Prosjektoppgave Rapport

Kjell Randby Kristensen

Studentnummer: 202894

Intro

Jeg valgte oppgve 2. som er å lage et 2d spill med pygame. Spillet må ha en scoreboard som blir lagret til en json, csv eller xml fil. Videre må spillet minimum importere og bruke et kodebibliotek i tillegg til filhåndteringen. Basert på disse kriteriene, har jeg valgt å lage et spill som tester taste-hastighet. Jeg har tatt inspirasjon fra typetest.io og typeracer.com sine "typing-spill" til å bestemme hvordan dette spillet skal fungere.

Info

Dette spille ble laget med Python 3.8.3. og benytter følgende kodebiblioteker:

- datetime – 4.3
- json – (built-in-module. i.e. standard library)
- pygame – 1.9.6
- random - (built-in-module. i.e. standard library)
- statistics – 1.0.3.5

Hva planen var ved starten av prosjektet

Det finnes mange grafiske versjoner av "typing-spill", men jeg har valgt å fokusere mer på vanskeligheten i ordene, samtidig som jeg prøver å standardisere målingen av tastehastighet mest mulig slik at resultatene blir mest mulig rettferdige. Videre vil jeg at hovedfokuset i spillet skal være tastingen, derfor tillater jeg ikke brukeren å benytte musen utenom til å krysse ut programmet. På grunnlag av dette utgangspunktet vil jeg at spillet skal inneholde følgende egenskaper:

- **Meny** – Den første siden brukerene kommer til som opplyser om hvilke handlinger man kan gjøre. Denne vil vise hvilke taster man skal trykke for å starte spillet, gå til infosiden, gå til high score siden, og gå til high score by username siden.
- **Infosiden** – Side som gir generell info om spillet (hvordan det fungerer og hvordan vi måler tastehastigheten).
- **Type test** – Selve spillet. Her bør brukerene få ett ord som han skal taste. Dersom det er en skrivefeil, bør teksten bli rød. Når brukerene trykker space eller enter bør neste

ord komme opp dersom ordet er korrekt, hvis ikke korrekt bør ingenting skje. Det bør være en standard lengde på setningen (e.g. 10 ord) slik at det blir rettferdig konkurranse. Når brukerene er ferdig med å taste inn alle ordene bør brukerene se resultatet, og blir promtet om å skrive hvilket brukernavn resultatet skal lagres som. Deretter bør resultatet lagres i en json fil. Og brukerene blir sendt til high score siden slik at de kan se om de har fått en topp 10 plassering.

- **High score** – På high score siden skal de 10 beste bruker scorene etter WPM (words per minute) bli vist frem. I tillegg vil gjennomsnitt WPM samt gjennomsnittstid bli vist frem. Dersom det ennå ikke er noen high score data lagret bør brukerene bli promptet om dette.
- **High score by username** – Her bør brukerene kunne skrive inn ett navn, hvis navnet eksisterer bør statistikk om brukerene bli vist frem (antall spill, beste WPM, gjennomstnittlig WPM, beste tid, og gjennomsnittlig tid). Dersom det ennå ikke er noen high score data lagret bør brukerene bli promptet om dette.

Basert på disse egenskapene forutser jeg at jeg vil trenge følgende kodebibliotek:

- **Pygame** (for selve spillet)
- **Time** (for å ta tiden brukerene benytter)
- **Datetime** (for å displaye tiden på en pen måte)
- **Json** (for å lagre samt å hente high score data)
- **Random** (for å velge ut 10 tilfeldige ord)
- **Statistics** (for lettere å kunne regne gjennomsnitt av high scorene)

Jeg trenger også en ordliste som inneholder så mange engelske ord som mulig, da dette vil øke vanskelighetsgraden på ordene (de fleste "typing-spill" benytter små ordlister som består av de mest brukte ordene, noe som gjør det betraktelig lettere for bruker å skrive ordene). Jeg har tidligere erfaring ved bruk av nltk.corpus sin ordlister med 236736 engelske ord (som bør være nok til å øke vanskelighetsgraden betraktelig), men ettersom nltk.corpus krever manuell input for å hente ordlisten kommer jeg heller til å lagre ordlisten jeg henter fra nltk.corpus i en tekstfil, som jeg leser og gjør om til en python liste når brukeren starter programmet.

