PROJECT 5/6: Autonome zeilboot

*Waïl Sahla*

*Thomas van Egmond*

*Khizer Butt*

*Amine Bouzelmat*

**Stuuralgoritme**

***Context***

Door een voorgaande projectgroep is er veel programmeerwerk gedaan, wat stuuralgoritme betreft. Zij hadden het voor elkaar gekregen om de zeilboot om een bepaalde koers te laten varen.

In principe horen er 3 modi te zijn in de zeilboot. De eerste modus is de ‘Remote-control mode’, waar de zeilboot volledig door een bestuurder bediend kan worden. De tweede modus is de ‘*Semi-autonomous mode*’. Daar is de koers gegeven en zal de zeilboot zelfstandig de koers van de roer afstellen om naar een bepaald doel varen. De derde modus is de ‘*Full-autonomous mode*’, hier wordt de roer en het zeil door de zeilboot zelfstandig afgesteld om een doel te bereiken. Bij de eerste twee modi wordt de zeilboot door een aparte motor aangedreven.

Voor een uitgebreidere uitleg over de opbouw en structuur van de technische details achter de koershouding, wordt er gerefereerd naar de Einddocument van de projectgroep ‘*B02*’. Voor algemene kennis kan de *Basis of Design* document beoogd worden.

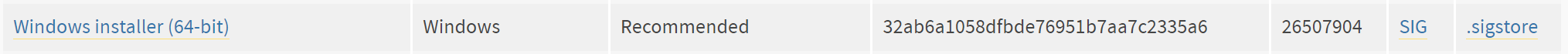
Op het moment werken alleen de ‘*Remote*’ en ‘*Semi-autonomous*’ modi. De bedoeling is dat het zeil nu ook een rol zal hebben tijdens het vaart. Bij alle windrichtingen moet het zeil zich zo aanpassen dat het zoveel mogelijk voortstuwende energie creëert voor de zeilboot. Als het met tegenwind te maken heeft moet de laveermanouvre plaatsvinden, om te voorkomen dat de boot niet stil komt te staan, of naar achter gaat.

***Opzet simulator***

Het gebruiken van een hulpmiddel dat het bewerken en testen van het programma vergemakkelijkt is zeer aangeraden. Er bestaat een simulator waar een zeilboot beïnvloed wordt door variërende windrichtingen en snelheden. Deze is door de projectgroep ook gebruikt.

In dit document zal de uitleg van de installaties niet plaatsvinden. Echter wordt er wel verwezen naar gebruikershandleidingen voor installatieuitleg, en waar de software vandaan gehaald kan worden.

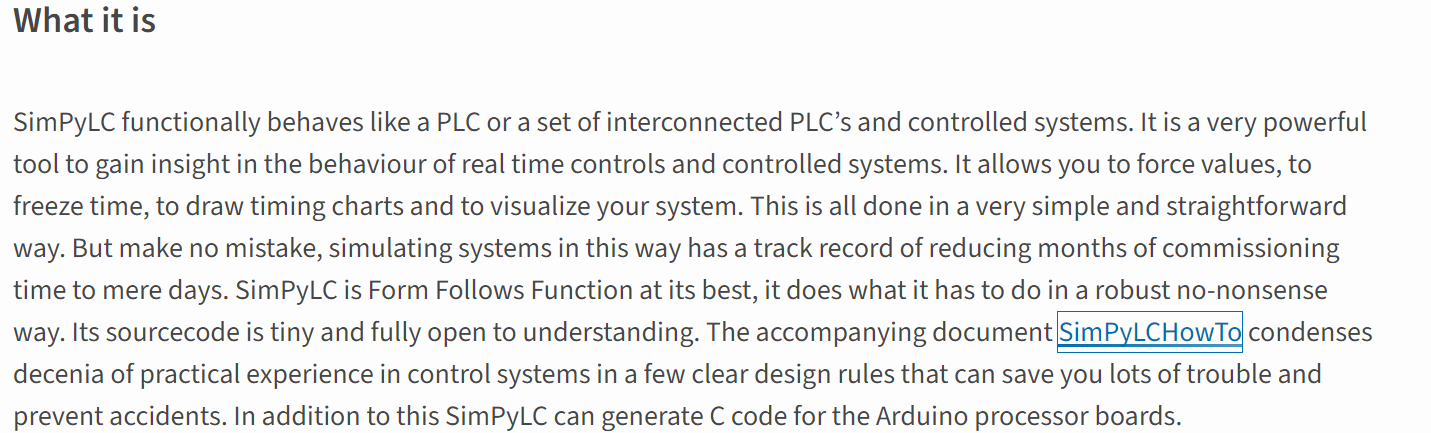
Voor het gebruik van de simulator is het nodig dat Python al klaarstaat op de computer. Hier kan Python gedownload worden: <https://www.python.org/downloads/>. Op het moment is Python 3.12.0 de stabiele versie. Na het kiezen van die versie zou de executable installer gedownload kunnen worden vanaf hier:



