Dokumentation von Werkstück A: Buzzword Bingo

Ouail Annour, 1309124
Rachid Yagubi, 1359190
Muhammad Irfan Amin,1308138

Diese Dokumentation enthält alle Informationen bzgl. unseres Projekts Buzzword-Bingo. Sie enthält die Problembeschreibung bzw. Anforderungsanalyse, die Implementierung, eine eigene Bewertung der Implementierung, die Erfolge und Misserfolge der Implementierung, sowie die Gründe, weshalb wir uns für jene Implementierung entschieden haben. Zum Schluss folgt ein Fazit.

Inhaltsverzeichnis

Inhalt

1.Vorwort

Auf diesem Dokument sind Aufbau und Struktur unseres Projekts festgehalten.
Der Entwicklungsprozess und letztendlich die einzelnen Schritte werden hier dokumentiert, um das Entstehen unseres Projektes verständlich, sowie nachvollziehbar aufzuzeigen und zu erläutern.

2.Problembeschreibung

Im Rahmen der BS&RN Veranstaltung, soll das Spiel “Buzzword-Bingo”, grafisch in der Shell, im Einzelspieler-Modus implementiert werden. Dabei gibt es mehrere Anforderungen, die erfüllt werden mussten. Das Buzzword-Bingo muss in eine der folgenden drei Programmiersprachen implementiert werden:

- C
- Python oder
- Bash

Welche Programmiersprache wir verwenden, war dabei uns überlassen.

Nun folgt eine Auflistung der weiteren Anforderungen [1]:

- Die Benutzer sollen in der Shell, eine Bingokarte generieren können.
- Die Anzahl der Felder in der Breite und Höhe, der zu generierenden Bingokarte, gibt der Nutzer per Kommandozeilenargument an.
- Eine Export-Funktionalität ist notwendig (via Kommandozeilenargument) ist notwendig, um mehrere verschiedene Bingokarten zu erstellen und um dieses Ausdrucken zu können
- Die Werte der Felder auf der Bingokarte sollen aus einer Textdatei eingelesen und per Zufall verteilt werden. Es werden dementsprechend so viele Wörter aus der Datei entnommen, wie es die definierte Höhe und Breite vorgibt
- Mit der Tastatur (oder Maus) wählt der Nutzer das Wort aus, um es zu Markieren oder durchzustreichen
- Hat der Benutzer eine ganze Spalte oder Zeile oder Diagonale der Bingokarte markiert oder durchgestrichen, so gilt das Spiel als gewonnen, was auch kenntlich gemacht werden muss durch eine Laufschrift, ein Blinken oder durch Invertieren der Farbe in der Shell, etc.
- Das Spiel soll komplett in der Shell ablaufen
- Bei 5x5 oder 7x7 Feldern, bleibt das Feld in der Mitte frei (ein Joker wird platziert)

3.Allgemeines zum Buzzword-Bingo

Buzzword-Bingo Ist eine belustigende Abwandlung des herkömmlichen Bingo-Spiels, wobei Wörter benutzt werden, die Redner oder Vortragende in Präsentationen benutzen und zum eigentlichen Sinn der Rede nichts beitragen. Es können jedoch auch Wörter verwendet werden, die die Präsentierenden besonders oft benutzen. Wenn eine Person in einer Reihe, sei es nun vertikal, horizontal oder diagonal, alle Wörter gehört und markiert hat, ruft er laut ‚Bullshit'. Diejenige Person hat das Spiel gewonnen.

Buzzword	Buzzword	Buzzword	Buzzword	Buzzword
Buzzword	Buzzword	Buzzword	Buzzword	Buzzword
Buzzword	Buzzword	Joker	Buzzword	Buzzword
Buzzword	Buzzword	Buzzword	Buzzword	Buzzword
Buzzword	Buzzword	Buzzword	Buzzword	Buzzword

Abbildung 1.1 Aufbau einer 5x5 Buzzword Bingokarte

4. Unser Lösungsweg

4.1 Umsetzung des Grundspiels

Die Wahl der Programmiersprache fiel ohne langes Überlegen auf Python, da es eine leicht zu erlernende Programmiersprache ist aufgrund der einfach gestalteten Syntax. Zudem sind unzählige Module/Libraries vorhanden, welche einen Großteil der Arbeit übernehmen können.

Die Umsetzung des Grundgerüsts des Spiels Buzzword-Bingo, welche die Bingokarte als Dictionary, sowie dazugehörige Methoden beinhielten, verliefe ohne Probleme und konnte zunächst als abgeschlossen gesehen werden. Beim nächsten Schritt handelt es sich um die grafische Darstellung in der Shell.

4.1 Umsetzungswerkzeug

Für die Umsetzung des Grundgerüsts in eine grafische Darstellung, stand eine breite Auswahl an Modulen zur Verfügung.

Module, welche für die Umsetzung der Aufgabe in Betracht gezogen wurden, sind folgende:

- **Asciimatics**[2]
- **Curses**[3]
- **Blessed**[4]

Die Entscheidung fiel nach mehreren Prototypen dann schließlich auf **Blessed**, da dieses im Vergleich zu den anderen beiden Modulen, die effizientesten Möglichkeiten bot, Texte, sowie das Terminal selbst, auf eigene Wünsche und Vorstellungen anzupassen.

