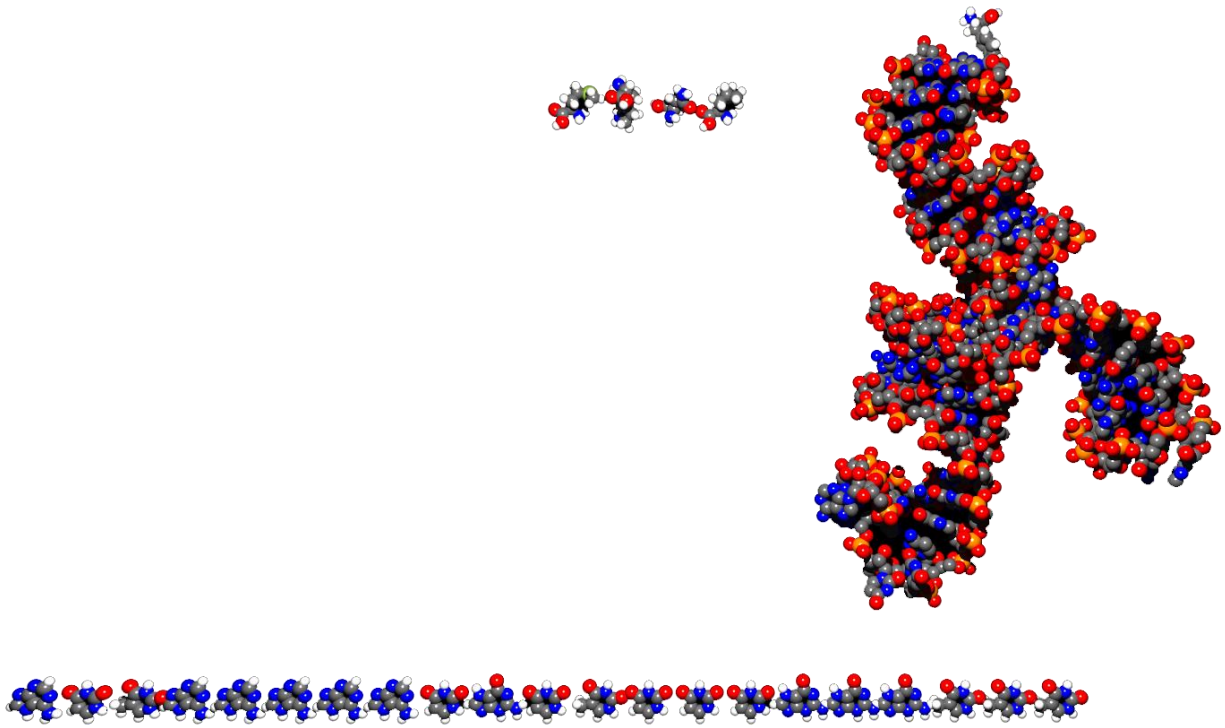
Visueel vertalen

Visualisatie van het translatieproces



Gijs Bakker & Niek Scholten
394545 & 388602
BFV1B
18-01-2019
Ronald Wedema

Visueel vertalen

Visualisatie van het translatieproces

Gijs Bakker & Niek Scholten
394545 & 388602
Bio-Informatica
Hanze Instituut voor Life Science & Technology
Ronald Wedema
18-01-2019

Samenvatting

Dit onderzoek heeft zowel medische als wetenschappelijke relevantie. De cellen in het lichaam van een organisme maken vrijwel constant nieuwe eiwitten aan, gemaakt van aminozuren. Dit proces heet Ribonucleïnezuur (RNA) Translatie, dat gebeurt door een zogenaamd Transfer-RNA (tRNA) molecuul te binden aan een RNA Triplet, waarna het tRNA een aminozuur loslaat en toevoegt aan een keten dat een eiwit vormt. Omdat dit proces erg vaak wordt uitgevoerd is de kans dus ook groot dat ergens hierin een fout op treedt. Daarom is het belangrijk dat er goed inzicht is in dit proces. Het doel was dus om een programma te schijven dat exact dit doet, het visualiseren van de translatie van RNA. Het programma kan een FASTA-bestand inlezen en een video genereren die laat zien hoe de translatie van dit specifieke stukje RNA op treedt. Zo kan de gebruiker goed zien hoe dit plaats vindt. Het programma voldoet aan de visie die voorheen bedacht was; het is flexibel, accuraat, uit te breiden als nodig, makkelijk om te begrijpen en uit te voeren. In de toekomst zou de onze code uitgebreid kunnen worden met een Peptidyltransferase eiwit, om zo nog accurater de situatie in de cel weer te geven en eventueel een functie om automatisch DNA om te zetten in RNA om vervolgens te verwerken in de video.

Lijst met afkortingen en symbolen

A)
- A = Adenine

C)
- C = Cytosine

D)
- DNA = Desoxyribonucleïnezuur

G)
- G = Guanine

M)
- mRNA = Messenger-RNA

R)
- RNA = Ribonucleïnezuur

S)
- SDL = Scene Description Language

T)
- tRNA = Transfer-RNA

U)
- U = Uracil

Inhoudsopgave

Inleiding

Het is belangrijk om het translatieproces middels het Peptidyl Transferase enzym te visualiseren, omdat het laatste proces is in het maken van een eiwit waar een kopieerfout kan optreden. En het visualiseren van dat proces kan ervoor zorgen dat mensen een beter inzicht in dit proces krijgen.
In dit onderzoek wordt dit proces gevisualiseerd door raytracing met behulp van Python, Povray & Pypovray. Waarbij de nadruk wordt gelegd op het tRNA die aan de mRNA string bindt en een aminozuur aan de keten toevoegt.

Theorie

In dit onderzoek wordt het translatieproces middels het Peptidyl Transferase enzym gevisualiseerd. Om dit te visualiseren wordt Povray gebruikt dat wordt aangestuurd door Python met behulp van de Pypovray library.

Translatie

Bij translatie wordt RNA met behulp van tRNA vertaald naar aminozuren om vervolgens een polypeptideketen te vormen. (Campbell, et al., 2018)
RNA bestaat uit 4 soorten nucleobasen;
Adenine (A), Cytosine (C), Guanine (G) & Uracil (U).
Drie van deze basen vormen samen een codon in het mRNA. Elk van deze basen kan binden met een andere base, A met U en C met G. Voor elk mogelijk codon is er dus een anticodon van tRNA, die aan een mRNA codon kan binden om een aminozuur los te laten, die vast wordt gemaakt aan de polypeptideketen om een eiwit te vormen.

Programma’s

Povray is een grafisch programma voor het maken van 3D-computer graphics en animaties. Het programma maakt gebruik van ray tracing en techniek om fotorealistische afbeeldingen te renderen. (Ltd., 2018)
Ray tracing is een techniek die gebruik maakt van licht ‘rays’ die vanaf een ‘camera’ getraceerd worden naar een object, om realistisch de textuur en doorzichtigheid van een oppervlak weer te geven. (Rademacher, 2018)

Materiaal

Gebruikte software:

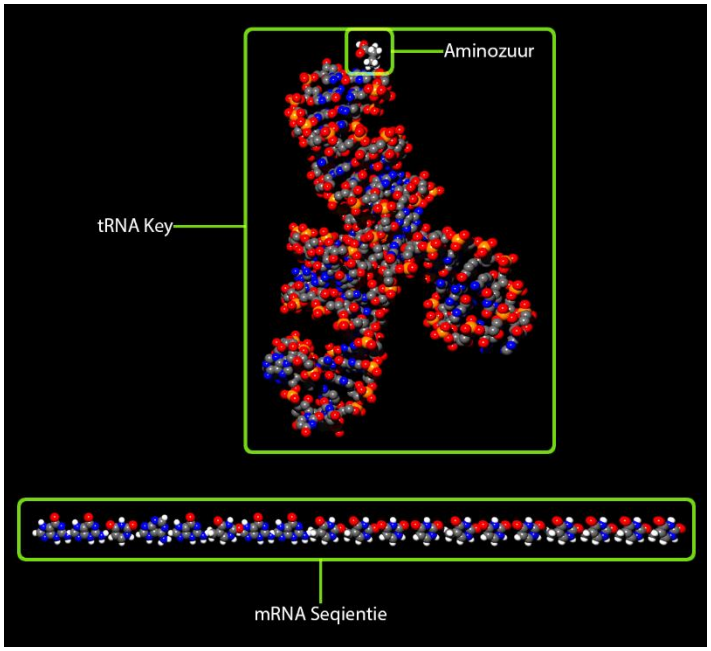
- Pyhton
- Vapory
    - Scene
    - Camera
    - Lightsource
- Pypovray
    - pypovray
    - SETTINGS
    - pdb
- Povray
- Math
    - pi
- FFMPEG
- sys

Methode

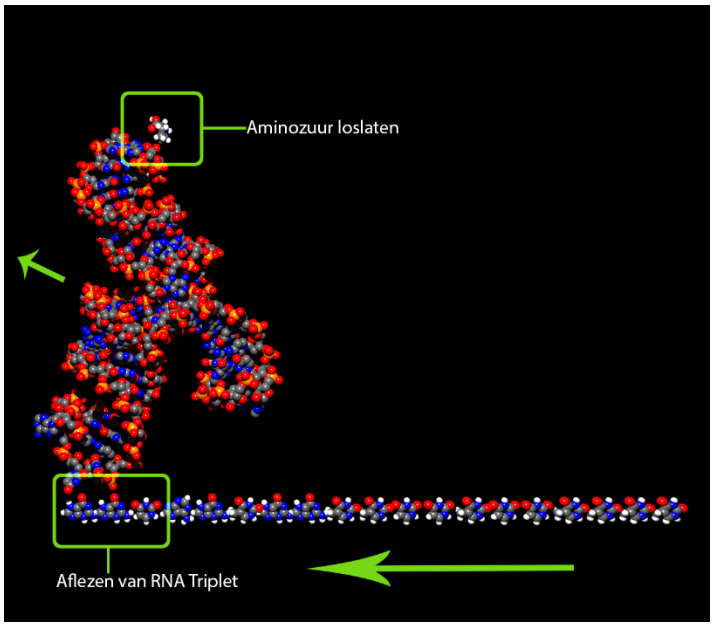
Om de code te laten werken werden eerst de boven genoemde functies geïmporteerd. Vervolgens was er een functie nodig om het, door de gebruiker gegeven, fasta bestand in te lezen, de nucleotiden eruit te halen en de nucleotiden in een string te zetten. Daarna werd er een functie gemaakt om 3d modellen van de nucleotiden te maken en deze op te slaan in een list.
Voor het maken van de 3d modellen van de aminozuren waren er drie functies nodig. Eerst moest er een functie geschreven worden om van de string van de nucleotiden een list van tripletten te maken. Vervolgens werd er een functie geschreven om van een triplet een 3d model van een aminozuur te maken, hiervoor werd ook een dictionary aangemaakt die als keys strings van tripletten heeft en als value de een letterige afkorting van aminozuren had. Om die twee functies samen te laten werken werd een derde functie geschreven. Deze functie roept voor elk triplet uit de triplet list de functie voor het maken van 3d modellen op en voegt deze modellen toe aan een list.
Vervolgens werd er een variabele key gemaakt, dat een 3d model van het tRNA molecuul bevat. Ook werd er een functie gemaakt om eenmalig de key in de goede rotatie te draaien.
Nadat dat gedaan was werd er een step afhankelijke functie voor het bewegen van het tRNA gemaakt. Deze beweegt het tRNA molecuul van rechtsboven, naar het midden, naar linksboven in het scherm.
Verder werden de twee functies die de aminozuren en de nucleotiden kunnen verplaatsen geschreven.
Tot slot werd de functie frame gemaakt. Deze functie plaatst alle modellen op de juiste positie en maakt dan de frames voor het filmpje. In deze functie worden de beweeg functies van de aminozuren en de nucleotiden aangeroepen afhankelijk van de positie en het nummer van de key zodat ook deze functies stap afhankelijk zijn.

Resultaten

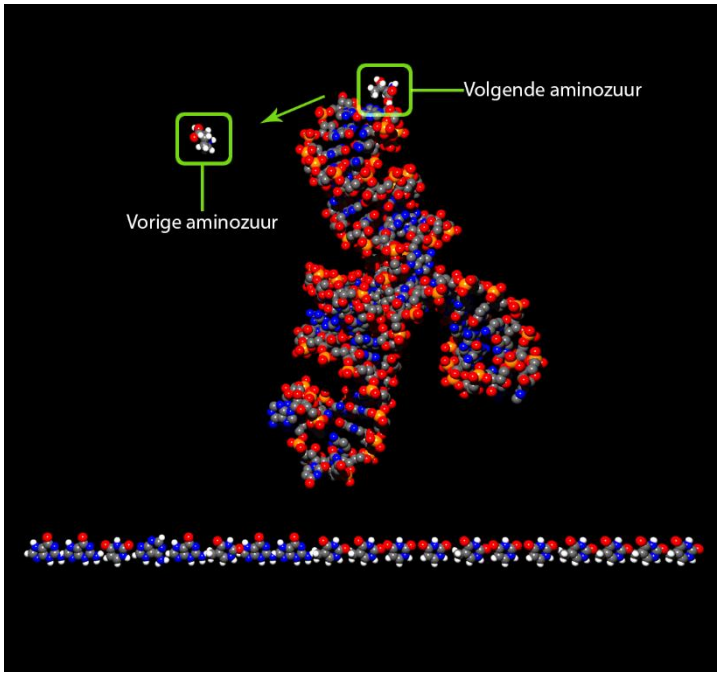
Als resultaat hebben wij een programma dat een video maak op basis van een FASTA-bestand. De algemene indeling van zo'n video wordt hier uitgelegd.



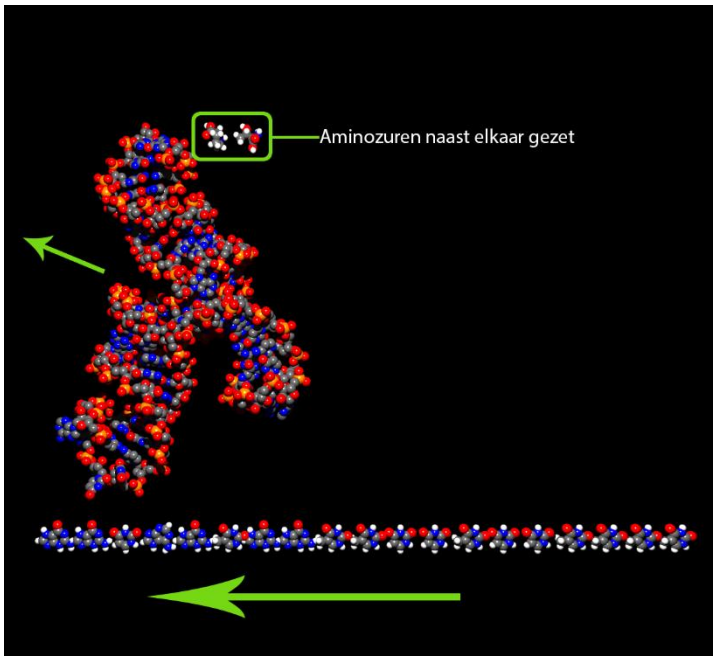
Stap 1
De elementen die ingeladen worden zijn:
Het aminozuur dat correspondeert met de desbetreffende triplet.
De tRNA Key die het aminozuur naar de goede plek leidt en langs het RNA gaat.
De mRNA sequentie, die verplaatst elke keer er een aminozuur wordt afgegeven.