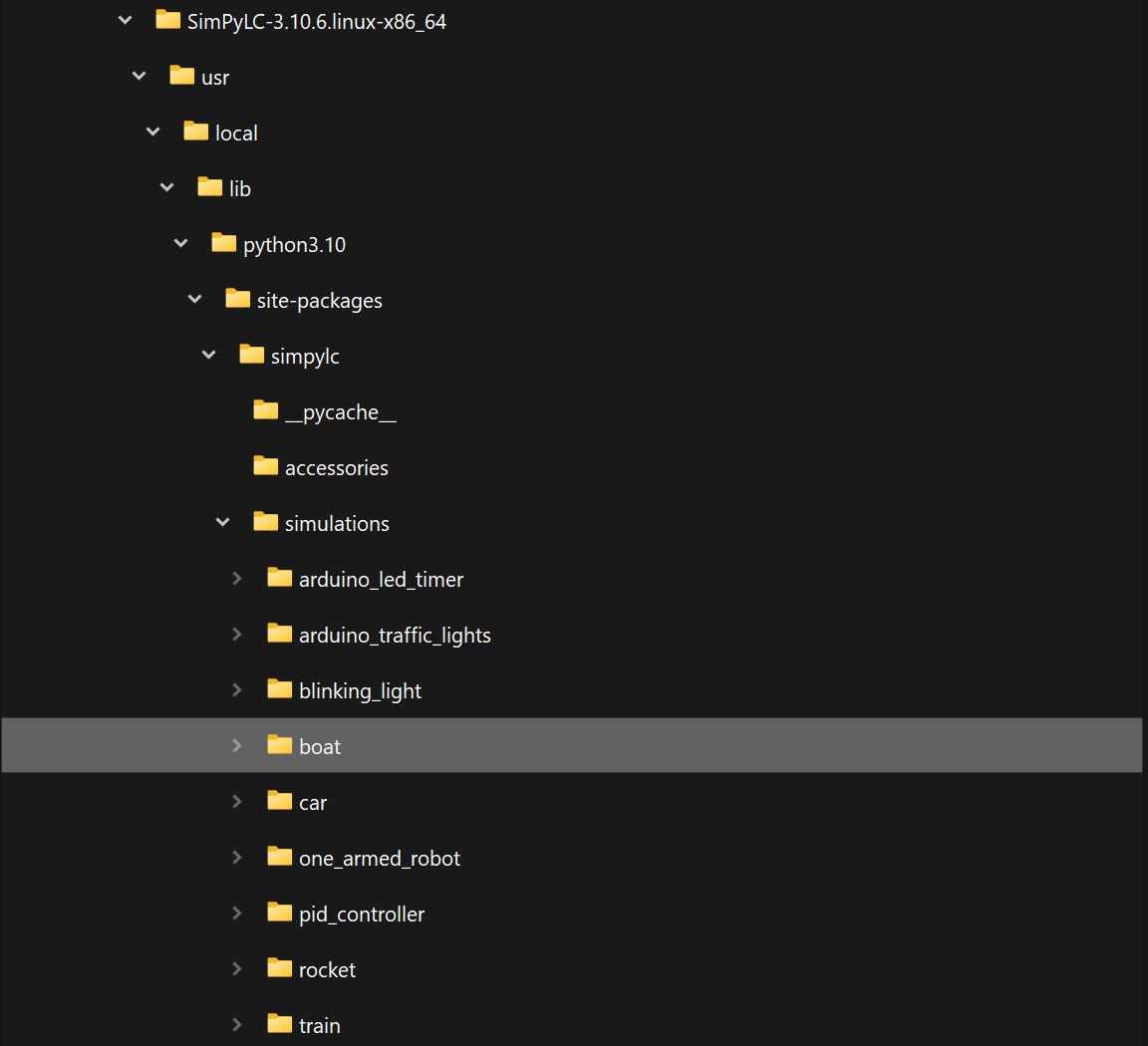
Nadere instructies voor het installeren volgen in de installer.

