Arbejdsspørgsmål 4: Hvordan kan man lave et brætspil/brætspilbane i Pygame? For at lave et brætspil i Pygame kræver det design af spilbrættet, herunder oprettelse af grafiske elementer og layout af spillebanen. Dette indebærer også programmering af reglerne for spillet og logikken for, hvordan spillerne interagerer med brættet.

Arbejdsspørgsmål 5: Hvordan implementeres spillogikken, herunder scoring og afslutning af spillet? Spillogikken implementeres ved at definere reglerne for spillet, herunder hvordan spillere scorer point og hvordan spillet afsluttes. Dette kræver oprettelse af funktioner, der opdaterer spillerens status og spillets tilstand baseret på spillerens handlinger og spillets regler.

Problemformulering: Hvordan kan man lave "Cashflow Get Out Of The Rat Race" ved hjælp af Python og Pygame-biblioteket? Ved at anvende Pygames værktøjer og komponenter har jeg skabt en digital version af "Cashflow Get Out Of The Rat Race". Dette inkluderede at designe spilbrættet, implementere spillogikken for bevægelse og scoring, samt håndtere brugerinput for at styre spilfigurerne. Projektet har demonstreret, at Pygame kan bruges til at udvikle et komplekst brætspil med både firkantede og runde baner.

Gennem dette projekt har jeg fået værdifuld viden om Pygame og spiludvikling, som kan anvendes i andre områder af softwareudvikling. For eksempel kan de teknikker, jeg har lært til håndtering af brugerinput og grafisk rendering, anvendes i udviklingen af andre interaktive applikationer og simulationsprogrammer.

Afslutningsvis vil jeg gemme en del af den dybere diskussion om, hvordan spillogikken blev implementeret, til eksamen, hvor jeg vil kunne gå mere i detaljer med specifikke tekniske udfordringer og løsninger.

Dette projekt har givet mig en solid forståelse af Pygame og en praktisk oplevelse i at udvikle et spil fra bunden, hvilket vil være nyttigt i fremtidige softwareudviklingsprojekter.

Refleksion

Gennem arbejdet med dette projekt har jeg været meget ambitiøs med, hvor meget jeg kunne opnå inden for den givne tidsramme. Jeg satte mig for at udvikle et komplekst brætspil ved hjælp af Pygame, hvilket har vist sig at være en større udfordring end først antaget. Selvom jeg har formået at skabe et funktionelt spil, har jeg indset, at valg af spiltype spillede en stor rolle i, hvor let det var at finde relevant research og vejledning. Mange ressourcer om Pygame fokuserer på platformsspil og andre mere traditionelle computerspil typer, hvilket har gjort det sværere at finde specifik hjælp til udviklingen af et brætspil. Denne erfaring har lært mig vigtigheden af at vælge et projekt, hvor der er tilstrækkelig støtte og eksempler tilgængelige.

Når jeg ser tilbage, kunne det måske have været mere effektivt at vælge et andet Python-bibliotek eller framework, der er bedre egnet til brætspiludvikling. At undersøge og vælge det rette værktøj til opgaven er en vigtig læring, som jeg vil tage med mig til fremtidige projekter.

Min metode har også vist sig at være lidt uhensigtsmæssig. Jeg har brugt meget tid på at fokusere på koden og tekniske detaljer i stedet for at balancere dette med arbejdet på synopsen. Dette har ført til en mindre sammenhængende og gennemarbejdet synopse end ønsket. I fremtidige projekter vil jeg sikre at opretholde en bedre balance mellem den tekniske udvikling og dokumentationen. Det er afgørende at dokumentere processen og refleksionerne grundigt for at sikre, at projektet ikke kun er teknisk solidt, men også godt præsenteret og let forståeligt.

Under reviseringen af mine arbejdsspørgsmål har jeg besluttet at fjerne spørgsmålet 'Hvordan kan man lave en basal AI i mit spil?' da jeg havde tidsbegrænsninger, som endte i at implementeringen af denne feature inden for projektets tidsrammer, ikke vil være muligt. Denne beslutning blev taget for at fokusere på mere centrale aspekter af spiludviklingen og sikre et mere tid til at arbejde på synopsen

Samlet set har dette projekt givet mig værdifuld indsigt i mine styrker og svagheder som udvikler. Jeg har lært meget om spiludvikling med Pygame og vigtigheden af at vælge det rette værktøj og metode for at optimere både udviklingsprocessen og det endelige resultat. Dette vil guide mig i fremtidige projekter, hvor jeg vil være mere strategisk i mine valg og sikre, at jeg opretholder en god balance mellem kodearbejde og dokumentation.

