PROJECT TWO-PLAYER GAME FRAMEWORK

INLEIDING

Frameworks worden veel gebruikt bij het ontwikkelen van applicaties. De voordelen liggen voor-de-hand : een framework biedt hergebruik van softwarecomponenten, en omvat vaak ook afspraken hoe die componenten worden gebruikt. Je hebt al kennis gemaakt met diverse frameworks, bijvoorbeeld Java Swing, wellicht met Hibernate en ook Eclipse is een framework.

In dit project gaan we een applicatie ontwikkelen waarmee diverse spellen kunnen worden gespeeld. Hierbij kan het gaan om mens tegen mens, mens tegen computer en computer tegen computer.

CASUS

Een groot verzekeringsbedrijf heeft jullie gevraagd om een paar tweepersoons bordspelletjes te ontwikkelen. Deze moeten gespeeld kunnen worden als speler-speler, speler-computer of computer-computer. Het verzekeringsbedrijf wil de spellen en het overkoepelende framework gebruiken bij hun reclamecampagne onder het motto “Pijnig uw hersens, maar bezuinig niet op de polis”. Als het een succes wordt dan willen ze in de toekomst ook andere spelletjes aanbieden die op hetzelfde platform kunnen worden gespeeld. Herbruikbaarheid is dus van groot belang.

OPDRACHT

Ontwikkel een framework voor een “two-player strategic game”. Met behulp van dit framework moet het mogelijk zijn om een nieuw spel te spelen door eenvoudig de klassen daarvoor in het framework te “pluggen”. Het framework zelf mag voor een nieuw spel niet gewijzigd worden (dit is immers een eigenschap van een framework). Er is nog geen client. (Het zou mooi zijn als je de client zodanig ontwerpt dat er een spel-generiek deel en een spel-specifiek deel is). Ontwikkel als “proof-of-concept” de spellen boter-kaas-en-eieren (tic-tac-toe) en Othello (Reversi). Je proof-of-concept moet je demonstreren in een toernooi.

Let er op dat een speler moeten kunnen kiezen wie er begint, hij moet kunnen kiezen tussen zwart/wit (kruisje/rondje) en de beginstelling moet correct zijn.

RANDVOORWAARDEN

- Aanwezigheid 4 dagen/week op school is verplicht.
- Op www.assembla.com maak je een "space" aan voor je projectgroep. Het gebruik van SVN is verplicht. Op Assembla kun je het game framework uit SVN halen.

TIPS

- Na opstarten van de gameserver kun je bijvoorbeeld met Putty (http://www.putty.nl) twee connecties maken naar localhost:7789 en experimenteren met de servercommando's. Zet de bedenktijd in de settingsfile wat langer dan 10 seconden, anders krijg je te snel een timeout.
- Bij het maken van je plan-van-aanpak kan het document "projectmanagement" handig zijn. Zie blackboard.
- Verdiep je in de structuur van het framework. Bestudeer protocol.txt, ook het bijgevoegde sequence diagram maakt veel duidelijk. Maak voor jezelf een klassendiagram.
- Zoek in de Wikipedia eens naar "minimax".
- Wat is de impact van de requirement dat de programma's van verschillende projectgroepen tegen elkaar moeten kunnen spelen ?
- Doorloop het gehele software engineering traject (inclusief projectmanagement) zoals je dat geleerd hebt in de eerste weken. Datzelfde geldt voor de aangereikte tools, UML diagramtechnieken, programmeeronderwerpen, etc. Hoe meer je laat zien dat je het geleerde kunt toepassen, hoe beter je scoort.
- **Ga zo snel als mogelijk tegen elkaar spelen**, zodat je netwerk en protocol/interface kunt testen.
- Begin met een zeer eenvoudige AI implementatie.
- Over het hergebruik van bestaand materiaal:
  - Het is toegestaan om code te zoeken op het internet die je kan helpen bij het maken van de speelborden. Je moet dan wel duidelijk aangeven wat niet van jezelf is en waar je het vandaan hebt. Zorg ervoor dat die code er netjes uitziet en dat je begrijpt hoe het werkt, en dat je de code kunt uitleggen.
  - Het is toegestaan om speelstrategieën van de verschillende spellen op het Internet te bestuderen en te gebruiken. Zie bijvoorbeeld othello.nl
  - Het is niet toegestaan om code te kopiëren voor de algoritmes van de spellen. Het spelen van de spellen moet je als projectgroep zelf uitprogrammeren.

WAT LEVER JE OP ?

- demonstratie (op het eindtoernooi)
- broncode
- documentatie
  - plan-van-aanpak
  - handleiding installatie en gebruiker
  - eindverslag

WAT MOET ER (MINIMAAL) IN HET VERSLAG ?

Het verslag moet de volgende zaken bevatten :

- inleiding
- requirements
  - functionele requirements (use case diagram en toelichting; de use cases zelf kunnen in een bijlage)
  - niet-functionele requirements en beperkingen/randvoorwaarden
- architectuur
  - welke modulen (packages) zijn er
  - klasse diagrammen (per package) met toelichting
  - sequence diagrammen met toelichting
  - hoe deployment
- ontwerp
  - beschrijving user interface
  - client/server protool
  - beschrijving en uitleg AI algoritmen
- implementatie
  - o.a. uitleg essentiële stukken code
- testen
  - testaanpak (laat zien dat je het testen gestructureerd hebt aangepakt !)
  - testbeschrijvingen kunnen in een bijlage
  - testresultaten
- conclusie/aanbeveling (wat is er nog niet af ?)
- evaluatie (proces en samenwerking)

BEOORDELINGS-CRITEREA

- de werking van je programma zoals gedemonstreerd tijdens het eindtoernooi
- robuustheid applicatie
- het komen tot afspraken met andere groepen
- beheerst ontwikkelproces (SE methodiek en projectmanagement)
- taalgebruik en structuur verslag (BBC normen)
- kwaliteit en netheid code
- toegepaste AI (bonus voor de beste AI)
- kwaliteit verslag
- aanwezigheid op school
- inleveren assessment groepsgenoten (bonus/malus)

De beoordeling van bovenstaande leidt tot een *groepscijfer*. Daarnaast heeft de docent het recht om een *individuele bonus/malus (docent)* te geven als hij daar reden toe ziet. Tot slot wordt aan de groep gevraagd om feedback te geven op elkaars functioneren en ook een *individuele bonus/malus (student)* uit te reiken.