Nadat Python klaarstaat kan de simulator geïnstalleerd worden. Dit is de webpagina waar de simulator gedownload kan worden en waar verwezen is naar de instructies m.b.t. de installatie: <https://pypi.org/project/SimPyLC/#description>

Op de pagina kan de ‘*SimPyLCHowTo*’ link te vinden zijn, die naar de gebruikershandleiding toe gaat.

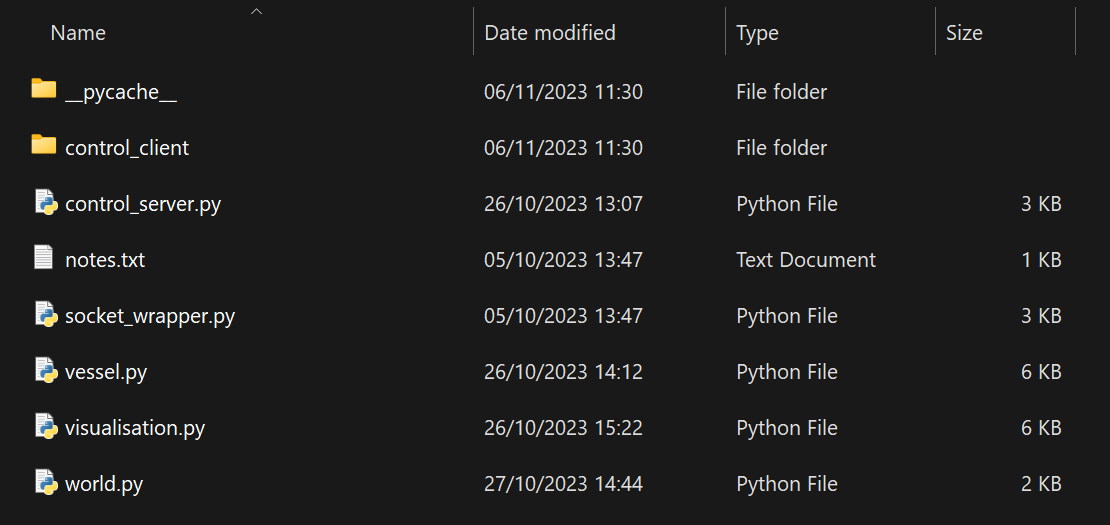


Nadat de simulator succesvol geïnstalleerd is kan er gebruik gemaakt worden van verschillende soorten simulaties. Hier zal de *‘boat*’ simulatie gebruikt worden. Hieronder is de mappenstructuur geïllustreerd, waarmee te zien is hoe er naar de bootsimulatie genavigeerd kan worden:

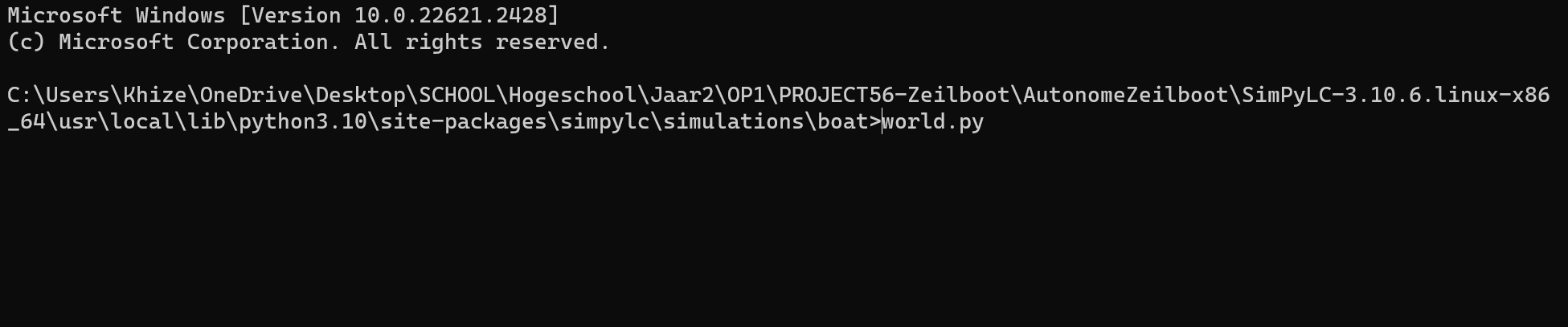


Uiteraard is tijdens de installatie een keuze gemaakt waar het ‘*SimPyLC-3.10.6.linux-x86\_64*’ map gemaakt zal worden.

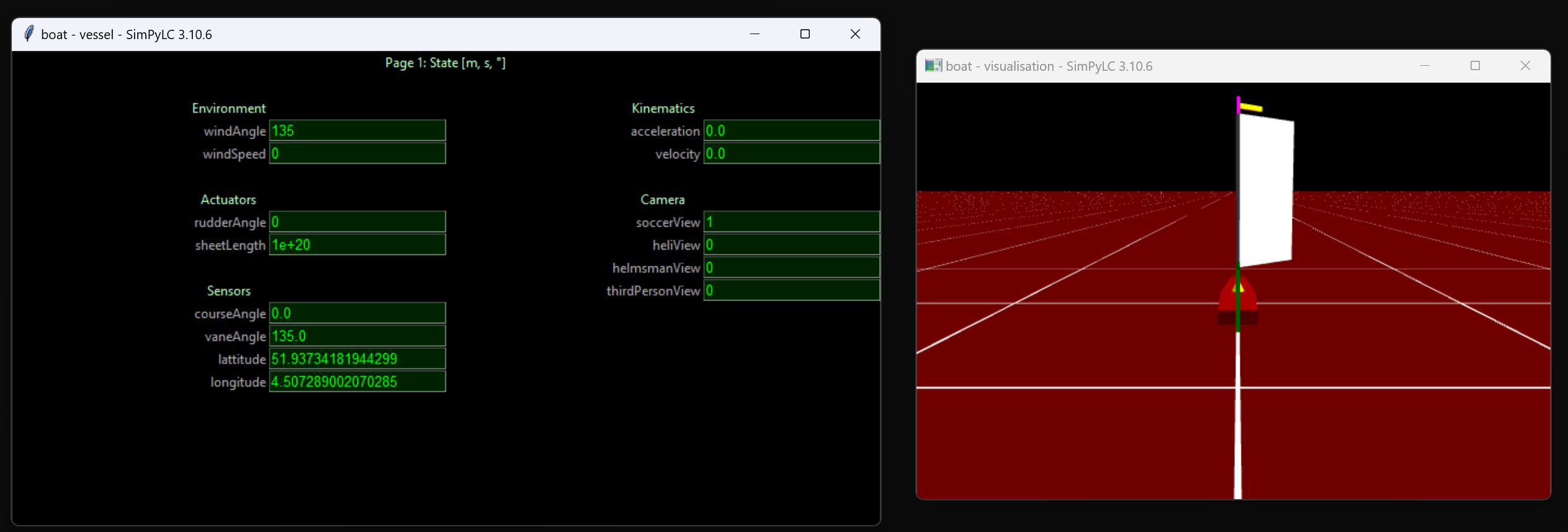
Vervolgens als de ‘*boat*’ map geopend is zal dit de inhoud ervan zijn:



Het Python bestand ‘*world.py*’ zal nodig zijn om de simulator op te starten. Om dat te doen moet er vanaf de command prompt genavigeerd worden naar het ‘*boat*’ map, om daar als volgt de ‘*world.py*’ bestand uit te voeren:



Zodra die commando uitgevoerd is horen de volgende vensters weergegeven te zijn:

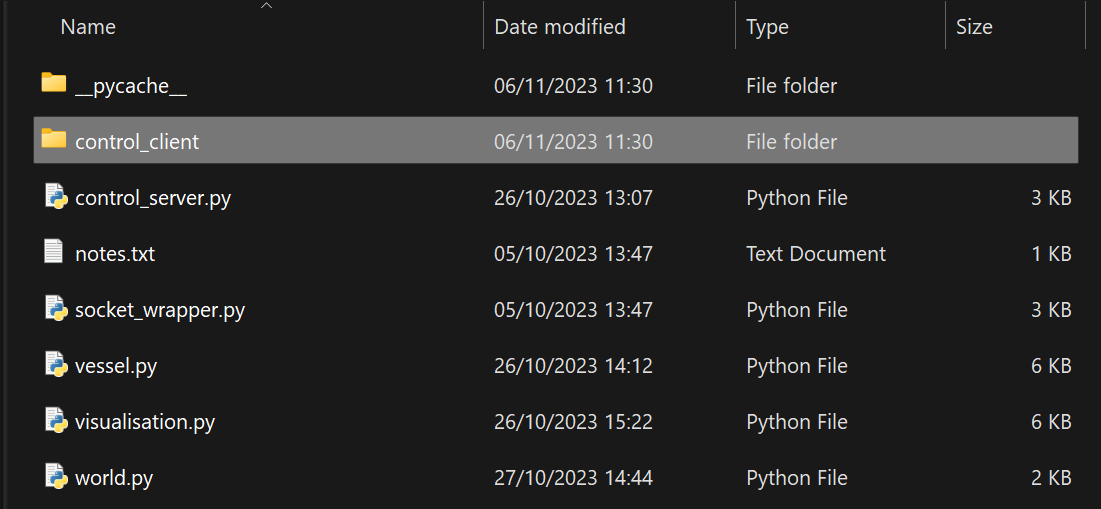
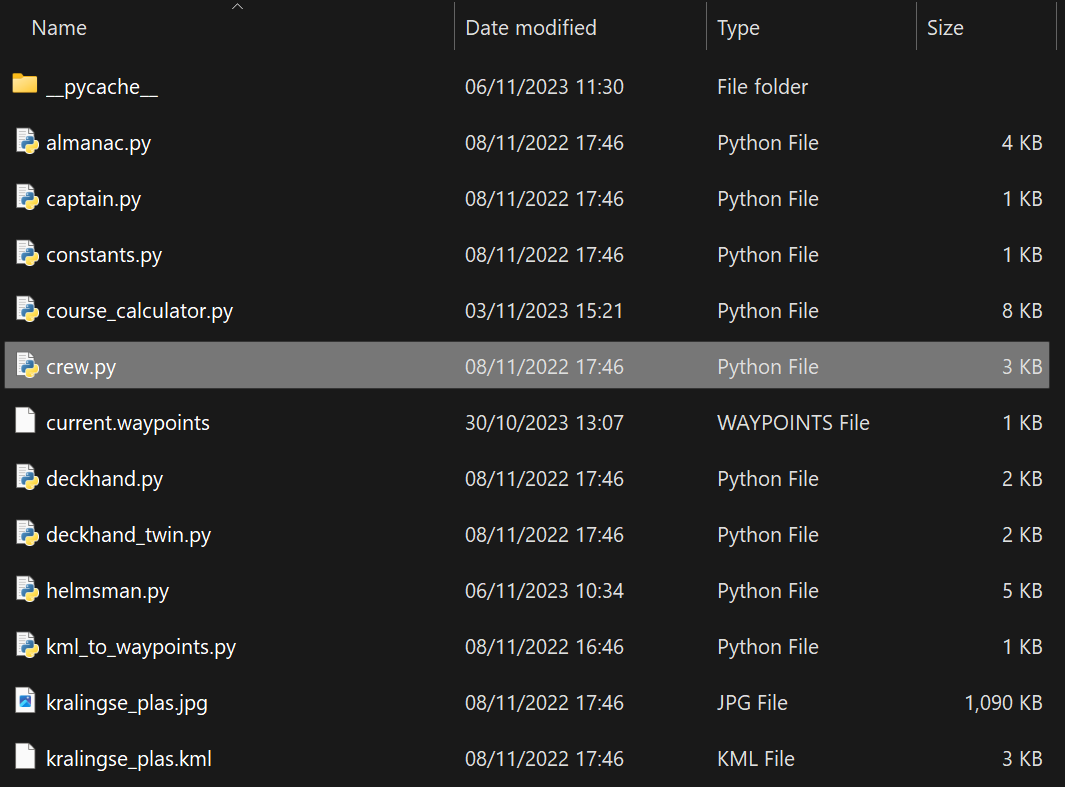


***Opmerking:*** *Als alleen de linkervenster te zien is, betekent dat de freeglut library niet geïnstalleerd is, of niet goed herkend wordt door de simulator. Zorg in ieder geval voor dat de naam van de dll-bestand, exact overeenkomt met de naam die vermeld is hieronder, en dat het in de aangegeven locatie is.*

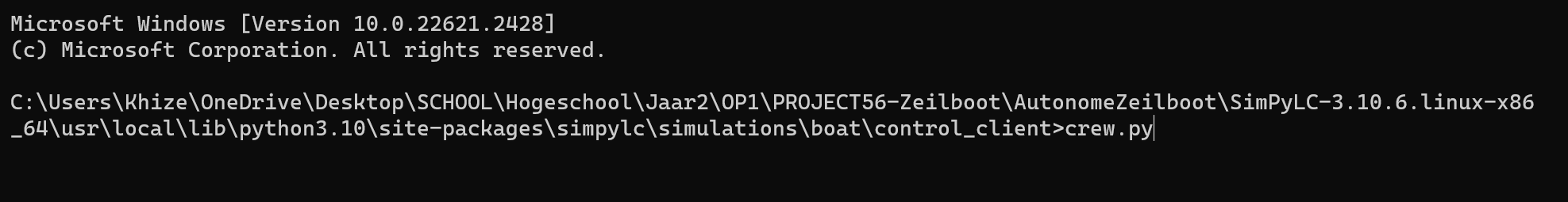
**

Als de bootsimulatie goed opgestart is kunnen we vanaf de linkervenster allerlij waardes toekennen aan parameters, zoals windrichting, windsnelheid, etc.

Voordat dat kan moet nog een python-script uitgevoerd worden. In de ‘*boat*’zit een ‘*control\_client*’ map waar de python-script ‘*crew.py*’ in zit, die uitgevoerd moet worden.

De ‘*crew.py*’ moet vanuit de command prompt uitgevoerd worden:



Als dat gedaan is kunnen de parameters aangepast worden en zullen de veranderingen in de simulatie te zien zijn.

Om een ingevoerde waarde in werking te brengen moet er na het invullen op ‘Esc’ getoetst worden.

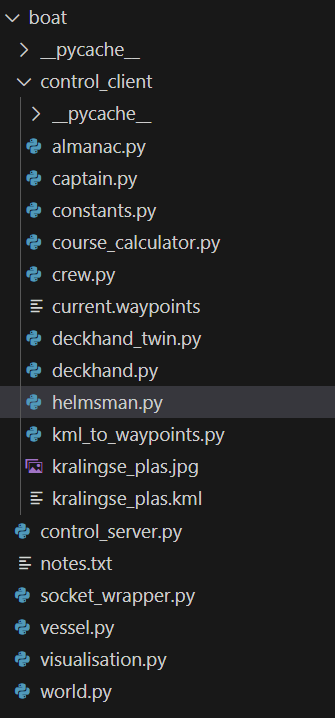
***Opzet werkomgeving***

Voordat er verder gewerkt kan worden, moet ervoor gezorgd worden dat de bestaande werk in de simulator gerealisereerd is. De code die de groep ‘*B02*’ geschreven heeft is op GitHub te vinden in de volgende repository: <https://github.com/wouterbt/RAAST/>

Navigeer naar de ‘*besturing*’ branch in de repo. Het bestand waar de code plaatsvindt is in de volgende locatie in de branch:

Het is aangeraden om al de code van de branch te downloaden, door de RAAST map over te nemen. Je hebt in GitHub de optie om de ‘RAAST’ code gezipt te downloaden. Zo kun je het altijd bij de hand houden door in Visual Studio Code twee workspaces te openen, ’boat’ en ‘RAAST-besturing’ mappen, wat het programmeren handiger kan maken.

Als de bootsimulatie uitgevoerd wordt, zal het gedrag van de zeilboot redelijk accuraat zijn vergeleken met het ‘echte wereld’, de zeil en roer stellen zich instantaan af om een bepaalde waypoint te bereiken, en de zeilboot voert de laveermanouvre correct uit. Er is gedacht om de programmeerwijze en de logica over te nemen van de simulator voor de echte zeilboot, echter kan dat alleen gedeeltelijk. Het programma in de simulator is zo gemaakt dat het niet het hardware elementen van de zeilboot in acht neemt. Het is dus suboptimaal voor fysieke toepassingen. Je wilt als het ware de ingebouwde stuuralgoritme overschrijven met RAAST code. Het idee is om de python-bestand ‘*Regeling.py*’, de stuuralgoritme van RAAST, in de simulatie te gebruiken om het roer te bedienen. Kopiëer de ‘*Regeling.py*’ bestand, en plaats de kopie in de *‘control\_client’* map.



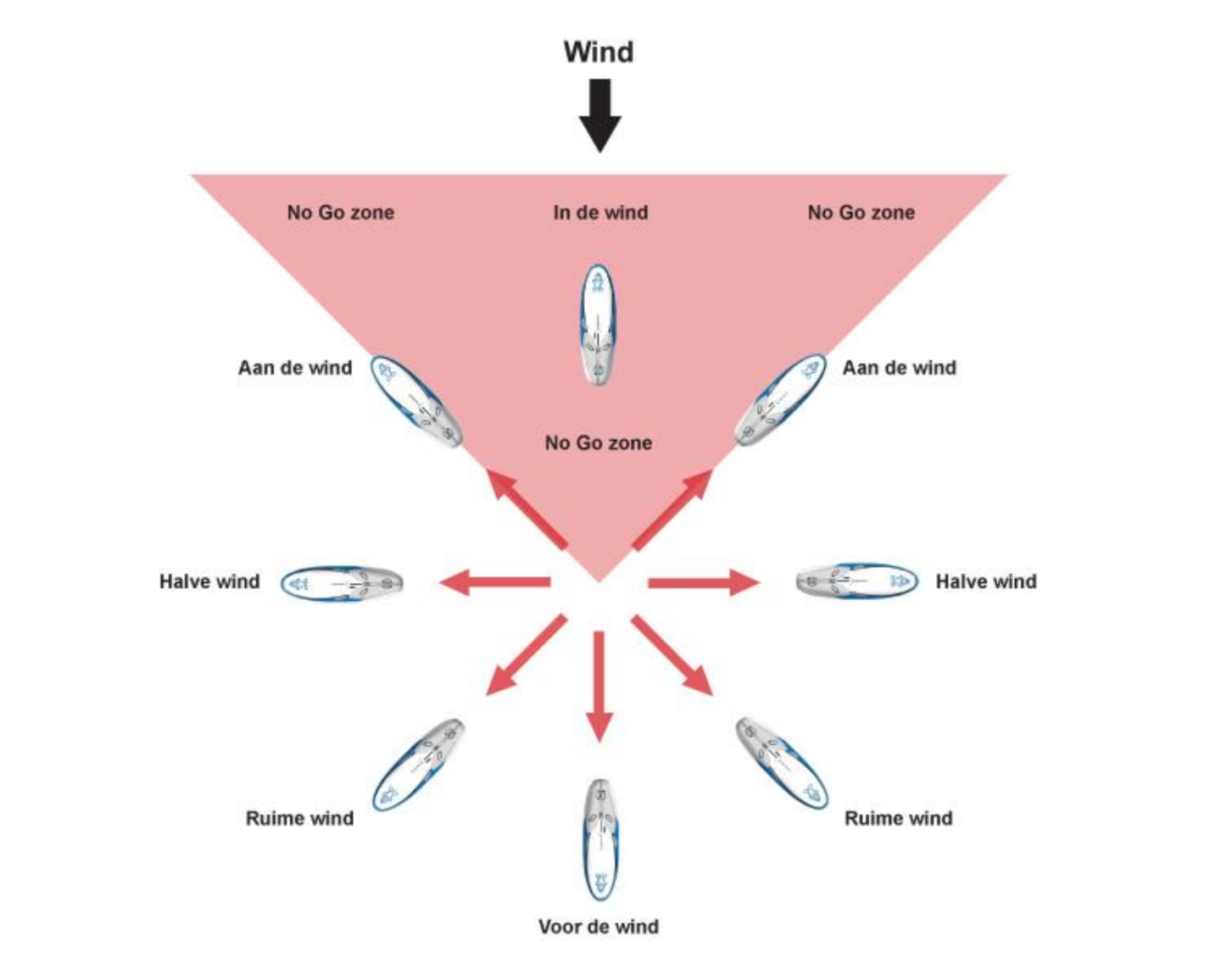
Er is een bestand in de simulator dat het besturing regelt van de digitale zeilboot. In de ‘*helmsman.py’* python-script zal de digitale zeilboot in de simulatie bestuurd worden:

***Uitleg laveermanouvre***

Er is al documentatie aanwezig over hoe de zeilboot moet gedragen in bepaalde situaties, in onder andere de Basis of Design en de Eindrapport gemaakt door groep *‘B02’*. Daarom zal hier niet een uitgebreide uitleg plaatsvinden over bepaalde termen.

In principe zal de zeilboot zich zonder problemen voort kunnen stuwen, als de wind niet vanuit een range van koersen komt.

Voor de zeilboot geldt er een ‘No-Go’ zone. Dat is een zone waarvan de koersen in die richting niet bezeild kunnen worden. Tijdens het zeilen kan een situatie ontstaan dat het koers tussen de zeilboot en het doel, enigszins hetzelfde is als de koers van de wind. Als de zeilboot in die richting probeert te varen, zal het in dat geval te maken krijgen met tegenwind en kan zich dan niet vooruit varen.

****

Tijdens zo’n situatie moet er een laveermanouvre uitgevoerd worden, om met tegenwind alsnog dichter bij het doel te kunnen komen. Tijdens het laveren zal er hoog aan de wind gezeild moeten worden (dicht bij de no-go range). Zodra het koers naar het doel niet meer onder de no-go range valt, gaat de boot *overstag*, dat houdt in dat de boot weer zoals gewoonlijk zijn gang gaat, en de directe koers naar het doel zal varen.

Aangezien de boot een flinke bocht maakt tijdens het overstag gaan, is er een kans dat de zeilboot tot stilstand kan komen door de wind. Om dat te voorkomen moet er ook extra snelheid gecreëerd worden, vlak voordat de boot overstag gaat. Om dat te doen moet de zeilboot voordat het overstag gaat, lager aan de wind (verder van de no-go range) varen.

Zie de *offset()* method in de *CourseCalculator* class. Daarin word al het laveerwerk inclusief de snelheidcreatie geregeld.

