

*Assignments*

C++ Programming & Software Engineering 2

**TICT-V2CPSE2-16**

Studiejaar 2017-2018

|  |  |
| --- | --- |
|  |  |
| Cursuseigenaar | Wouter van Ooijen |
| **Auteur(s)** | Wouter van Ooijen |
| **Datum** | 2017-11-19 |
| Versie | 2.0 |
|  | |

© Institute for ICT, Hogeschool Uterecht, 2017

# Algemeen

Alle opgaven moeten individueel gemaakt worden en aan de docent getoond worden ter goedkeuring. Alle opgaven moeten voldoende beoordeeld worden.

De herkansing voor de practica gaat over de zelfde practica. Voor de herkansing hoeven dus alleen de practica gedaan te worden die nog niet als voldoende beoordeeld waren.

De practica zijn een aparte entry in Osiris, een voldoende voor alle practica blijft dus staan, ook als het tentamen niet voldoende is.

Als slechts een deel van de practica voldoende gemaakt is dan is het mogelijk dat in een volgend jaar een nieuwe set practica gedaan moet worden.

# Practicum opgaven

Bij het uitwerken van de opgaven mag je klassen die logisch bij elkaar horen bij elkaar voeg in het zelfde .hpp en .cpp duo. Je hoeft dus niet perse voor iedere klasse een apart file duo te maken.

## Week 1 : Installeren, video bekijken

* Installeer MSVS (Microsoft Visual Studio), SFML, en boost.
* Check dat MSVC en SFML werken aan de hand van het 01-02 project.
* Maak zelf een project aan en check dat je kan compileren en builden.

Het installeren duurt wel even. Dan heb je dus de tijd op deze video te bekijken. Dit is niet vrijblijvend, het singleton template is onderdeel van de stof. Overigens heeft google nog veel meer interessante ‘tech talks’ op youtube staan.

<https://www.youtube.com/watch?v=-FRm3VPhseI>



## Week 2 : Muren en stuiteren

Neem de code van het einde van hoofdtsuk 7 (example 03-08) als uitgangspunt. Op het scherm moet je de bal, 4 muren, en een blokje afbeelden. Geef die drie soorten objecten verschillende kleuren. De bal beweegt uitzichzelf over het scherm, en stuitert (op een realistische manier, ) tegen de muren en tegen het blokje. Het blokje kan je bewegen met de cursor toetsen.

Aanbevolen stappen:

* Maak voor de scherm objecten een hierargie met een superklasse die in om te beginnen een virtele draw() functie heeft.d
* Hou de main loop sober. Definieer de 6 objecten in de main(), en voeg de draw calls toe in de main loop. Je hebt nu het statische deel van de applicatie, en het blokje reageert op de cursor toetsen.
* Het bewegen van de bal is een actie die onvoorwaardelijk gebeurt. Voeg daarvoor een constructor toe in de action klasse, en een virtuele methode in de schermobject basis klasse (voor de andere objecten is die nu nog niet nodig, maar dat lijkt een reeele optie voor de toekomst). Je hebt nu een applicatie met een bewegende bal.
* SFML heeft een functie voor het bepalen of twee objecten overlappen. Gebruik die voor het stuiteren. Voeg aan de lijst van actions de actions toe die het stuiteren implementeren.

## Week 3 : Factory

Maak een klasse hierargie van beweegbare scherm objecten. Handel het selecteren en bewegen af in de superklasse.

Maak een factory function die scherm objecten creeert: circle, rectangle, line, picture. Ieder object heeft een positie, en behalve picture ook nog een kleur, circle heeft een diameter, en line en rectange een tweede punt (positie is het eerste punt). Check alle fouten die kunnen optreden en genereer daarvoor passende exceptions. Let ook op einde-file.

Maak een applicatie die

* De objecten lees uit een text file en toont,
* Je de gelegenheid biedt ze te slecteren en te verschuiven,
* Bij het afsluiten de textfile overschrijft met de nieuwe lijst van objecten.

Sla de (pointers naar) de objecten op in een container waar je tijdens het afbeelden en afhandeklen van gebruikershandelingen met een for( : ) loop doorheen loopt.

Let op wat je doet als er een fout wordt geconstateerd in de textfile. Zorg er in ieder geval voor dat de gebruiker duidelijk te zien krijgt wat de fout is, en dat de textfile niet overschreven wordt.

## Week 4 : STL

Haal van project gutenberg de txt versie van de King Jame’s Bible.

1. Lees alle characters van deze file in een vector.
2. Print hoeveel characters de file bevat (vraag het aan de vector).
3. Print hoeveel regels de file bevat (gebruik de vector en een STL algorithme).
4. Print hoeveel alfabetische characters de file bevat (vector + algorithme).
5. Zet de letters in de vector om in kleine letters (max 3 regels code).
6. Tel in een lijst voor iedere letter (a..z) hoe vaak hij voorkomt in de vector (diverse manieren mogelijk).
7. Druk deze lijst af 1) op lettervolgorde 2) op hoe vaak een letter voorkomt (gebruikt een algoritme).
8. Bepaal welke woorden er in de tekst voorkomen en druk de 10 meest voorkomende woorden af. Een woord is aaneengesloten reeks letters. (gebruik een map)

## Week 5 : Tic-Tac-Toe

Schrijf een Tic-Tac-Toe (3-op-een-rij) spelletje. Maak hiertoe een object voor het spel en een object voor een zet (command pattern). Sla het verloop van het spel op in een STL container. Implementeer een ‘undo’ feature door het her-afspelen van de gedane zetten behalve de laatste. Biedt, met zo min mogelijk code duplicatie, zowel een grafische (SFML) interface als een tekst-only interface.