Project SE1 – KBS1

1. Opdracht

Maak in C# een spel. De initiële requirements staan verderop in dit document. Het project zal volgens de *Scrum* methodologie aangepakt worden. Er is echter voor dit project geen externe *product owner* beschikbaar. Deze rol neemt een van de studenten uit het *development team* op zich. Realiseer je wel dat dit voor *Scrum* ongebruikelijk is, Scrum verbiedt dit echter niet. De *product owner* vertolkt de wensen van het hele team.

Het gebruik van *Scrum tooling* (zoals *ScrumDo*, *Trello* en dergelijke) is tijdens dit project niet toegestaan! In plaats daarvan gebruiken je een fysiek scrumbord.

2. Opleveringen

Van iedere groep verwachten we een portfolio. Het portfolio bestaat uit de producten die jullie met de groep gemaakt hebben, het groepsverslag en de individuele verslagen. Het portfolio kan digitaal ingeleverd worden.

2.1. Beroepsproducten

- Een functioneel ontwerp en een technisch ontwerp. Het FO bevat o.a. de gerealiseerde backlogitems met testscenario's. Maak gebruik van UML! Laat zien dat je de verschillende UML diagrammen op een goede manier weet in te zetten.
- De implementatie van het spel in C#. Zorg ervoor dat het FO/TO en de code met elkaar in overeenstemming zijn. Vergeet niet de code van commentaar te voorzien (gebruik hiervoor bijvoorbeeld *XML comments*).
- De unit tests.
- Een beschrijving van de handmatig uitgevoerde testen.
- De gebruikershandleiding (inclusief installatiehandleiding).
- Scrumdocumentatie zoals verslag van demo's, verslag van retrospectives, burndowncharts, DoD.

2.2. Groepsverslag

In dit verslag leg je vast wat jullie gedaan hebben om de beroepstaken (op het niveau dat past bij dit semester) op het gebied van 'ontwerpen' en 'realiseren' aan te kunnen tonen. We verwachten reflecties op de beroepsproducten.
Om je te helpen geven we hieronder een aantal vragen die je jezelf zou kunnen stellen, zoals:

- welke keuze hebben we gemaakt en waarom?
- welke andere keuzes zijn overwogen en waarom zijn die afgevallen?
- hoe is de gemaakte keuze in de praktijk bevallen?
- welke problemen hebben jullie hiermee in de praktijk ervaren?
- hoe zijn jullie met deze problemen omgegaan?
- in welke mate hebben jullie je oorspronkelijke keuze moeten herzien?
- waren de oorspronkelijke argumenten goed?
- wat gaan jullie in een volgende situatie anders aanpakken of kiezen?
- wat is de leerervaring?

Het is niet de bedoeling deze vragen letterlijk, vraag-voor-vraag, te beantwoorden, maar ze juist te verwerken in een goed lezend, compleet verhaal.
Illustreer de reflectie op de diverse onderwerpen altijd met representatieve voorbeelden, door (delen van) documenten, diagrammen van ontwerpen, fragmenten van code of testresultaten, en dergelijke in je reflectie op te nemen (of er naar te verwijzen) en te bespreken.

2.3. Individuele verslagen

In je individuele verslag reflecteer je op je eigen functioneren in dit project. Omschrijf hoe de samenwerking wat jouw betreft is gelopen, wat je rol in de groep was en voor welke (deel)producten je verantwoordelijk bent geweest. Benoem hierbij de tasks binnen de userstories waar je verantwoordelijk voor bent geweest. Bovenstaande vragen (in 2.2) zouden een goede hulp kunnen zijn bij het beantwoorden van deze vragen.

De individuele verslagen hoeven niet al te groot te zijn. Je kunt denken aan 1 à 2 A4-tjes per student.

3. Beoordeling

Je wordt beoordeeld op de beroepstaken 'Ontwerpen' en 'Realiseren'. Beide onderdelen moeten voldoende zijn. We gebruiken hiervoor een uitgekleed afstudeerbeoordelingsformulier (beoordeling op twee i.p.v. vijf beroepstaken). De beoordeling is op niveau 2 (tijdens het afstuderen is dit niveau 3). De docent geeft feedback en beoordeelt met een voldoende of onvoldoende.

4. Weekplanning

Het uitvoeren van de opdracht wordt door de coachingsgroep gedaan d.m.v. van drie sprints van een week.

Aan het begin van de week (hopelijk lukt het rooster-technisch om alle klassen voor het begin van de week in te roosteren) sluiten we de vorige sprint af en starten we de nieuwe sprint op. De eerste week valt er nog niets af te sluiten, de eerste projectwerkdag staat in het teken van kick-off en de eerste sprint.