Litteratur liste

About Pygame. Pygame.org. <https://www.pygame.org/wiki/about>

Pygame.event. Pygame.org. <https://www.pygame.org/docs/ref/event.html>

Martin, Breuss. The Zen Of Python: Writing Idiomatic Python. RealPython.com.
<https://realpython.com/lessons/zen-of-python/>





Pygame tutorial 1-10. Youtube.com. <https://www.youtube.com/watch?v=i6xMBig-pP4&list=PLzMbBGfZo4-lp3jAExUCewBfMx3UZFkh5>

How can I go about creating a board game? [Det gav mig idden til at bruge tuples til at lave banen]Reddit.
https://www.reddit.com/r/learnpython/comments/ynvtd5/how_can_i_go_about_creating_a_board_game/

Bilag

Tidsplan

Uge 1

	uge 1			
dage	onsdag	torsdag	fredag	weekend
synopse	Oprettelse og start af synopse			
projekt	Oprettelse og start af Projekt	Oprettelse og start af synopse ----- Test/ lej med pygame	Oprettelse og start af synopse ----- Test/ lej med pygame	
ekstra		Research: introduktion til Pygame	Research: introduktion til Pygame	
Tidsplan i timer	 5	 4	 4	 0

Uge 2

uge 2					
mandag	tirsdag	onsdag	torsdag	fredag	weekend
Opdatering af tidsplan ----- Dokumentation	Skriv på synopsen om dice komponent og dokumentering	Skriv på synopsen om dice komponent og dokumentering	Skriv på synopsen om Board komponent og dokumentering	Skriv på synopsen om Board komponent og dokumentering	
Oprettelse af Dice component	Arbejde på Dice component	Arbejde på Dice component	Oprettelse af Board component	Arbejde på Board component	
Research: Dice	Research: Dice	Research: Dice	Research: Board	Research: Board	
↑ 6	↑ 6	↑ 6	↑ 6	↑ 6	↓ 0

Uge 3

uge 3					
mandag	tirsdag	onsdag	torsdag	fredag	weekend
Opdatering af tidsplan ----- Skriv på synopsen om Board komponent og dokumentering	Dokumentering af arbejde på board	Skriv på synopsen om board komponent og dokumentering	Dokumentering af arbejde på game logic	Skriv på synopsen om game logic komponent og dokumentering	
Arbejde på Board component	Arbejde på Board component	Færdiggørelsen af Board component	Oprettelse af Game-logic Component	Oprettelse af Game-logic Component	
Research: Board	Research: Board	Research: Board	Research: Game-logic	Research: Game-logic	
↑ 6	↑ 6	↑ 6	↑ 6	↑ 6	↓ 0

Uge 4

uge 4					
mandag	tirsdag	onsdag	torsdag	fredag	weekend
Opdatering af tidsplan	Dokumentering af arbejde på sekundært	Dokumentering af arbejde på sekundært	Skriv på synopsen om sekundært	Skriv på synopsen om sekundært	
Dokumentation af arbejde på game logic	Board	Board	Board komponent og dokumentering	Board komponent og dokumentering	
Oprettelse af Game-logic Component	Arbejde på sekundært Board component	Arbejde på sekundært Board component	Arbejde på sekundært Board component	Arbejde på sekundært Board component	
Research: Game-logic	Research: sekundært Board	Research: sekundært Board + Præsentation på skolen	Research: sekundært Board	Research: sekundært Board	
↑ 6	↑ 6	↑ 6	↑ 6	↑ 6	↓ 0

Uge 5

uge 5					
mandag	tirsdag	onsdag	torsdag	fredag	weekend
Opdatering af tidsplan	Skriv på synopsen	Skriv på synopsen	Skriv synopsen færdig	Sidste ændringer hvis nødvendigt, aflever synopsen	
Dokumentation					
↑ 6	↑ 6	↑ 6	↑ 6	↓ 0	↓ 0

Reserveret tid og tid brugt

Planlagte timer er baseret på mængden af fyldte dage minus afleverings dag. weekender er ikke talt med i timer men er med på skemaet da jeg forventer at bruge dem til at opveje for		
Sum af timer brugt	planlagte timer	dif
127	132	5
	22 fulde hværd dage * 6 timers arbejde	

Billede af Spil

