Eindopdracht "Zeeslagje" 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.

Hints

Begin zo eenvoudig mogelijk, pas versiebeheer toe, met git/github of simpelweg met folders op datum en bewaar alle werkende versies, zodat je niet met lege handen zit op je assessment. Daarna kun je de zaak optuigen (het gaat tenslotte over schepen) tot je er niet meer tijd in wil steken.

Deze werkwijze heet "timeboxing". Maak van te voren een "MoSCoW" (Must have, Should have, Could have, Won't have) analyse zodat je je prioriteiten helder hebt en niet steeds hoeft na te denken over "what's next". Bewaar die analyse om tijdens de assessment te laten zien dat je systematisch gewerkt hebt.

Als je demo complex is, maak dan een filmpje *(ten hoogste 5 minuten)*. Dan kan het demonstratie-duiveltje geen roet in het eten gooien.

Ontwerp- en programmeer-stijl

De opzet dient object-georiënteerd te zijn. De scheeps-klassen Patrolboat, Cruiser en AircraftCarrier erven van klasse Ship. Elk schip kent haar eigen positie en kan vertellen wat voor schip zij is. Er is een object van klasse Sea en een object van klasse (Game)Engine. Ook een eventuele score-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 (AircraftCarier, redSea, hmsPotemkin), de naam van een functie is (of bevat) bijna altijd een werkwoord (start, shoot, tellNameAndKind, getScore). De naam van een boolean is meestal een bijvoeglijk naamwoord, predicaat of een voltooid deelwoord (horizontal, wasHit, sunk). Wees nauwkeurig. Noem iets geen ‘ship’ als het de index van een schip in een lijst is (maar shipIndex). 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.

Deze aanpak leidt tot een flexibel ontwerp waaraan gemakkelijk zaken toe te voegen zijn, zoals meer dan twee spelers of nieuwe soorten schepen.

Niveau 1

Maak het spel "Zeeslagje" waarbij het de bedoeling is dat je schepen van je tegenspeler, voor de gelegenheid aangeduid met "de vijand", tot zinken brengt. Het spel wordt gespeeld met 2 spelers, iedere speler heeft xijn eigen "helft" van de zee, die uit 20 x 20 rechthoekige vakjes

bestaat. Er zijn drie soorten schepen: patrouilleboten (2 vakjes), kruisers (3 vakjes) en vliegdekschepen (4 vakjes). Schepen kunnen zowel horizontaal als verticaal liggen. De spelers kunnen elkaars schepen niet zien. Iedere speler voert x'n schepen dus voorafgaand in terwijl de andere speler niet kijkt. De speler die de beurt heeft mag schieten. Als xij iets raakt mag xij doorgaan met schieten. Als xij niets raakt is x'n beurt voorbij. De speler die het eerst alle schepen van de ander heeft vernietigd, heeft gewonnen. Het user interface mag werken met tekst, bijv. X voor een deel van een schip en - voor een leeg vakje. Richten mag worden gedaan door middel van intypen van coordinaten. Extra: Zorg dat je kunt richten met het keyboard zonder de coordinaten in te typen.

Niveau 2

Zelfde als bij niveau 1, maar nu speel je tegen de computer. De geautomatiseerde tegenstander is een object van class Agent. Let op: Het API van de menselijke speler en van de Agent zijn deels hetzelfde, omdat beiden in principe op dezelfde manier met het "singleton" (Google) object van klasse GameEngine communiceren. Los dit op met overerving van een abstracte Player basisklasse en een polymorfe verzameling van twee of meer Player's. Extra 1: Zorg dat de computer tegen zichzelf kan spelen. Exra 2: Zorg dat er, naast een Agent, 2 of meer menselijke spelers kunnen zijn.

Niveau 3

Zelfde als bij niveau 2, maar nu met een GUI. Dit mag bijv. een multiline text window zijn voor de zee met daarop zichtbaar de geraakte schepen en mouse- en/of keyboard control voor richten en buttons voor schieten. Extra 1: Een zee-background met plaatjes van schepen. Extra 2: Wil je helemaal los gaan? Maak dan een multiplayer game dat op meerder computers kan draaien waarbij de spelers verbonden zijn via "kale" TCP/IP sockets of web sockets. Het mag een desktop game of een browser game zijn. Voor een demo tijdens de assessment is het OK als de spelers op de zelfde fysieke computer spelen.