Zudem ist durch Blessed eine gemütlichere Eingabe für den Nutzer möglich, da der Cursor ausgeblendet werden kann und die Eingaben ohne Bestätigung eingelesen werden, um den Fluss des Spiels nicht unnötig ins Stocken zu bringen.

Für die farbliche Gestaltung der Texte, sowie der Bingokarte selbst, kam das Modul **termcolor**[5] zum Einsatz. Dadurch können Ausgaben mit verschiedensten Attributen versehen werden, um ihr Äußerliches anzupassen, und somit die Ausgabe auch visuell ansprechender zu gestalten.

5. Die Implementierung

5.1 Einleitung in das Spiel

Nachdem dem Nutzer die Spielregeln erklärt wurden, und dieser die geforderten eingaben getätigt hat, fängt das eigentliche Programm innerhalb einer von zwei While-Schleifen an.
So gibt es eine für den Mehr- und eine für den Einzelspieler.

Die Generierung der Bingokarte im Einzelspieler erfolgt durch folgende Methode (siehe Listing 1.1).

```
#Generierung einer Bingokarte
def generate_card(laenge,breite,line):
        card = {

            }
        for x in range(0,laenge):
            card[str(x+1)]=[random.choices(line,k=breite)]
        #Einsetzen der Joker in die Karte
        for letter in card:
            if letter=='2':
                if laenge==3 & breite==3:
                    card[letter][0][1] =colored('         x          ','grey','on_white')
            elif letter =='3':
                if laenge==5 & breite==5:
                    card[letter][0][2] =colored('         x          ','grey','on_white')
            elif letter=='4':
                if laenge==7 & breite==7:
                    card[letter][0][3] =colored('         x          ','grey','on_white')

        return card
```

Listing 1.1 Methode zur Generierung einer Bingokarte

Für die Grundlage der Bingokarte dient ein Dictionary.
In diesem wird mithilfe einer For-Schleife für jede Zeile ein zweidimensionales Array angelegt, welches mithilfe einer Methode aus dem Modul **random** gefüllt wird.
Diese Methode erlaubt es, als Parameter eine Liste, sowie die Anzahl der zu ziehenden Wörter anzugeben.
Im fortschreitenden Prozess werden nun die Joker in die Mitte der Karte eingefügt, sofern diese den dafür vorgegebenen Maßen (5x5, 7x7) entspricht.

Durch die folgende Methode (siehe Listing 1.2) ist eine saubere Ausgabe in der Shell möglich, da jedes Wort dadurch gleich viel Platz in der Karte einnimmt.

```
#Worte werden mit Leerzeichen verlängert, um eine saubere Darstellung zu garantieren
def card_order(card,laenge):
    for letter in card:
        for x in range(0,laenge):
            word=card[letter][0][x]
            length=len(word)
            if length<20:
                zahl=20-len(card[letter][0][x])

                wort=card[letter][0][x]
                card[letter][0][x]=wort+(zahl)*" "
```

Listing 1.2 Methode, welche jedem Wort einen gleich großen Platz in der Bingokarte verschafft

5.2 Beginn des Spiels

In jedem Durchlauf der While-Schleife in beiden Modis wird ein Wort gezogen, welches daraufhin aus der Liste gelöscht wird, um doppelt gezogene Wörter zu verhindern.

```
def draw(line):
   word_drawn = random.choice(line)
   line.remove(word_drawn)
   return word_drawn
```

Listing 1.3 Methode, um Wörter zu ziehen

Sollte ein gezogenes Wort in der Bingokarte vorkommen, so kann der Benutzer durch das Eingeben eines ‘x’ einen Prozess starten, bei welchem die Koordinaten für das durchzustreichende Wort abgefragt werden.

Für das Feststellen eines Sieges ist die folgende Methode (Siehe Anhang, S.1) zuständig, welche nacheinander alle Möglichkeiten für einen Sieg durchgeht.

Wenn der Nutzer gewonnen hat, so wird die Hintergrundfarbe im Terminal blau und ihm zum Sieg beglückwünscht.
Daraufhin kommt das Spiel zu einem Ende.

5.3 Umsetzung des Mehrspielers

Beim Mehrspieler werden keine Karten im Terminal ausgegeben.
Erforderlich ist im Mehrspieler nur die Ausgabe der gezogenen Wörter.

Somit besteht die einzige Aufgabe im Mehrspieler darin, die Karte in ein PDF zu exportieren, welche dann vom Nutzer gedruckt werden.

