

Programmering Eksamen

Af Mads T.F. Knudsen

Indholdsfortegnelse

[Synopsis (12000 – 19000 tegn): 3](#_Toc36568789)

[Forblad: 3](#_Toc36568790)

[Abstract: 3](#_Toc36568791)

[Problemformulering: 3](#_Toc36568792)

[Funktionsbeskrivelse: 3](#_Toc36568793)

[Implantation: 3](#_Toc36568794)

[Tale Til Tekst: 3](#_Toc36568795)

[Databse: 3](#_Toc36568796)

[Åben hjemmeside: 4](#_Toc36568797)

[Start program: 4](#_Toc36568798)

[Handlinger: 4](#_Toc36568799)

[Flowchart 4](#_Toc36568800)

[Test: 4](#_Toc36568801)

[Konklusion: 4](#_Toc36568802)

[Bilag: 4](#_Toc36568803)

[Vigtig kode: 4](#_Toc36568804)

[Billeder: 4](#_Toc36568805)

# Synopsis (12000 – 19000 tegn):

## Forblad:

Mit programs navn er Hall, som er en reference til ”Im afraid I cant do that Dave”.

Dette projekt er lavet af Mads T.F. Knudsen.

Dato: 17 / 4 / 20

GitHub link:

<https://github.com/Mfknudsen/ProgrammeringsEksamen>

## Abstract:

Jeg har i dette projekt arbejdet med at lave en simpel computer assistent, som skal kunne forstå brugerens tale og kunne påføre forskellige handlinger, som f.eks. at åbne en hjemmeside og starte et program.

## Problemformulering:

Hensigt med produkt

## Funktionsbeskrivelse:

Hvad kan produktet

## Implantation:

Hvordan er produktet lavet + flowchart

### Tale Til Tekst:

### Databse:

Eftersom Hall starter med forskellige kommandoer, som er nødvendig og påvirker forskellige dele af dens systemer, men det er dog muligt for brugeren, at tilføje flere kommandoer. Disse kommandoer er gemt i en liste af forskellige ”dictionary”, hvor man sætter forskellige værdier til en nøgle. Jeg convertere så de forskellige nøgler og laver dem om til et ”array” af ”string”. Hver værdi til de forskellige nøgler er et ”array” af ”string”. Jeg omdanner de forskellige ”strings” til en enkelt ”string”, og når jeg har gjort dette for alle, bliver de gemt i et nyt ”array” og begge vil blive gemt i et object. Dette vil blive gjort for hver kommando.

Bruger så Unitys indbyggede funktionen kaldet ”JsonUtility.ToJson(’object’)”, som omdanner objekterne til Json tekst, som så bliver gemt i en tekstfil.

Når den skal læse hvad der er blevet gemt, så bliver funktionen ”JsonUtility.FromJson<’object’>(’inputString’)” brugt til at omdanne teksten som den aflæser fra tekstfilen, til de samme typer objecter, som da de blev gemt.

Når den så starter på ny, vil den lave en ny liste af kommandoer ud fra de aflæste objekters værdier.

Objektiverne som bliver til at gemme, er alle ”Serializable”, hvor Serialization er en måde, hvorpå Unity kan automatisk gemme og rekonstruere. Data kan normalt ikke gemmes direkte i Unity og objekterne er derfor nødvendige for at gemme og indlæse data.

### Åben hjemmeside:

Unity har indbyggede funktioner og metoder, hvorpå man kan åbne og kommunikere med forskellige hjemme sider.

Hall kan åbne hjemmesider ved at starte en ”Corotine” ud fra en ”IEnumerator”, som er en funktion der kan sættes på pause baseret på forskellige typer ”yield”, som f.eks. an være en ”bool”. Når Hall skal åbne en hjemmeside, så bliver den sat på pause indtil den får et resultat tilbage fra hjemmeside.

### Start program:

For at åbne et eksternt program, har Unity igen indbyggede funktioner og metoder, som kan bruges.

Efter en ny proces bliver skabt, bliver den givet et filnavn, som vil inkludere filtypen, så f.eks. ”Discord.exe”. Starter derefter en ny proces ud fra den skabte proces.

### Handlinger:

Flowchart:

## Test:

Virker produktet efter hensigt

## Konklusion:

Opsummering af projekts vigtige punkter

## Bilag:

### Vigtig kode:

### Billeder: