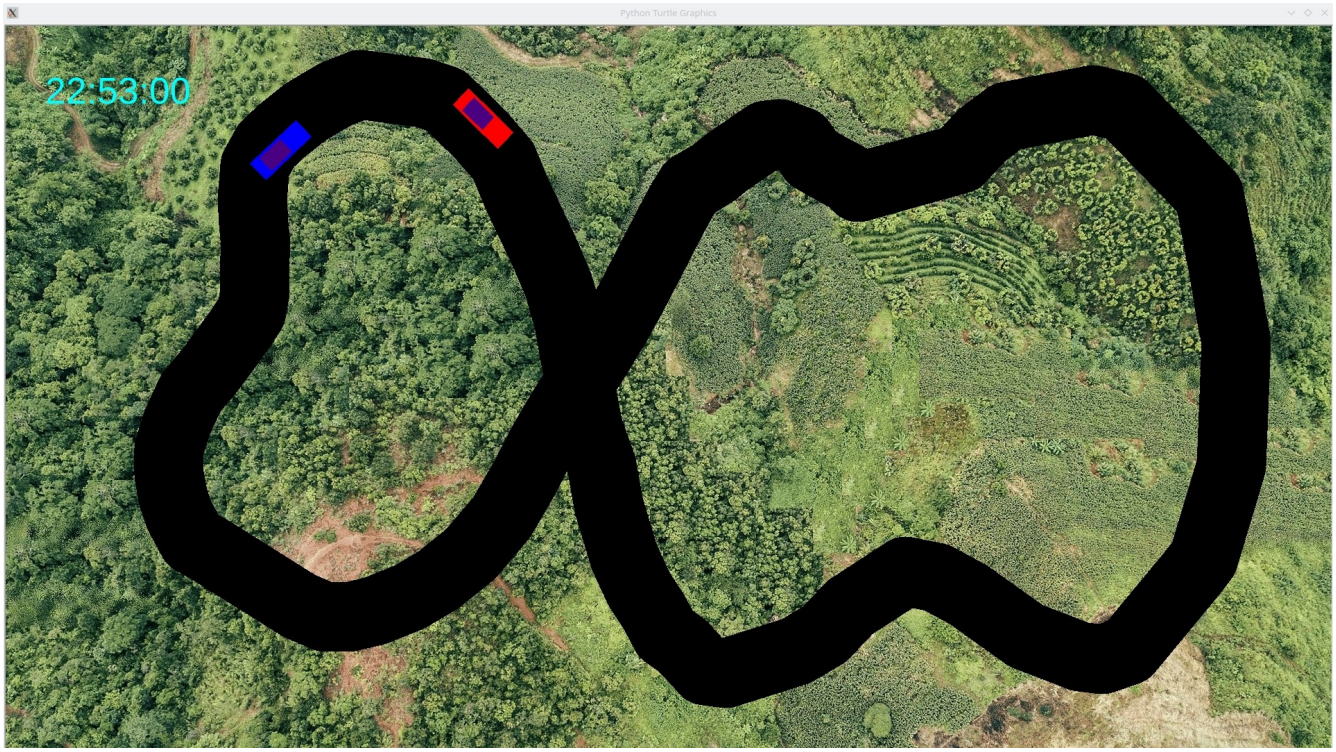
Eindopdracht keuzevak Python



Randvoorwaarden

Voor de eindopdracht mag je binnen je eigen groep samenwerken. Code overnemen van andere groepen of van derden is niet toegestaan. De assessment is individueel, zonder raadpleging van anderen binnen of buiten je groep.

Zorg dat je elke regel code die deel uitmaakt van de uitgewerkte eindopdracht volledig snapt en kunt uitleggen, dit is een voorwaarde voor succesvolle afronding van dit vak. **Bereid je assessment voor, zodat je er goed "in zit".** Alleen werkende code is dus NIET voldoende.

Voeg een klasse-diagram toe, bij voorkeur gegenereerd uit je code.

Maak een helder testplan met verifieerbare criteria. Test je applicatie grondig zodat je niet voor verrassingen komt te staan, leg het resultaat vast in een testrapport. Voeg dit toe aan wat je oplevert en houdt het bij de hand tijdens de assessment.

Technische hints

Voor de uitwerking van deze opdracht is het nodig, in een eindeloze lus te kunnen detecteren welke toets wordt ingedrukt, zonder de lus te onderbreken. Dit heet non-blocking console I/O. Een voorbeeld van hoe dit werkt vind je in het programmaatje *keypress_demo.py* in dezelfde

folder als dit document.

Op vergelijkbare wijze zijn mouse clicks te detecteren. Zie hiervoor de documentatie van de *turtle* library. De bedoeling is dat je geen 3rd party packages gebruikt. De kunst is juist om de opdracht te realiseren binnen de beperkingen van *turtle* en andere packages die deel uitmaken van de standaard Python distributie zoals die te downloaden is op *www.python.org*. Raadpleeg *turtle_intro.py* uit les 1 om te kijken hoe je meerder *turtle* objecten aanmaakt en gebruikt. Dit is nodig voor deze opdracht.

Onderwerp van de opdracht is een racebaan met auto's. Het gebruik van meerdere threads of processen is niet nodig. Wel moet het programma realtime zijn. Dit houdt in dat de snelheid en verplaatsing van de auto's niet mag afhangen van de processor-belasting. Dit kan worden opgelost door het tijdsinterval *deltaT* tussen twee updates van de positie van een auto te berekenen en dit als volgt te gebruiken om snelheid *v* en afgelegde weg *s* te berekenen uit versnelling *a*:

*v += a * deltaT*

*s += v * deltaT*

Roep in de hoofd-simulatielus uitsluitend eenmalig method *time ()* uit de standard module *time* aan om *deltaT* te berekenen, zoals getoond in *keypress_demo.py*.

Ontwerp- en programmeer-stijl

De opzet dient object-georiënteerd te zijn. Elke auto is een object van klasse Car en de racebaan is een object van klasse Track. Ook een eventuele rondetijd-display is een object. Al deze objecten samen bevinden zich in een object van klasse World. Er kunnen nog meer klassen en objecten zijn.

De hier gegeven benamingen zijn maar voorbeelden. Kies echter betekenisvolle namen. Namen als i, j en k zijn bij uitstek onwenselijk. De naam van een klasse of object is (of bevat) meestal een zelfstandig naamwoord (leftCar, Track, world), de naam van een functie is (of bevat) bijna altijd een werkwoord (run, drive, stop, getTime). De naam van een boolean is meestal een bijvoeglijk naamwoord, predicaat of een voltooid deelwoord (larger, isFirstCar, done). Wees nauwkeurig. Noem iets geen 'car' als het de index van een auto in een lijst is (maar carIndex). Kort geen namen af.

Indien een getal in je code meer dan 1 x voor komt in dezelfde betekenis, geef 't dan een naam en gebruik die. Plaats waar nodig commentaar om de motivatie achter ontwerpkeuzen te documenteren. Zorg echter vooral dat je broncode zoveel mogelijk "voor zichzelf spreekt".

Maak objecten zo zelfstandig mogelijk. Zo berekent en onthoudt elke auto z'n eigen snelheid en plaats op de baan, tekent zichzelf op de baan en kent zaken zoals z'n afmetingen en kleur. De baan houdt dus niet de positie van de auto's bij! Wel kan de baan bijvoorbeeld aan de auto's vragen waar ze zijn, om vervolgens een eventuele rondetijd-display hiervan op de

hoogte te stellen. De wereld weet niet hoe de baan in elkaar zit, dat weet de baan alleen zelf.

Deze aanpak leidt tot een flexibel ontwerp waaraan gemakkelijk zaken toe te voegen zijn, zoals meerdere auto's, pit-stations, signalerings-lampen, een parkeerterrein en een aparte fun-baan voor racelustige toeristen.

Niveau 1

Maak een short-track racebaan met 2 auto's. Elke auto is een rechthoekje, namelijk een *pen* of *turtle* met een custom *Shape.* De racebaan is een rechte zwarte strip van links naar rechts op je scherm. Nabij het einde van de racebaan is een gele finish-lijn.

Speler 1 accelereert / decelereert met keys *a* en *z*, speler 2 met keys *k* en *m.* De auto's hebben een maximale acceleratie, een maximale deceleratie en een maximale snelheid.

De auto's zijn niet bestuurbaar, maar volgen een vast spoor. De bedoeling is zo snel mogelijk de finish over te gaan zonder aan het einde van de baan af te schieten.

Extra: Zorg dat de cabine van een auto te onderscheiden is en ook de voor en achterruit, van bovenaf gezien. Gebruik hiervoor een *compound Shape* (zie *turtle* docs).

Niveau 2

Zelfde als bij niveau 1 incl. extra, maar de baan is nu een gesloten, vast circuit, op gebouwd uit minimaal 6 rechte stukken waarbij elk stuk loodrecht staat op het vorige. Elke auto blijft op z'n eigen weghelft.

Bij te snel passeren van de overgang tussen twee stukken vliegt de auto uit de bocht en belandt naast de baan. De andere auto kan verder rijden. Er kunnen meerdere rondjes worden gereden. De eis van het niet te ver doorschieten vervalt.

Extra: Er wordt een landschappelijke achtergrond getoond.

Niveau 3

Zelfde als bij niveau 2 incl. extra, maar de baan is opgebouwd uit een willekeurig aantal segmenten met willekeurige hoeken ten opzichte van elkaar. Elke auto blijft op z'n eigen weghelft. Indien een auto te hard door een bocht gaat (afhankelijk van scherpte van bocht) vliegt hij van de baan.

Voorafgaand aan de race kan door middel van mouse clicks een willekeurig circuit worden aangelegd, dat zichzelf kan kruisen.

Er is een real time clock zichtbaar in het grafische venster van de baan en een landschappelijke achtergrond. Deze maakt uitsluitend gebruik van de eerder via *time ()* opgevraagde tijd.

Extra: De klok laat in het grafische venster de rondetijden zien voor elke auto, zonder extra call naar *time ()*, of andere wijzen van toegang tot de systeem-klok.

Extra 2 (pittig): Als de baan zichzelf kruist, worden botsingen gedetecteerd. Deze leiden tot het van de baan vliegen van beide auto's.

Extra 3 (profi): Verschillende circuits kunnen onder naam (*'zandvoort'*, *'nuerenburgring'* etc.) worden opgeborgen naar en geladen van achtergrondgeheugen (zoals schijfgeheugen).