Verwirklicht wurde es mithilfe des Moduls ‘FPDF’ in einer For-Schleife:

```
185       #for-Schleife, zum Übertragen der Bingokarten in PDF-Dateien
186       for x in range(0,int(anzahl_spieler)):
187           pdf =FPDF()
188           pdf.add_page()
189           pdf.set_font('Arial', size=12)
190
191           card=bingo_karten[x]
192           card_order(card,int(laenge))
193           for letter in card:
194
195
196                   pdf.cell(200,10,txt=letter,ln=1,align='C')
197                   pdf.cell(200,10,txt='| '.join(card[letter][0]),ln=0,align='L')
198
199           pdf.output('Karte'+str(x+1)+'.pdf','F')
200
```

Listing 1.4 For-Schleife zum übertragen der Bingokarten in PDF-Dateien

6. Bewertung der Ergebnisse

6.1 Gute Design-Entscheidungen

Die Nutzung eines Dictionaries als Basis für die Bingokarten erwies sich als gute Entscheidung, da die einzelnen Zeilen der Karte so einem festen Schlüssel zugeordnet sind. Dadurch können diese ohne weiteres mit Hilfe dieses Schlüssels aufgerufen und für den weiteren Verlauf genutzt werden.

6.2 Herausforderungen und Schwierigkeiten

Mit dem Übertragen der Bingokarte auf eine PDF-Datei kam es zu einigen Schwierigkeiten.
So kam es dazu, dass die Methode (Siehe Listing 1.7), welche wir im Einzelspieler für eine ausgeglichene Ausgabe in der Shell verwendeten, nicht die gleiche Wirkung erzielte beim Übertragen in eine PDF-Datei.
Dies führte Dazu, dass keine saubere Ausgabe der Bingokarte auf der PDF-Datei erfolgte. Dennoch sind die einzelnen Wörter klar erkennbar, was den weiteren Verlauf des Bingo Spiels somit nicht zu stark behindert.

Ein weiteres Problem sind Duplikate in der Bingokarte selbst.
Die Implementierung einer Methode, Bzw. das Anpassen der Methode zur Generierung der Karte, um so etwas zu verhindern, gelang uns nicht. Dadurch kann es in manchen Fällen zu Karten kommen, welche einzelne Wörter mehrfach enthalten.

Leider konnten wir die Implementierung der Maus- oder Tastatureingabe nicht umsetzen, dennoch konnten wir dieses ausgleichen mit einer eigenen Umsetzung bzw. Idee und haben dies auch getan.

7. Fazit

Das Projekt Buzzword-Bingo wurde im Rahmen der BS&RN Veranstaltung verwirklicht. Wir haben die Programmiersprache Python benutzt und die Bibliotheken Blessed, FPDF und termcolor. Zusammenfassend können wir sagen, dass es Probleme in unserem Projekt gab und wir diese auf unsere Weise lösen konnten.
Trotz dieser Problematiken kam es insgesamt zu einem erfolgreichen Buzzword-Bingo, welches eine ansprechende Darstellung in der Shell genießen darf.

8.Literatur

1. Anforderungen und Problemstellung (Seite 3)
   https://www.christianbaun.de/BSRN21/Skript/BSRN_Paper_Vorlage.pdf
2. Offizielle Asciimatics Dokumentation:
   https://asciimatics.readthedocs.io/en/stable/
3. Offizielle Blessed Dokumentation:
   https://blessed.readthedocs.io/en/latest/
4. Curses Dokumentation:
   https://docs.python.org/3/library/curses.htm
5. Termcolor Dokumentation:
   https://pypi.org/project/termcolor/

```python
#Methode, welche die Gewinne nachprüft
def check_win(card,laenge,breite):

    win = False
    #Zunächst werden die horizontalen Gewinnmöglichkeiten geprüft

    for letter in card:
        cnt=0

        for x in range(0,int(laenge)):

            if card[letter][0][x]==colored('         x          ','grey','on_white'):
                cnt+=1

        if cnt==int(laenge):
            win=True
            return win

    #jetzt werden die vertikalen Gewinnmöglichkeiten geprüft
    for x in range(0,int(laenge)):
        cnt=0

        for letter in card:
           if card[letter][0][x]==colored('         x          ','grey','on_white'):
               cnt+=1
        if cnt==int(laenge):
            win=True
            return win

    # jetzt werden die diagonalen Gewinnmöglichkeiten geprüft
    for x in range (0,int(laenge)):
        y=0
        cnt=0
        for letter in card:
            if card[letter][0][y]==colored('         x          ','grey','on_white'):
                cnt+=1
            y+=1
        if cnt==int(laenge):
            win=True
            return win

    for x in range (int(laenge),-1,-1):
        y=int(laenge)-1
        cnt=0
        for letter in card:
            if card[letter][0][y]==colored('         x          ','grey','on_white'):
                cnt+=1
            y-=1
        if cnt==int(laenge):
            win=True
        return win
```

Listing 1.6 Methode zum Prüfen der Gewinne

```
53  #Worte werden mit Leerzeichen verlängert, um eine saubere Darstellung zu garantieren
54  def card_order(card,laenge):
55      for letter in card:
56          for x in range(0,laenge):
57              word=card[letter][0][x]
58              length=len(word)
59              if length<20:
60                  zahl=20-len(card[letter][0][x])
61
62                  wort=card[letter][0][x]
63                  card[letter][0][x]=wort+(zahl)*" "
64
```

Listing 1.7 Methode zum Ausgeben der Karten