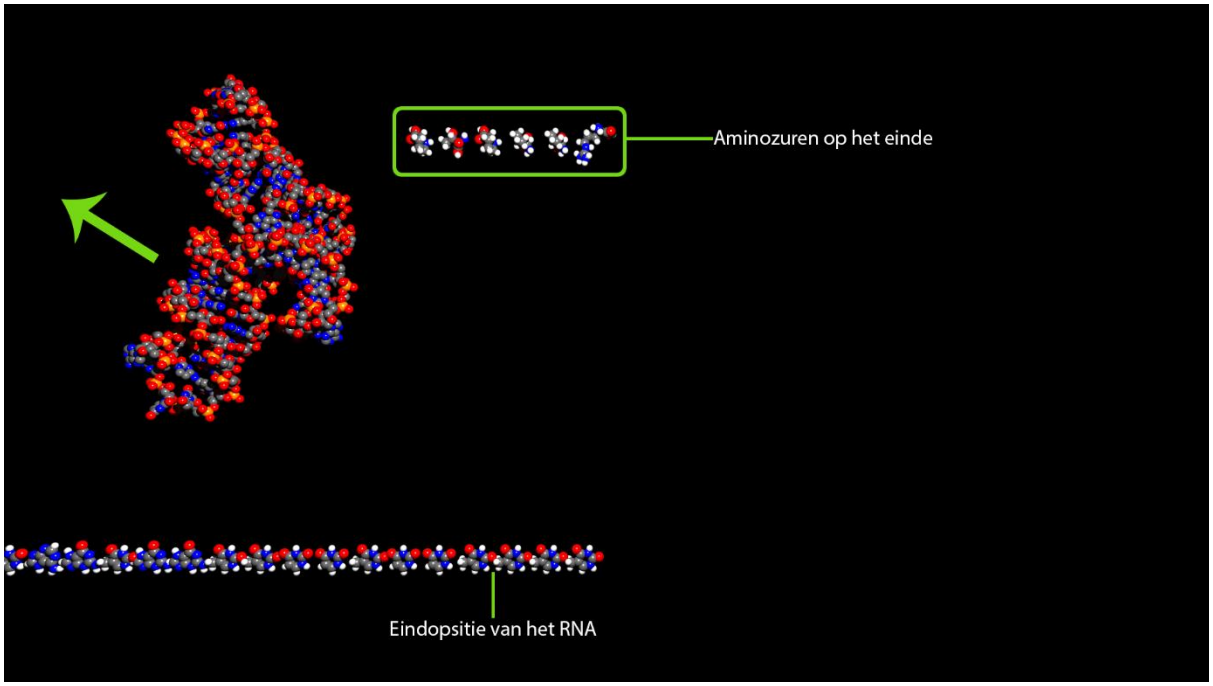
Stap 2
De key gaat langs de mRNA sequentie en laat het aminozuur los. Het aminozuur blijft op dezelfde plek staan en de key gaat de andere kant op, van het scherm af.
Tegelijkertijd verplaatst de mRNA sequentie zodat deze klaar staat voor de volgende key.



Stap 3
De volgende key wordt ingeladen en verplaatst richting de mRNA sequentie om daar het aminozuur af te zetten.



Stap 4
Het aminozuur dat er al stond wordt een stukje opgeschoven en de volgende wordt ernaast geplaatst. De key wordt weer van het scherm af geplaatst en de mRNA sequentie wordt opgeschoven.



Einde van de video

De mRNA sequentie staat nu op de laatste positie en de key gaat voor de laatste keer van het scherm af. Alle aminozuren die overeenkomen met de mRNA staan nu op een rij.

Conclusie en discussie

Er is een code geschreven die een animatie maakt van het translatie proces van mRNA. De code genereerd aan de hand van een gegeven fasta bestand en door de gebruiker aan gegeven instellingen een animatie met behulp van het programma Povray. In de animatie wordt de nucleotide string, uit het fasta bestand, vertaalt naar de bijbehorende aminozuur keten.
Deze code kan nuttig zijn voor mensen geïnteresseerd in dit onderwerp. Deze code kan namelijk het translatie proces van elke mRNA string animeren en door het translatie proces te zien, is het makkelijker te begrijpen. Het is echter discutabel of het zien van de translatie van een specifieke mRNA string, meer inzicht geeft dan een al bestaande animatie die het proces algemeen weergeeft. Het zien van de specifieke aminozuren kan namelijk inzicht geven naar de volgende stap, de eiwit vouwing. Aangezien de kijker de posities van de atomen kan zien. Maar het kennen van de vormen van de aminozuren zou dit ook al duidelijk kunnen maken.
Voor onderzoekers zou deze code niet veel waarde hebben aangezien zij waarschijnlijk al begrijpen hoe het translatie proces werkt. Ook kennen zij de vormen van de aminozuren waarschijnlijk al en zal de animatie voor hun dus ook niet meer inzicht geven in de eiwit vouwing.
De animatie en code zouden uitgebreid kunnen worden door de processen voor en na translatie ook toe te voegen. Bijvoorbeeld het vouwen van de eiwitten. Verder zou het peptidyl transferase enzym ook aan de animatie toegevoegd kunnen worden. Ook zouden er verschillende keys met de juiste anticodon voor de nucleotide string ingeladen kunnen worden. Al deze aanpassingen zouden ervoor zorgen dat de animatie realistischer wordt en er worden dan meer onderwerpen in de animatie behandelt. Hierdoor geeft de video extra inzicht in de aanmaak van eiwitten.