Tijdens een sprint zorg je steeds voor een:

- bijgewerkt FO
- bijgewerkt TO
- bijgewerkte *burn down chart*

Aan het einde van iedere projectdag zorgt iedere student voor een bijgewerkt individueel projectverslag.

Afsluiten van de sprint doe je d.m.v. een *sprint review (maximaal 10 minuten per groep)*:

- De *product owner* geeft aan wat '*done*' is en wat niet '*done*' is.
- Het *development team* vertelt wat goed ging tijdens de sprint, tegen welke problemen het team aangelopen is en hoe dit opgelost is;
- Het *development team* geeft een demonstratie van het gedane werk en beantwoordt vragen over het *increment*
- De docent en je klasgenoten zijn hierbij aanwezig

Na het houden van de *sprint review* ga je met je groep een *sprint retrospective* houden.

Starten van de sprint doe je d.m.v. een *sprint planning meeting (maximaal 2 uur)*:

- Maak daarbij gebruik van *scrum poker*. Het resultaat is een *sprint backlog*;
- De *product owner* vertolkt zo goed mogelijk het team en heeft het laatste woord als het gaat om het bepalen van de prioriteiten;
- De *scrum master* (docent) begeleidt deze meeting. De docent zal zijn/haar tijd zo goed mogelijk verdelen over de verschillende groepen;

5. Voorwaarden

- Iedereen in de projectgroep levert een bijdrage aan de documentatie én de implementatie van het product.
- De code en documentatie wordt in SVN gezet en elk projectlid gebruikt een eigen account. Bij elke commit moet aangegeven worden wat diegene toegevoegd/veranderd heeft.

6. Definition of Done

Stel zelf een Definition of Done op. Hierin moeten in elk geval de volgende punten staan:

- *Unit tests* geschreven en deze geven 'groen licht';
- Overige testgevallen beschreven en uitgevoerd, code bevat geen fouten meer;
- De code is voorzien van commentaar;
- De code is 'gereviewed';
- Broncode '*gecommitted*' op de server;
- De applicatie is 'gedeployed' op de laptops van alle studenten (de applicatie kan draaien buiten de IDE om);
- De gebruikershandleiding (inclusief installatiehandleiding) is bijgewerkt.
- Functioneel ontwerp is bijgewerkt en '*gecommitted*' op de server;
- Technisch ontwerp is bijgewerkt en '*gecommitted*' op de server;
- De *How-to-Demo* is gecontroleerd en kan dus zonder problemen tijdens de review uitgevoerd worden;
- De individuele projectverslagen zijn bijgewerkt;

7. Gameloop

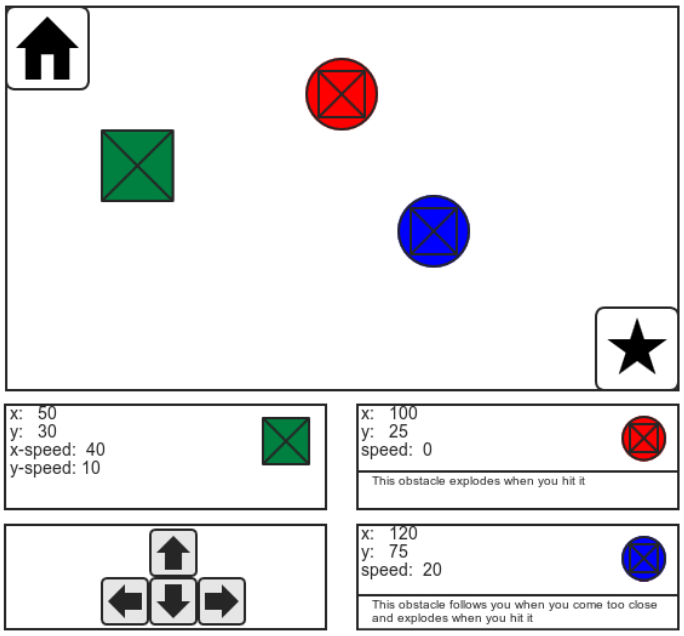
De Timer class is niet ontworpen met het idee om te gebruiken als de 'engine' voor een snelle game loop. Toch volstaat een timer zeer waarschijnlijk voor dit project. Doe in ieder geval geen poging om *threads* te gebruiken, dat is op dit moment nog een stap te ver.