Lengden på vert ord, og dermed lengden på antall bokstaver i hver setning (10 ord) varierer mye (som et resultat av den store ordlisten). Her kan en test bestå av ti korte ord som utgjør 40 bokstaver, mens en annen test kan bestå av lange ord som utgjorde 100 bokstaver. Dermed ville en kalkulasjon basert på hvor fort de tastet de ti ordene ikke gjenspeile faktiske

hastigheten til brukeren. Derfor må jeg benytte en annen måte å beregne antall ord brukeren taster i minuttet, enn det som vanligvis benyttes. Jeg har funnet at lengden til et gjennomsnittlig engelsk ord inneholder 4.7 bokstaver (Medium, 2020). Jeg kommer til å sjekke hvor mange ord brukeren taster totalt, for så å benytte 4.7 tallet til å beregne hvor mange ord de har tastet, deretter vil jeg benytte tiden til å beregne hastigheten. Som et resultat vil ikke tiden i seg selv gi noen indikasjon på tastehastigheten, men ettersom det fortsatt kan være interessant å se hvor mye tid en gjennomsnittlig tastetest tar, planlegger jeg å vise frem denne informasjonen også.

Hva jeg fikk til og hva som var vanskelig

Jeg har klart å få til alt jeg var ute etter å inkludere i spillet, men ettersom dette er det første pygame spillet jeg har laget, støtte jeg på mange problemer i forhold til hvordan jeg hadde tenkt til å implementere de forskjellige elementene.

Det største problemet jeg støtte på var å forstå hvordan jeg skulle strukturere de forskjellige statusene/loopene i spillet (e.g. menyen, high score, infosiden osv.). Jeg prøvde først å neste uendelige looper, men det skapte problemer med hastigheten til applikasjonen. Jeg prøvde så å benytte en loop, og endre statusen til spillet gjennom å kalle på funksjoner, men dette var vanskelig å gjennomføre uten å kalle på globale variabler (noe jeg har hørt at en del programmeringsselskaper ikke tillater, så jeg valgte å ikke bruke denne teknikken). Til slutt endte jeg opp med å bruke en klasse for å representere spillet. Dette gjorde det mulig å endre på statusen som var attributter i "Game"-klassen innenfra klassen sine methods.

En annen utfordring jeg støtte på var å benytte time kode bilblioteket til å holde oversikt over tiden brukerene benyttet til å gjennomføre tastetesten. Etter mye feiling fant jeg ut at pygame lar deg starte en timer, ved å starte to timere, en for når spillet starter, og en for når spillet slutter, kan vi måle hvor mye tid som har gått ved bruk av ganske simpel logikk. Jeg benyttet Clear Code sin pygame timer tutorial til å forstå hvordan timere fungerer i pygame (Clear Code, 2020).

Det siste problemet jeg støtte på var å sorteringen av high scorene. Første gangen prøvde jeg å gjøre om hver bruker sin high score informasjon til en strenge som startet med wpm scoren, for så å sortere listen av strenger. Som man kan se på eksempelkoden under fungerer denne fremgangsmåten bare når det er like mange siffer i tallet på starten av strengen, men ettersom de fleste test scorer har et to sifferet nummer (gjennommsnittsiden på en test ligger på ca. 30

sekunder), så var det vanskelig å oppgdage feilen i koden, og ble ikke oppdaget før en test havnet over 60 sekunder.

```
In [8]: my_list = ['25text','55text', '90text', '34535text', '211text', '112text']
        print(my_list)
        my_list.sort()
        print(my_list)

        ['25text', '55text', '90text', '34535text', '211text', '112text']
        ['112text', '211text', '25text', '34535text', '55text', '90text']
```

Løsningen på problemet var å sortere scorene mens high scoren var lagret som en liste av dictionaries. Her kunne jeg enten benytte en lambda funksjon eller så kunne jeg importere itemgetter funksjonen fra operator modulen. Jeg valgte å benytte en lambda funksjon, for å begrense antall kodebiblioteker jeg benyttet, men jeg kunne likeså greit ha brukt itemgetter.

Hva jeg vil anbefale til andre studenter

Det jeg hadde mest nytte av var å spille spillet mitt, noe som gjorde at jeg fant feil i hvordan jeg hadde programmert spillet. Så jeg vil anbefale andre studenter å gjøre det samme.

Jeg hadde også stor nytte av å strukturere hele spillet teoretisk før jeg startet å programmere, dette gjorde at jeg fortere identifiserte problemet med å bruke nestede looper for de forskjellige statusene, og kunne begynne å identifisere løsninger på et tidlig tidspunkt.

Kilder

- Garbe, W. (2018, July 31). The average word length in English language is 4.7. Retrieved Oktober 24, 2020, from https://medium.com/@wolfgarbe/the-average-word-length-in-english-language-is-4-7-35750344870f
- Code, C. (2020, April 15). Python / Pygame Tutorial: Creating timers. Retrieved November 04, 2020, from https://www.youtube.com/watch?v=YOCt8nsQqEo