

Code Design

Deelopdracht 1: The Temple of Doom

Samenvatting

Wat	C# Console applicatie
Wie	Maak dit in een duo
Deadline	Zondagavond week 5 (12-12-2021 23:59)
Inleveren	<p>Brightspace:</p> <ul style="list-style-type: none">- Je toelichtingsdocument- Je code (zip)- Je zelfingevulde beoordelingsmodel <p>Github classrooms:</p> <ul style="list-style-type: none">- Je repository dient de meest recente versie van je code te hebben. <p>(Zie verderop in dit document)</p>



https://en.wikipedia.org/wiki/Indiana_Jones_and_the_Temple_of_Doom

Het is het jaar 1935...

...en jij, Indiana Jones, bent in een tempel beland op zoek naar 5 Sankara stones. Vind deze 5 stenen zodat je deze aan het volk van de Mayapore kan teruggeven en ze kan redden van een zekere ondergang. Easy enough right?

Wrong, deze tempel heet niet voor niets de Temple Of Doom! De tempel bestaat uit vele kamers met in iedere kamer weer boobytraps en oude, cryptische puzzels die opgelost moeten worden voordat je verder kan. Help Indy met het vinden van de 5 stenen voordat de Tempel je insluit en je verzwolgen wordt door de vele boobytraps!

De opdracht

Schrijf een applicatie die deze casus realiseert. Gebruik hiervoor de lesstof van het vak Code Design en lever het volgende in voor de bovengenoemde deadline:

- Toelichtingsdocument inclusief klassendiagram waarmee je de door jullie gebouwde oplossing uitlegt Denk hierbij (naast het klassendiagram) aan:
 - o Toelichtingen bij het klassendiagram;
 - o Eventueel benodigde sequence diagrammen;
 - o Eventueel code die toelichting behoeft;
 - o Et cetera.
- (Zelf)ingevuld beoordelingsformulier;
- De broncode die nodig is om deze applicatie te realiseren.

Je maakt deze opdracht in een duo met een klasgenoot. Deze opdracht telt voor 25% van je eindcijfer.

De exacte vervolgoopdracht zal aan de start van week 6 bekend gemaakt worden, maar houd er rekening mee dat...

- ...de vervolgoopdracht een uitbreiding is op deze opdracht.
Maak de oplossing dus onderhoudbaar en uitbreidbaar.
- ...de vervolgoopdracht een individuele opdracht is.
Zorg er dus voor dat je jullie uitwerking van binnen en buiten kent om vervolgens door te kunnen werken.
- ...de vervolgoopdracht anders is voor jou dan voor je duo-partner.
- ...de vervolgoopdracht voor de overige 75% van je eindcijfer van het vak meetelt.

De applicatie

De applicatie die jullie gaan maken zal een C# Console Application worden. In deze applicatie kan je met de pijltjestoetsen door een speelwereld bestaande uit meerdere kamers heen lopen. Elke keer zie je slechts de kamer waarin je je bevindt. In deze kamer kunnen zich items bevinden die je oppakt zodra je over ze heen loopt. Daarnaast heeft een kamer tot maximaal 4 uitgangen: In het noorden, oosten, zuiden en westen. Deze uitgangen zijn altijd in het midden van een wand (in onze bestanden zijn de hoogte en breedte van een kamer altijd een oneven getal). Echter wordt deze uitgang soms geblokkeerd door een deur die enkel geopend kan worden wanneer je een sleutel hebt die dezelfde kleur als de deur heeft.

Indy start altijd met een aantal levens en verschillende boobietraps kunnen er voor zorgen dat je levens verliest. Zodra het aantal levens op 0 staat heb je de game verloren en zou je opnieuw moeten spelen. Zodra je de 5 Sankara stones in je bezit hebt, heb je direct gewonnen.

De levels zijn opgeslagen in JSON bestanden, deze dien je in te lezen in je applicatie zodat je levels dynamisch opgebouwd zijn.

Het is de bedoeling dat je de GIT-repository maakt via Github classrooms. Volg de instructies op de volgende URL: <https://classroom.github.com/a/kRy-38BW>

Onderneem de daarbij volgende stappen:

- Maak een team aan volgens de naamgeving: {voor+achternaam student 1}_{voor+achternaam student 2};
- Creëer een git repository volgens de instructies;
- Gebruik deze git repository voor je code (de documentatie zal op blackboard ingeleverd worden)

Let op dat je van meet af aan in deze repository werkt. We kunnen dit gebruiken om inzicht te krijgen in jullie individuele bijdrage aan de code in jullie uitwerking.

Requirements

Functionele eisen

De speler kan...

- ...(enkel) de ruimte waar hij zich op dat moment in bevindt zien;
 - Een ruimte is volledig omringd door niet doorgaanbare muren.
 - Uitzondering:
In elke windrichting (Noord, Oost, Zuid, West) kan een opening in het midden van de muur zitten, dit is een doorgang naar een aangrenzende kamer;
- ...zien waar hij zich in de ruimte bevindt;
- ...middels de pijltjestoetsen naar andere posities in de ruimte bewegen;

- ...de items die geplaatst zijn op hun positie in de kamer zien;
- ...interactie hebben met de items zodra hij zich op dezelfde positie als een item bevindt;
- ...door een open doorgang naar de volgende ruimte gaan;
- ...zien dat hij gewonnen heeft zodra hij zijn vijfde Sankara Stone oppakt;
- ...zien dat hij verloren heeft zodra zijn levens op zijn.

Technische eisen

De applicatie...

- ...kan de bijgevoegde JSON- en XML-bestanden inlezen;
Tip: Kijk eens naar het strategy-patroon om dit te realiseren.
- ...is een Console applicatie dat geschreven is in C#;
- ...code is opgebouwd middels libraries waarin je de applicatielagen scheidt: Eén voor de user interface (in- en output), één voor de game-logica en één voor het uitlezen de leveledata uit de JSON-bestanden;
 - Enkel in de visualisatie library mag naar de Console output geschreven worden;

Bestandsformaat van een level-bestand

Het te lezen bestand is een JSON-bestand. Dit bestand is bijgevoegd op blackboard.

Het bestand bestaat uit de volgende onderdelen:

Root-object

Dit heeft 3 properties:

- **rooms**: Een array van room objects;
- **connections**: Een array van connection objects;
- Een **player** object met:
 - o **startRoomId** (int): de startruimte;
 - o **startX** (int): startpositie x in de startruimte (0-based);
 - o **startY** (int): startpositie y in de startruimte (0-based);
 - o **lives** (int): aantal levens van de speler;

```
{
  "rooms": [...],
  "connections": [...],
  "player": {
    "startRoomId": 1,
    "startX": 2,
    "startY": 2,
    "lives": 3
  }
}
```

Room-object

Dit heeft de volgende properties:

- **id** (int);
- **type** (string): type ruimte, nu altijd "room";
- **width** (int): breedte $3 \leq x \leq 50$;
- **height** (int): hoogte $3 \leq x \leq 50$;
- **items** (array), de items in de ruimte, met:
 - o **type** (string): het type item (later toegelicht)
 - o **x** (int): x-locatie in de ruimte (0-based);
 - o **y** (int): y-locatie in de ruimte (0-based);
 - o *Optioneel: typespecifieke eigenschappen;*
(In dit voorbeeld is "damage" een typespecifieke eigenschap)

```
{
  "id": 3,
  "type": "room",
  "width": 5,
  "height": 5,
  "items": [
    {
      "type": "disappearing boobietrap",
      "x": 2,
      "y": 1,
      "damage": 1
    }
  ]
}
```

Connection-object

Dit heeft de volgende properties:

- **NORTH** (optioneel, int): Het id van de ruimte aan de noordkant;
- **EAST** (optioneel, int): Het id van de ruimte aan de oostkant;
- **SOUTH** (optioneel, int): Het id van de ruimte aan de zuidkant;
- **WEST** (optioneel, int): Het id van de ruimte aan de westkant;
- **door** (array): Een array met de eigenschappen van de deur. Een deur kan dus verschillende eigenschappen tegelijk hebben.

Tip: Bekijk eens of je een decorator kan inzetten hiervoor.

Per object in deze array de eigenschappen van de connectie, met:

- o **type** (string): het type item (later toegelicht)
- o *Optioneel: typespecifieke eigenschappen;*
(In dit voorbeeld is "color" een typespecifieke eigenschap)

```
{
  "WEST": 4,
  "EAST": 7,
  "door": [{
    "type": "colored",
    "color": "red"
  }, {
    "type": "closing gate"
  }]
}
```

Type items in de ruimte

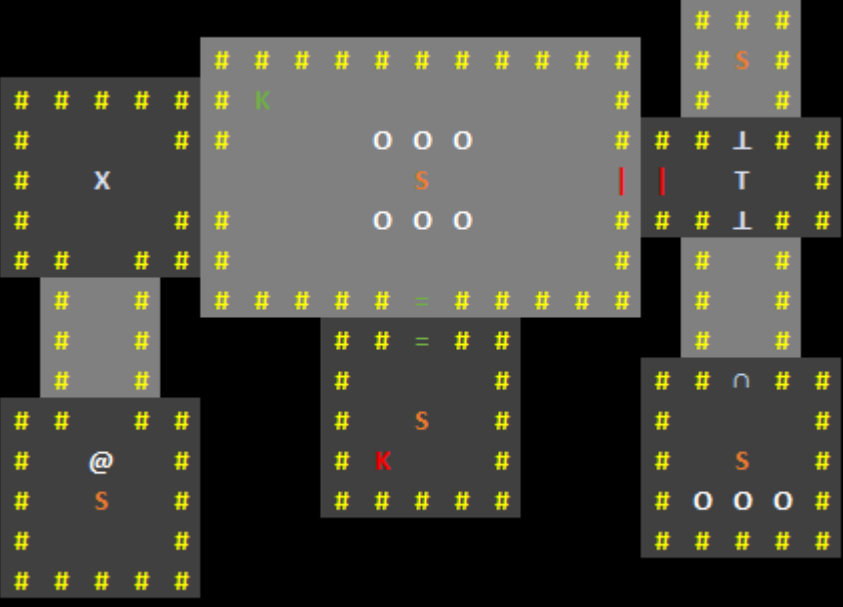
Alle items bieden interactie met je zodra je er op gaat staan.

- **“boobytrap”**
Interactie: Hij doet je schade.
Extra properties:
 - o **damage** (int): De hoeveelheid schade die hij doet;
- **“disappearing boobytrap”**
Interactie: Hij doet je schade en verdwijnt vervolgens uit het spel.
Extra properties:
 - o **damage** (int): De hoeveelheid schade die hij doet;
- **“sankara stone”**
Interactie: Indy pakt deze op en de stone telt mee voor de 5 stones die je moet hebben om het spel te winnen.
- **“key”**
Interactie: Je pakt deze op en draagt hem dan bij je.
Extra properties:
 - o **color** (string): De kleur van de sleutel.
- **“pressure plate”**
Interactie: Opent alle gesloten toggle-deuren in dezelfde ruimte én sluit alle geopende toggle-deuren in dezelfde ruimte.

Type deuren

- **“colored”**
Initiële staat: Gesloten
Interactie: Opent enkel als je een sleutel met dezelfde kleur bij je draagt.
Extra properties:
 - o **color** (string): De kleur van de deur.
- **“toggle”**
Initiële staat: Gesloten
Interactie: Geen (maar zie items/pressure plate).
- **“closing gate”**
Initiële staat: Open
Interactie: Zodra je deze deur door bent, sluit hij en kan hij niet meer open.
- **“open on odd”**
Interactie: Je kan alleen door deze deur als je een oneven aantal levens hebt.
- **“open on stones in room”**
Interactie: Je kan alleen door deze deur als er in de kamer exact het aantal stenen aanwezig zijn als deze deur nodig heeft.
Extra properties:
 - o **no_of_stones** (int): Het exacte aantal Sankara Stones dat in de kamer (dus de nog niet opgepakte stenen) aanwezig moet zijn om de deur open te laten zijn.

TempleOfDoom.JSON visuele representatie

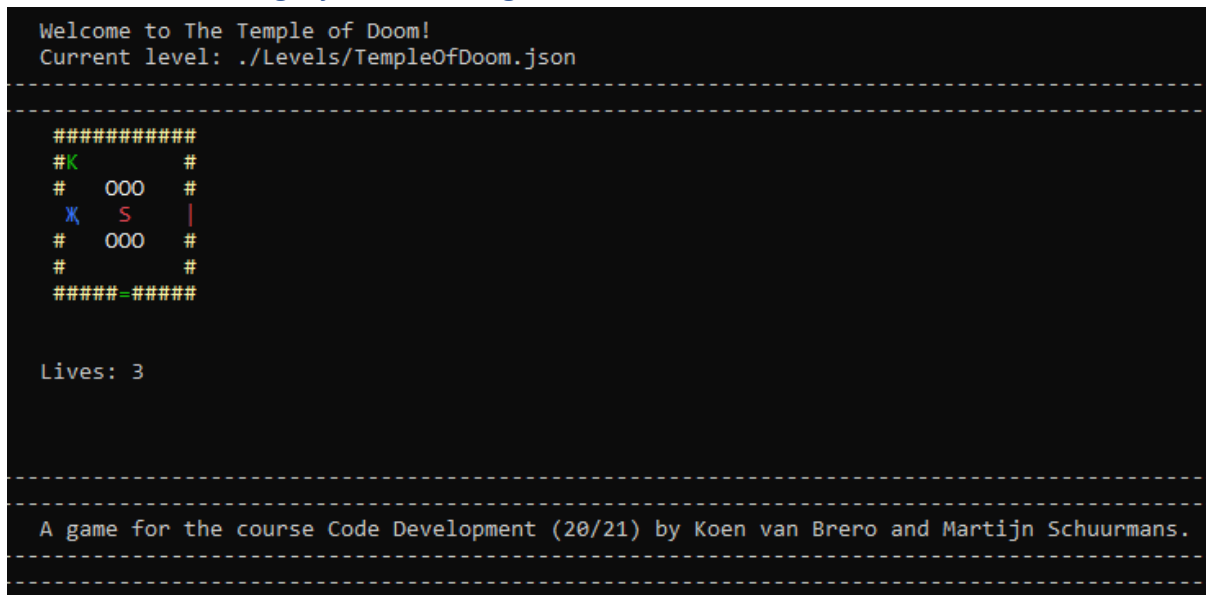


Legenda:

- # Muur
- X Startpositie van Indy
- S Sankara Stone
- K Sleutel met betreffende kleur
- | Deur (verticaal)
- = Deur (horizontaal)
- T Drukplek, zodra je hier op stapt gaan alle dichte deuren ⊥ in dezelfde ruimte open en alle open deuren ⊥ in dezelfde ruimte dicht.
- ⊥ Een deur die met drukplek open of dicht kan gaan. Initiële status is dicht.
- ∩ Een deur die open staat, maar dicht gaat (en voor altijd dicht blijft) als je er doorheen gaat.
- O Een boobytrap die je schade doet.
- @ Een boobytrap die je schade doet en vervolgens verdwijnt uit de game.

Kamers hebben voor de duidelijkheid een andere achtergrondkleur gekregen. In de applicatie is dat niet nodig.

Screenshot van mogelijke uitwerking



Codevoorbeeld parsing JSON

Je kan JSON heel erg gemakkelijk importeren in C#. Visual Studio kan je helpen door zogenaamde DTO's (Data Transfer Objects, https://en.wikipedia.org/wiki/Data_transfer_object) te genereren. Dit zijn objecten zonder logica en reflecteren enkel de data zoals deze in je JSON file staan.

Je zult dus zelf nog je domeinklasses moeten maken en je objecten vanuit je DTO's om te zetten naar de domein-objecten.

Hoe maak je jouw DTO-klasses?

1. Maak een nieuwe file in Visual Studio en zorg dat er enkel de namespace declaratie in staat. (Verwijder dus de klasse);
2. Kopieer alle inhoud van je JSON-bestand;
3. Selecteer in Visual Studio in je hoofdmenu: Edit -> Paste Special -> Paste JSON as Classes;
4. Geef je klasse RootObject een wat meer zinnige naam;
5. Klaar!

Nu kan je de volgende code gebruiken en je JSON-bestand is ingelezen:

```
Rootobject root = JsonSerializer.Deserialize<Rootobject>("path.to.my.json");
```

Voor meer informatie zie:

<https://davecallan.com/creating-strongly-typed-dto-classes-from-a-json-string-in-visual-studio/>

Codevoorbeeld parsing XML

Zoals hierboven beschreven kan je ook gemakkelijk XML plakken als DTO-klasses. Daarna kan je ze inlezen met de volgende code:

```
XmlSerializer serializer = new XmlSerializer(typeof(Rootobject));  
FileStream fileStream = new FileStream(@"path.to.my.xml", FileMode.Open);  
Rootobject temple = (Rootobject)serializer.Deserialize(fileStream);
```