8. Het Spel

Hieronder is een schematische weergave van het spel te zien. Het is de bedoeling dat de speler met behulp van de pijltoetsen of de pijltjes op het scherm een object (het groene blokje) van het huisje naar het sterretje beweegt. Op het veld worden obstakels geplaatst die het object verhinderen om bij het eindpunt te komen. De obstakels hebben verschillende eigenschappen waardoor ze elk op hun eigen manier het object hinderen. Voorbeelden van obstakels zijn obstakels die:

- Ontploffen als het object er tegenaan botst
- Je volgen en ontploffen als het object tegenaan botst
- Gewoon in de weg staan

- De snelheid van het object beïnvloeden als het object binnen een bepaalde straal komt



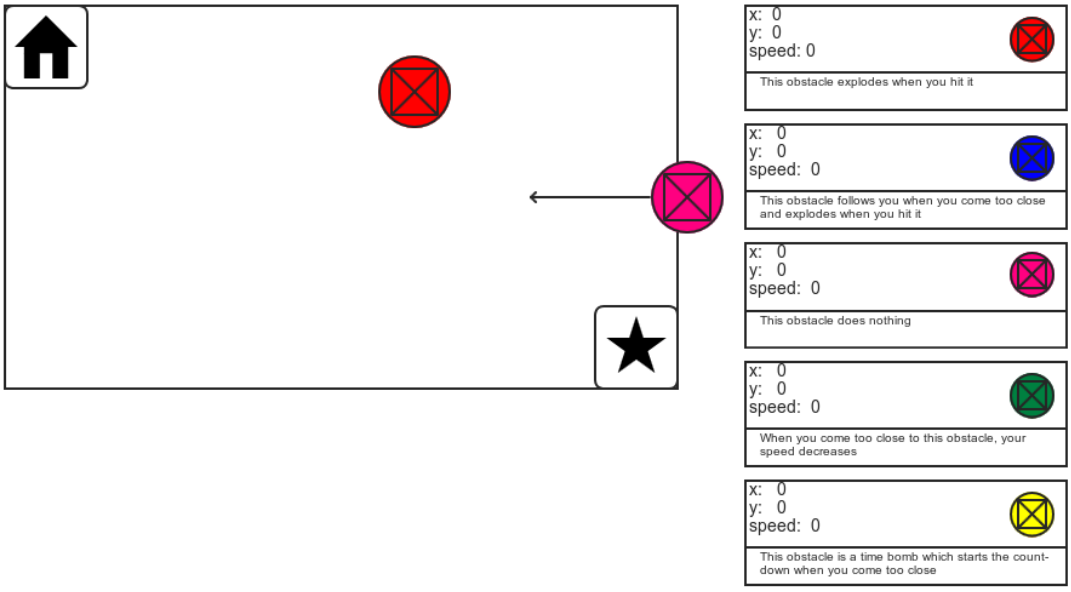
Op het scherm zie je het speelveld en daaronder rechts de gegevens van de obstakels en links de gegevens van het object en de besturing. Dit is echter geen vaste lay-out. De lay-out moet makkelijk aan te passen zijn, ook andere manieren van besturing moeten makkelijk toegevoegd kunnen worden. Dit is heel goed mogelijk als je projectcode een MVC-structuur heeft.

8.1. Requirements

- Maak voor het ontwerp gebruik van MVC gecombineerd met een game-loop. De gameloop zou je kunnen zien als een soort 'super controller'.
- Voorzie de code van unit testen.
- Er moeten duidelijk OO-principes terugkomen in het TO en dus ook in de implementatie.
- Het spel moet zo opgezet zijn dat er eenvoudig nieuwe obstakels met nieuwe eigenschappen toegevoegd kunnen worden. Denk hier goed over na en gebruik de behandelde OO-principes om dit voor elkaar te krijgen.
- Er moeten makkelijk nieuwe controllers en nieuwe views toegevoegd kunnen worden.
- Het is de bedoeling dat er uit een menu een speelveld geladen kan worden aan de hand van een XML-bestand.

9. Het Veld (bonus)

In het spel zit een editor waarmee je een speelveld kan ontwerpen. Dit werkt heel eenvoudig door de verschillende obstakels uit de lijst rechts op het scherm te slepen. Dit speelveld kan dan vervolgens worden opgeslagen en in het spel worden geladen.



9.1. Requirements

- Alle gedefinieerde Obstakels worden automatisch ingeladen.
- De obstakels worden met behulp van drag&drop op het veld geplaatst.
- Het opslaan van het speelveld wordt gedaan m.b.v. XML zodat deze ingeladen kan worden door het spel.

10. Naslagwerk

- Voorbeeld van MVC + game + java: https://www.javacodegeeks.com/2012/02/building-games-using-mvc-pattern.htm
- Voor meer informatie over MVC zie: http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller
- SVN: http://www.visualsvn.com/visualsvn/