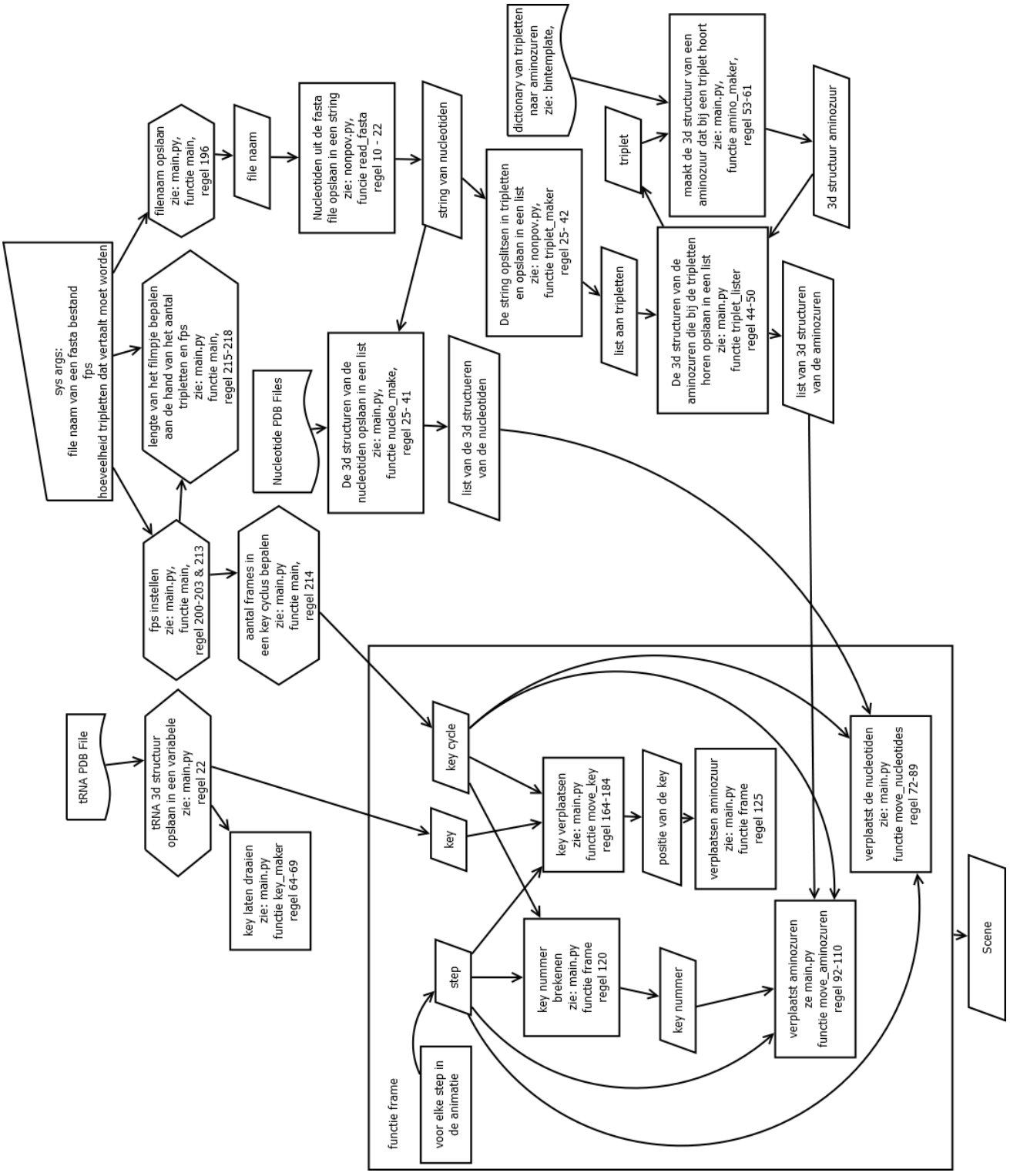
Bibliografie

Campbell, N. A., Reece, J. B., Urry, L. A., Cain, M. L., Wasserman, S. A., & Minorsky, P. V. (2018). *Biology A Global Approach.* New York: Pearson.

