Opgave/tutorial – skift mellem scener i Godot

For at kunne sikre en god dynamisk måde at skifte scener på i Godot, er man nødt til at benytte en parent node/parent scene til ens scener som står for selve sceneskiftet. Ved at gøre det på den måde vil det også fremadrettet være muligt at overføre data fra den ene scene til den næste uden at skulle lave en masse global-data som alle scener har adgang til (på store projekter vil programmøren let drukne i datamuligheder og performance af spillet vil falde grundet ekstra brug af hukommelse).

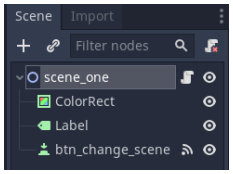
Koden til denne opgave er stærkt inspireret fra følgende tutorial fra nettet:
https://www.youtube.com/watch?v=XHbrKdsZrxY&t=179s

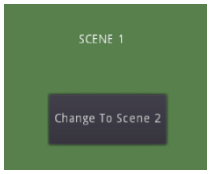
Opgave:

1. Opret et nyt projekt, navnet er frivilligt

2. Opret en 2D scene med et passende navn f.eks. ”scene_one”.

3. Tilføj tre ”Child notes” (ved at højreklikke på scenen) af typerne: ColorRect, Label og Button (en af gangen)

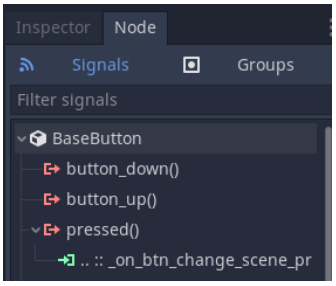
   

   Stræk ColorRect elementet ud så det danner en farvet baggrund.
   Sæt Label elementet midt i vinduet og skriv en kort tekst.
   Placer Button elementet midt i vinduet et sted og giv den ligeledes en sigende tekst f.eks. ”Change scene” og et navn.

4. Lev en nye scene der er magen til den første, kald den evt. ”scene_two”. Vælg en anden farve til denne scenes ColorRect.

5. De to scener skal nu tilknyttes til samme script, så højreklik på en af scenerne og tilføj et nyt script med et fornuftigt navn f.eks. ”level.gd”. Højreklik dernæst på den anden scene og tilføj det samme script.

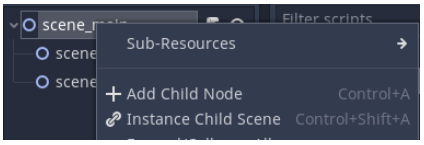
6. I scriptet skal tilføjes den funktion som kaldes når der klikkes på en af de to knapper. Den dannes lettest ved at vælge en af knapperne, og dernæst ”Signal” under fanebladet ”Node”.
   Dobbeltklik på ”pressed()” og Godot forslår et navn til funktionen hvilket er en sammensætning af knappens navn og det signal den reagerer på f.eks. _on_btn_change_scene_pressed
   Accepter navnet ved at klikke ”connect” og funktionen skrives nu ind i scriptet.
   (Der står pass som udgangspunkt i stedet for print(…) hvilket er tilføjet efterfølgende og ”level_name” beskrives senere)

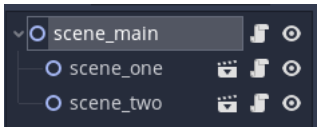
   


```
func _on_btn_change_scene_pressed():
    print("change from scene " + level_name)
```

7. Kopier navnet på funktionen og gør nu det samme for den anden knap. Hvis Godot forslår et andet funktionsnavn (hvilket den vil medmindre navnet på de to knapper er ens), paste navnet ind på den eksisterende funktion således at når der trykkes ”connect”, så vil begge knapper kalde den samme funktion. Hvis der dannes en ny funktion i scriptet, er der gået noget galt og man skal disconnect den nye funktion (under ”signals”) og prøve igen.

8. Der laves en tredje scene der gives et navn og tilføjes et nyt script f.eks. ”scene_main” og scene_main.gd. De to andre scener tilføjes nu som ”Child Scene” til denne scene ved at højreklikke på scenen og vælge ”Instance Child Scene”. I listen der vises, vælges de to scener således at de nu står som underscener.



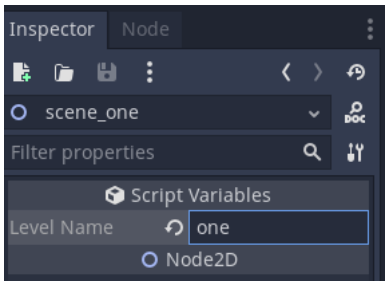


9. Det resterende der skal laves, er kode. I top scenens script (scene_main.gd) skal skrives en funktion der håndterer sceneskiftet og i underscene scriptet (level.gd) skal knapfunktionen vi tidligere lavede, kalde denne funktion.
   Disse to funktioner ligger ikke i samme script og derfor skal der benyttes nogle nye teknikker: signal og export (de er kun beskrevet kort her i opgaven).
   Signal bruges som når man klikker på en af knapperne ved at kalde en bestemt funktion. Ved at lave vores eget signal kan vi selv bestemme i vores script hvornår signalet skal sendes.
   Export benyttes til at gøre en variabel tilgængelig for flere noder eller scripts. Funktionen der laver sceneskiftet skal modtage en værdi (den har en input parameter) så derfor skal vi bruge en export variabel.
   Indsæt i toppen af level.gd scriptet (under extends…), følgende linjer kode →
   Default værdien af level_name kan sættes til hvad som helst, her er den sat til ”level”.

```
signal level_changed(level_name)

export (String) var level_name = "level"
```

10. Klik nu på den første underscene og tilføj et navn (f.eks. one) til Level Name der står til højre i ”inspector” under ”Script Variables”. Gør det samme for den anden underscene f.eks. med navnet two.
    Disse navne skal afgøre hvilken scene det er som kalder sceneskfitfunktionen.



11. Der tilføjes nu den kode til knapfunktionen der sender det signal der kalder sceneskiftfunktionen. Den kan sættes ind efter print(…) linjen således:

```
func _on_btn_change_scene_pressed():
    print("change from scene " + level_name)
    emit_signal("level_changed", level_name)
```

12. Sceneskriftfunktionen og en variabel skrives dernæst i scene_main.gd. Koden til variablen er:

```
onready var current_level = $scene_one
```

Den benyttes til at huske hvilken scene der er den aktive scene.
Sceneskiftfunktionen ser således ud:

```
func handle_level_changed(current_level_name: String):
    var next_level
    print(current_level_name + " handle")
    match current_level_name:
        "one":
            next_level = load("res://scene_two.tscn").instance()
        "two","first":
            next_level = load("res://scene_one.tscn").instance()
        _:
            return

    add_child(next_level)
    next_level.connect("level_changed", self, "handle_level_changed")
    current_level.queue_free()
    current_level = next_level
```

Funktionen modtager et input som er enten ”one” eller ”two” alt efter hvilken scene der har kaldt den. I kodelinjen match… der svarer til switch i mange andre sprog, bestemmes hvilken scene der nu skal skiftes til.
Scenen loades in i variablen next_level og tilføjes til scene_main som underscene med kommandoen add_child(…). (Den er allerede en underscene fordi alle scener der skal vises er gjort til underscene fra start af hvilket afviger fra den tutorial der blev benyttet til denne opgave).
”next_level.connect(…)” sørger for at funktionen ”handle_level_changed” kaldes når der sendes signalet ”level_changed” fra knapfunktionen, altså den sikre at vores knapper virker.
Dernæst frigives den nuværende scene så den ikke længere er aktiv og skiftet sker.
”current:level = next_level” skrifter variablens data omkring hvilken scene der er den aktive.
(Print(…) kan undlades (det kan den også i den anden funktion))

13. Som det sidste skal vi kalde funktionen _ready(…) i scene_main.gd for at sætte hvilken scene der skal startes med. Koden til det ser sådan ud:

```
func _ready():
    print("main_ready")
    handle_level_changed("first")
```

Det skal nu være muligt at starte vores projekt og scene_one vises. Når der trykkes på knappen skiftes til scene_two og tilbage igen når knappen på scene_two aktiveres. Det ser ca. sådan ud: