Kryds og bolle spil

Opdeling af programmet til ugerne.

1. Tegn brættet
2. Tegn brikkerne
3. Håndter musen
4. Lav logikken
5. Lav slut skærmen

Plan:

Planen over ugerne/møderne bliver måske ikke udført i samme tidsinterval af alle eleverne.

F.eks. er det meget muligt at nå både at tegne brættet og brikkerne på et enkelt møde.

De følgende noter skal måske gives til børnene direkte eller bare bruges som guideline.

Basis .py pygame fil

Dette er de basale ting enhver pygame spil skal indeholde. Øverst importere man pygame (men også sys siden sys skal bruges til at lukke vinduet).

Derefter “starter” man pygame op med pygame.init().

Så skal man også bruge et ur til at sørge for at tingene sker inden for det tidsinterval man vil have.

Sidst, men ikke mindst, definere man skærmen. Til at starte med definere man bredden og højden i pixels og bagefter laver man selve displayet.

|  |
| --- |
| import sys import pygame from pygame.locals import \*  pygame.init()   FPS = 60.0 FPSClock = pygame.time.Clock()  WIDTH, HEIGHT = 300, 300 SCREEN = pygame.display.set\_mode((WIDTH, HEIGHT))  def terminate():  pygame.quit() #Luk pygame  sys.exit() #Få systemet til at lukke de forskellige processer |

For at spillet skal kunne køre skal man også have en hoved løkke (mainloop) som man kalder det. Den skal blive ved med at gentage de ting der bare skal blive ved med at køre i vores spil. Den kunne se sådan ud.

|  |
| --- |
| while True:  #Håndter input  #Håndter hvad der skal ændres ved spillets data  #Tegn på skærmen |

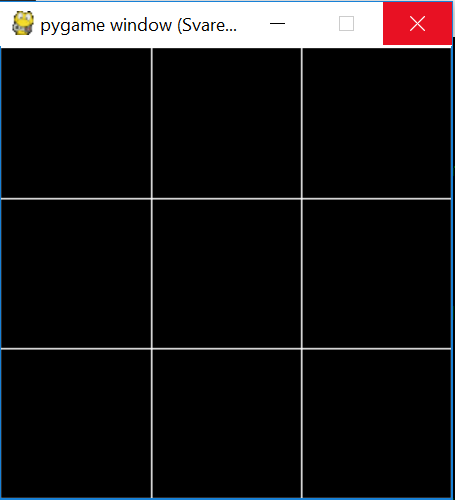
Tegn brættet

Brættet består af to horisontale linjer og to vertikale linjer.

man tegner en linje således:

|  |
| --- |
| pygame.draw.line(SCREEN, color, (x1, y1), (x2, y2)) |

Man skal nu tegne de fire linjer således:



Definer en funktion drawboard(), der tegner de fire linjer.

Tips:

* Det er en god ide at få defineret de x og y punkter der bruges hyppigt så koden er nemmere at læse og forstå.

Tegn brikkerne

I kryds og bolle er der to forskellige slags brikker, kryds og bolle. Dem skal man også have med i vores spil. Man starter med at tegne et kryds i øverste venstre hjørne. Kryds består af to diagonale linjer.

Derefter skal man tegne bolle. Den er lidt mere besværlig.

|  |
| --- |
| pygame.draw.circle(SCREEN, color, (x,y) r) |

Dog tegner pygame cirklen fra inderpunktet og så langt ud radius er. Så man skal finde midterpunktet af det gæ ldende felt.

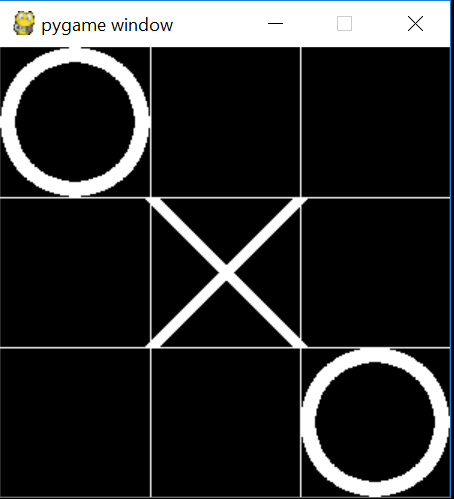
Nu skal man lave en funktion, den kunne f.eks. hedde drawPieces, hvor man kan tegne hele brættet ligegyldigt i hvilket felt der er placeret enten kryds eller bolle. Jeres bræt vil se således ud når det er tomt.

|  |
| --- |
| BOARD = [[None, None, None],  [None, None, None],  [None, None, None]] |

Kryds er True, Bolle er False.

|  |
| --- |
| BOARD = [[False, None, None],  [None, True, None],  [None, None, False]] |

Brættet ovenover vil således se sådan ud ↓.



Tips:

* Det kan være at indekset i loopet kan svare til de koordinater du definerede da du skulle tegne brættets streger.
* Brug BOARD som en parameter til funktionen drawPieces.

Håndter musen

Nu skal vi til at håndtere vores input. Hvad brugeren gør. I pygame bliver input såsom mus og keyboard bliver kaldet events. Man kan hente alle disse events ned gennem funktionen pygame.event.get(). For at håndtere inputtet til vores spil skal vi vide hvad det er vi vil kigge på, i kryds og bolle er det en god ide at bruge musen. Pygame inddeler de forskellige events i typer, hvor der er en række typer der alle loades ind som konstanter. Ligesom i det tidligere wormy.py skal vi sørge for at brugeren kan komme ud når brugeren trykker på krydset, denne type event hedder QUIT i pygame, når QUIT “aktiveres” skal programmet termineres. Musseklikket har typen MOUSEBUTTONDOWN, så skal musens position bruges af vores program. Musens position findes ved pygame.mouse.get\_pos() .

Når man først har musens position ved et klik skal man finde ud af hvor på “liste”brættet klikket svarer til og om der er en brik der i forvejen. For at tjekke dette er det godt at lave en funktion, man kunne kalde den validClick(). Denne funktion skal vide hvor brugeren har klikket og hvordan brættet ser ud lige nu. For at vide hvor klikket er henne kan vi bruge listen af koordinater, der beskrev vores tegnede spilleplade og se om klikket er i de gældende intervaller, den tegnede spilleplades koordinaters indeks passer sjovt nok med indeks i spilleplade listen. Når vi ved hvad koordinatet svarer til på spillepladelistens indeks kan vi se om pladsen er besat af en anden brik eller om den er fri og kan optages af den pågældendes spillers tur. Se hvordan det er gjort i mit kodeeksempel, link <https://drive.google.com/open?id=16KVuz9N9e3IXLobfjMzar5eFkdo_yj6s>.

Det er også først nu at begrebet tur bliver relevant, pas dog på da der kan opstå problemer i forhold til lokale variabler.

Lav logikken

Nu kan vi opdatere pladen så man reelt set kan spille kryds og bolle, men hvad godt er det hvis man ikke kan finde en vinder! Der er fire forskellige muligheder man skal tjekke mellem hver runde for at være sikker på at der ikke er en vinder eller at det står lige. Man skal tjekke for at man kan vinde horisontalt, vertikalt, diagonalt og at det kan være uafgjort. Diagonalt er der to muligheder, der skal tjekkes, vertikalt og horisontalt er der tre, og uafgjort skal hele brættet tjekkes for at der ikke nogle tomme felter. Det er en god ide for klarhedsskyld at samle det hele i en funktion den kunne hedde checkBoard(), derefter integrer den i spillets løkke.

Lav slutskærmen

På samme måde som i wormy.py er det en god ide at have en slutskærm så man ved hvem der vandt og spilleren kan spille igen. Lav en funktion der kunne hedde winScreen() hvori man får at vide hvem der vandt fra funktionen checkBoard() og tegn det på skærmen. Derefter så tjek for “hvilket” som helst input, så længe det hverken er QUIT eller ESCAPE, hvis der er et input og det ikke er QUIT eller ESCAPE så start et nyt spil.