Ltd., P. o. (2018, 11 28). *What is POV-Ray?* Opgehaald van Povray: http://www.povray.org/documentation/view/3.6.2/766/

Rademacher, P. (2018, 11 28). *Ray Tracing: Graphics for the Masses*. Opgehaald van Department of Computer Science University of North Carolina at Chapel Hill: https://www.cs.unc.edu/~rademach/xroads-RT/RTarticle.html

Bijlages


sys args:
file naam van een fasta bestand
fps
hoeveelheid tripletten dat vertaalt moet worden
filenaam opslaan
zie: main.py,
functie main,
regel 196
file naam
Nucleotiden uit de fasta
file opslaan in een string
zie: nonpov.py,
funcie read_fasta
regel 10 - 22
string van nucleotiden
dictionary van tripletten
naar aminozuren
zie: bintemplate,
triplet
maakt de 3d structuur van een
aminozuur dat bij een triplet hoort
zie: main.py,
functie amino_maker,
regel 53-61
3d structuur aminozuur
lengte van het filmpje bepalen
aan de hand van het aantal
tripletten en fps
zie: main.py
functie main,
regel 215-218
De string opsiltsen in tripletten
en opslaan in een list
zie: nonpov.py,
functie triplet_maker
regel 25- 42
list aan tripletten
De 3d structuren van de
aminozuren die bij de tripletten
horen opslaan in een list
zie: main.py
functie triplet_lister
regel 44-50
list van 3d structuren
van de aminozuren
Nucleotide PDB Files
De 3d structuren van de
nucleotiden opslaan in een list
zie: main.py,
functie nucleo_make,
regel 25- 41
list van de 3d structueren
van de nucleotiden
fps instellen
zie: main.py,
functie main,
regel 200-203 & 213
aantal frames in
een key cyclus bepalen
zie: main.py
functie main,
regel 214
tRNA PDB File
tRNA 3d structuur
opslaan in een variabele
zie: main.py
regel 22
key laten draaien
zie: main.py
functie key_maker
regel 64-69
functie frame
key cycle
key
step
voor elke step in
de animatie
key verplaatsen
zie: main.py
functie move_key
regel 164-184
positie van de key
verplaatsen aminozuur
zie: main.py
functie frame
regel 125
key nummer
brekenen
zie: main.py
functie frame
regel 120
key nummer
verplaatst aminozuren
ze main.py
functie move_aminozuren
regel 92-110
verplaatst de nucleotiden
zie: main.py
functie move_nucleotides
regel 72-89
Scene

```
#!/usr/bin/env python3

"""
File met dictionaries
"""

__author__ = "Gijs Bakker, Niek Scholten"

AMINOTOTRIPLET = {"A": "GCC", "R": "AGA", "N": "AAT", "D": "GAT", "C":
"TGC",
                  "Q": "CAG", "E": "GAA", "G": "GGA", "H": "CAC", "I":
"ATT",
                  "M": "ATG", "L": "CTG", "K": "AAA", "F": "TTT", "P":
"CCA",
                  "S": "TCT", "T": "ACA", "W": "TGG", "Y": "TAT", "V":
"GTG",
                  "STOP": "TGA", "\n": ""}


TRIPLETTOAMINO = {"GCU": "A", "GCC": "A", "GCA": "A", "GCG": "A",
                  "UGU": "C", "UGC": "C",
                  "GAU": "D", "GAC": "D",
                  "GAA": "E", "GAG": "E",
                  "UUU": "F", "UUC": "F",
                  "GGU": "G", "GGC": "G", "GGA": "G", "GGG": "R",
                  "CAU": "H", "CAC": "H",
                  "AUU": "I", "AUC": "I", "AUA": "I",
                  "AAA": "K", "AAG": "K",
                  "UUA": "L", "UUG": "L", "CUU": "L", "CUC": "L", "CUA":
"L", "CUG": "L",
                  "AUG": "M",
                  "AAU": "N", "AAC": "N",
                  "CCU": "P", "CCC": "P", "CCA": "P", "CCG": "P",
                  "CAA": "Q", "CAG": "Q",
                  "CGU": "R", "CGC": "R", "CGA": "R", "CGG": "R", "AGA":
"R", "AGG": "R",
                  "UCU": "S", "UCC": "S", "UCA": "S", "UCG": "S", "AGU":
"S", "AGC": "S",
                  "ACU": "T", "ACC": "T", "ACA": "T", "ACG": "T",
                  "GUU": "V", "GUC": "V", "GUA": "V", "GUG": "V",
                  "UAU": "Y", "UAC": "Y",
                  "UAA": "#", "UAG": "#", "UGA": "#",
                  "UGG": "W"}

```

```python
#!/usr/bin/env python3

"""
Voor de functies van main.py die geen povray nodig hebben
"""

__author__ = 'Gijs Bakker, Niek Scholten'


def read_fasta(file_naam):
    """Leest een fasta bestnad in en slaat de nucleotiden op in een string
    Args: file_naam: de naam van de fasta file waar de nucleotiden uit
gehaald moeten worden.
    Return: Een string van nucleotiden."""
    file_obj = open(file_naam)
    nucleotiden = ''
    for line in file_obj:
        if line[0] != '>':
            line.strip()
            nucleotiden += line

    file_obj.close()
    return nucleotiden


def triplet_maker(nucleo_string):
    """Haalt uit een string van nucleotiden de tripletten en
    slaat deze strings van tripletten op in een list.
    Args: nucleo_string: een string van nucleotiden
    Return: een list van strings van tripletten."""
    count = 0
    tripletten = []
    triplet = ''
    for nucleotide in nucleo_string:
        if nucleotide != "\n":
            triplet += nucleotide
            count += 1
            if count == 3:
                count = 0
                tripletten.append(triplet)
                triplet = ''

    return tripletten

```

```
#!/usr/bin/env python3

"""
Animeert het translatie proces van mRNA
"""

__author__ = 'Gijs Bakker, Niek Scholten'

import sys
from math import pi
import nonpov
import bintemplate
from pypovray import pypovray, SETTINGS, pdb
from vapory import Scene, Camera, LightSource


nucleomoddel_list = []
triplet_list = []
key_cycle = 0
aminomoddel_list = []
AMINO_AFSTAND = 41
key = pdb.PDBMolecule('{}/pdbeind/tRNA.pdb'.format(SETTINGS.AppLocation),
center=True)


def nucleo_maker(nucleo_str):
    """Voegt voor elke nucleotide in de nucleotide string een 3d structuur
en voegt deze
    aan een list toe.
    Args: nucleo_str: een string van nucleotiden"""
    global nucleomoddel_list
    naam = ""
    for nucleotide in nucleo_str:
        if nucleotide == 'A':
            naam = '{}/pdbeind/Adenine.pdb'
        elif nucleotide == 'U':
            naam = '{}/pdbeind/Uracil.pdb'
        elif nucleotide == 'C':
            naam = '{}/pdbeind/Guanine.pdb'
        elif nucleotide == 'G':
            naam = '{}/pdbeind/Thymine.pdb'

nucleomoddel_list.append(pdb.PDBMolecule(naam.format(SETTINGS.AppLocation),
                                                    center=True))


def amino_lister(tripletten):
    """Voegt de 3d structuur van elk aminozuur in de tripletten lijst
    toe aan een globale list.
    Args: tripletten: Een list die strings van tripletten bevat."""
    global aminomoddel_list
    for triplet in tripletten:
        aminomoddel_list.append(amino_maker(triplet))


def amino_maker(triplet):
    """Genereerd een aminozuur bij een triplet.
    args: triplet: Het triplet waar het aminozuur bij ge genereerd moet
worden.
    return: De PDBMolecule van het aminozuur"""

```

```
62          locatie = '{a}/pdbeind/{b}.pdb'
63          aminozuur = bintemplate.TRIPLETTOAMINO[triplet]
64          amino = pdb.PDBMolecule(locatie.format(a=SETTINGS.AppLocation,
65      b=aminozuur), center=True)
66          return amino
67
68
69      def key_maker():
70          """Zet de 3d structuur van de key in de juiste rotatie."""
71          global key
72          key.rotate([0, 1, 0], 0.5)
73          key.rotate([1, 0, 0], 2*pi)
74          key.rotate([0, 0, 1], pi-0.6)
75
76
77      def move_nucleotides(step, cycle, key_nmbr, bewegend=True, afstand=7):
78          """Verplaatst de nucleotiden aan de hand van hoelang 1 cyclus van de
79          key duurt en welke cyclus momenteel bezig is.
80          Args: step: het nummer van het frame. cycle: het aantal frames dat de
81      key er over
82          doet om over het scherm te verplaatsen. key_nmbr: het nummer van de key
83      cyclus.
84          bewegend: een boolian die aangeeft of de nucleotiden moeten bewegen.
85          afstand: de afstand die tussen de nucleotiden moet zitten."""
86          x_pos = -17     # de x positie van de eerste nucleotide.
87          # De 1e step in de 2e helft van de cyclus als step 1 zetten
88          step2 = step - (cycle*key_nmbr) - (cycle/2)
89          afs_per_step = afstand / (cycle / 2) * step2 * 3    # *3 omdat het een
90      triplet moet opschuiven.
91          afs = afstand * key_nmbr * 3
92          if not bewegend:
93              afs_per_step = 0
94
95          for element in nucleomoddel_list:
96              element.move_to([x_pos - afs_per_step - afs, -42, -10])
97              x_pos += afstand
98
99
100     def move_aminozuren(step, cycle, key_nmbr, bewegend=True, afstand=7):
101         """Verplaatst de aminozuren aan de hand van hoelang 1 cyclus van de
102         key duurt en welke cyclus momenteel bezig is.
103         Args: step: het nummer van het frame. cycle: het aantal frames dat de
104     key er over
105         doet om over het scherm te verplaatsen. key_nmbr: het nummer van de key
106     cyclus.
107         bewegend: een boolian die aangeeft of de nucleotiden moeten bewegen.
108         afstand: de afstand die tussen de nucleotiden moet zitten."""
109         x_pos = 0
110         # De 1e step in de 2e helft van de cyclus als step 1 zetten
111         step2 = step - (cycle*key_nmbr) - (cycle/2)
112         # Berekent de afstand die per step afgelecht moet worden
113         afs_per_step = afstand / (cycle / 2) * step2
114         afs = afstand * key_nmbr
115         if not bewegend:
116             afs_per_step = 0
117
118         for element in aminomoddel_list:
119             element.move_to([x_pos - afs_per_step - afs, AMINO_AFSTAND, 0])
120             x_pos += afstand
121
122
```

```
123 def frame(step):
124     """Berekeent de posities van alle elementen van de animatie. En
125 simuleert de animatie.
126     Args: step: het nummer van het frame.
127     Return: een Scene"""
128     global aminomoddel_list
129 
130     # juiste triplet pakken key en aminozuur bij maken en key verplaatsen
131     key_nmbr = step // key_cycle  # key_nmbr = het nummer van de momentele
132 key cyclus
133     triplet = triplet_list[key_nmbr]  # welk triplet de key moet mee nemen
134     amino = amino_maker(triplet)
135 
136     pos = move_key(step, key_cycle, key)
137     amino.move_to([pos[0], pos[1] + AMINO_AFSTAND, pos[2]])
138 
139     camera = Camera('location', [0, 8, -140], 'look_at', [0, 0, 0])
140     light = LightSource([0, 40, -100], 2)
141     objects = [light]
142     move_aminozuren(step, key_cycle, key_nmbr, False)   # zet de aminozuren
143 op de juiste positie
144     move_nucleotides(step, key_cycle, key_nmbr, False)  # zet de nucleotide
145 op de juiste positie
146 
147     if pos[0] > 0:
148         if key_nmbr > 0:
149             # voegt alle amino modellen toe uit de aminomoddel_list
150             # tot aan het molecuul wat de key momenteel mee neemt
151             for i in range(key_nmbr):
152                 objects += aminomoddel_list[i].povray_molecule
153         objects += key.povray_molecule + amino.povray_molecule
154 
155     elif pos[0] == 0:
156         # voegt alle amino modellen toe uit de aminomoddel_list
157         # tot en met aan het molecuul wat de key momenteel mee neemt
158         for i in range(key_nmbr+1):
159             objects += aminomoddel_list[i].povray_molecule
160         objects += key.povray_molecule
161 
162     elif pos[0] < 0:
163         # voegt alle amino modellen toe uit de aminomoddel_list
164         # tot en met aan het molecuul wat de key momenteel mee neemt
165         # en verplaatst de nucleotiden en aminozuren
166         move_aminozuren(step, key_cycle, key_nmbr)
167         move_nucleotides(step, key_cycle, key_nmbr)
168         for i in range(key_nmbr+1):
169             objects += aminomoddel_list[i].povray_molecule
170         objects += key.povray_molecule
171 
172     for element in nucleomoddel_list:
173         objects += element.povray_molecule
174 
175     return Scene(camera, objects)
176 
177 
178 def move_key(step, nsteps, key_moddel, start_pos=(150, 50, 0),
179              eind_pos=(-200, 50, 50), mid_pos=(0, 0, 0)):
180     """Laat de key van rechts boven naar midden onder naar links boven
181 bewegen.
182     args: step: welke frame het programma bezig is.
183     nsteps: totaal aantal steps dat het duurt om van de beweging te maken.
```

```
184         key: Het PDBMolecule dat verplaatst moet worden
185         return: De positie van de center van de key"""
186 
187         cycle_step = step % (nsteps/2)
188         if step % nsteps < nsteps / 2:
189             # de volgende positie die in de cyclus aan genomen moet worden
190             pos_now = [start_pos[0] + (mid_pos[0] - start_pos[0]) / (nsteps /
191 2) * cycle_step,
192                        start_pos[1] + (mid_pos[1] - start_pos[1]) / (nsteps /
193 2) * cycle_step,
194                        start_pos[2] + (mid_pos[2] - start_pos[2]) / (nsteps /
195 2) * cycle_step]
196         else:
197             pos_now = [mid_pos[0] + (eind_pos[0] - mid_pos[0]) / (nsteps / 2) *
198 cycle_step,
199                        mid_pos[1] + (eind_pos[1] - mid_pos[1]) / (nsteps / 2) *
200 cycle_step,
201                        mid_pos[2] + (eind_pos[2] - mid_pos[2]) / (nsteps / 2) *
202 cycle_step]
203 
204         key_moddel.move_to(pos_now)
205         return key_moddel.center
206 
207 
208 def main(args):
209     """Maakt een animatie van het translatie proces van mRNA aan de hand
210 van de input.
211     args: 2e argument is de filenaam, 3e argument is hoeveel fps de video
212 gerenderd moet worden,
213     wordt er niets ingevuld dan wordt de fps uit default.ini gehaalt.
214     4e argument hoeveel tripletten er vertaalt moeten worden in de
215 animatie."""
216     global triplet_list, key_cycle
217 
218     # user data gebruiken
219     if len(args) >= 2:
220         file_naam = args[1]  # de mRNA sequencie die getransleert moet
221 worden
222     else:
223         print("You did not gave a file name")
224         return 0
225     if len(args) >= 3 and args[2].isdigit():
226         fps = int(args[2])      # fps
227     else:
228         fps = int(SETTINGS.RenderFPS)
229 
230     # verwerken
231     mrna = nonpov.read_fasta(file_naam)
232     triplet_list = nonpov.triplet_maker(mrna)
233     nucleo_maker(mrna)
234     amino_lister(triplet_list)
235     key_maker()
236 
237     # lengte filmpje en key cycle instellen
238     SETTINGS.RenderFPS = fps
239     key_cycle = fps*2
240     if len(args) >= 4 and args[3].isdigit() and int(args[3] <=
241 len(triplet_list)):
242         SETTINGS.NumberFrames = str(int(args[3]) * key_cycle)   #De lengte
243 van de video berekenen
244     else:
```

```
245             SETTINGS.NumberFrames = str(len(triplet_list) * key_cycle)
246
247         pypovray.render_scene_to_mp4(frame)
248         return 0
249
250
251     if __name__ == "__main__":
252         exitcode = main(sys.argv)
253         sys.exit(exitcode)
254
```