Afbeelding met buiten, gebouw, gras, spel

Automatisch gegenereerde beschrijving

Eindopdracht C++ en 3D Graphics

Technische Informatica

Datum: 11-6-2023

Door: Merijn Gommeren

Studentnummer: 2181491

Inhoudsopgave

[Inleiding 3](#_Toc137393267)

[1. Requirements C++ 4](#_Toc137393268)

[2. Requirements 3D 6](#_Toc137393269)

# Inleiding

In dit document worden de eisen van C++ en 3D Graphics doorgenomen en vergeleken met mijn eindopdracht en waar ze te vinden zijn.

# 1. Requirements C++

Ik voldoe aan alle eisen voor een voldoende.

|  |  |  |
| --- | --- | --- |
| **Eis** | **Voldaan?** | **Waar te vinden** |
| De code bevat geen overbodige #defines en bevat geen magische constanten. | Voldaan | Nergens in de code staan onnodige defines en er zijn geen magische constanten. Elke waarde heeft een duidelijk variable name of een comment. |
| Gebruik maken van OO-principes in C++, zoals gebruik van klassen met constructors en destructors, polymorfisme en/of inheritance. | Voldaan | Ik maak gebruik van de Component structuur. Hierin zit polymorfisme en inheritance. Daarnaast heeft elke klasse een Constructor en een Destructor, deze is wel vaak leeg. **Bijvoorbeeld:** Component.h (update) DrawComponent.h MoveToComponent.h FloorTile.h Texture.cpp |
| De code gebruikt op ten minste 1 plek op een zinnige wijze van overerving en virtuele  functie(s). | Voldaan | In mijn eindopracht gebruik ik de Component structuur. Hierbij zitten virtuele functies en overerving. **Bijvoorbeeld:** PlaneComponent.h MoveToComponent.h Component.h |
| De code heeft ten minste 1 unit-test waarbij het resultaat in een bestand wordt  weggeschreven. | Voldaan | Er is een UnitTest aanwezig die test of de kleur rood binnen een bepaalde range valt. Vervolgens wordt dit geschren naar test-log.txt. **Bijvoorbeeld:** test.cpp |
| Met behulp van FileIO, bestanden kunnen lezen en schrijven | Voldaan | Bij mijn UnitTest maak ik gebruik van FileIO, daarnaast gebruik ik FileIO om bestanden in te laden in ObjModel. Ook Gebruik ik FileIO voor Scores op te slaan in GameManager. **Bijvoorbeeld:** test.cpp ObjModel.cpp 80 GameManager 110 |
| Toepassen van standaard datastructuren in C++. | Voldaan | Ik maak gebruik van Enums, Lists, en Vectors.  **Bijvoorbeeld:** GameStatus.h GameManager.h |
| Gebruik maken van streams | Voldaan | Bij de FileIO maak ik gebruik van streams.  **Bijvoorbeeld** ObjModel.cpp GameManager.cpp test.cpp |
| Rekening gehouden met het voorkomen van memory leaks. | Voldaan | Bijna overal maak ik gebruik van shared\_ptr. Dit helpt met het automatisch beheren van geheugen. Het wordt ook vrijgegeven als het niet langer in gebruik is. |
| Zinnige foutmeldingen mocht er iets fout gaan, afhandeling via exceptions | Voldaan | Bijvoorbeeld bij FileIO wanneer het bestand niet kan worden geopend, wordt er een error message gegeven. **Bijvoorbeeld:** test.cpp 17 main.cpp 42 / 48 GameManager.cpp 121 / 135 |
| Gebruik maken van C++ standaard, dus niet op C manier alloceren. | Voldaan | Ik maak vooral gebruik van std::shared\_ptr. Daarnaast maak ik gebruik van new en delete. |
| Weloverwogen gebruik van algoritmes binnen de applicatie | Voldaan | Bijvoorbeeld bij de MoveToComponent heb ik gekozen om een minimalesnelheid te implementeren. Dit zorgt ervoor dat het object daadwerkelijk op locatie aan komt, anders gaat hij steeds langzamer. Ook heb ik gekozen voor een ForceComponent, zodat deze alleen collision test en de andere componenten die een object willen laten bewegen voegen een force toe aan deze component. **Bijvoorbeeld:** BoundingBoxComponent.cpp ForceComponent.cpp PlayerComponent.cpp |

Extra eisen:

|  |  |  |
| --- | --- | --- |
| **Extra Eis:** | **Voldaan?** | **Waar?** |
| Smart pointers. | Voldaan | Ik gebruik eigenlijk overal std::shared\_ptr **Bijvoorbeeld:** main.cpp |

# 2. Requirements 3D

Ik voldoe aan alle gevraagde eisen voor 3D Graphics.

|  |  |
| --- | --- |
| **Eis:** | **Voldaan?** |
| OpenGL gebruiken voor het weergeven van 3D graphics | Voldaan. |
| Een camera hebben die bestuurd kan worden | De camera kan bestuurd worden met de muis en WASD. |
| Minimaal 5 verschillende objecten bevatten | Er is een Bulbasaur, papieren vliegtuigje, muur, camera / player, hart, gebroken hart en een vleermuis. |
| Een of meerdere objecten bewegen automatisch | Blokken vallen automatisch. |
| Een of meerdere objecten zijn door de gebruiker te beïnvloeden | De gebruiker kan de player beïnvloeden. |
| Een van de objecten is een ‘complex’ object | Er zijn meerdere 3d modellen. Daarnaast zijn er kubussen en rechthoeken. |
| Belichting gebruiken | Er is belichting. |
| Texture mapping gebruiken | Er wordt texture mapping gebruikt op elk 3d model. De blokken hebben een kleur. |

Extra eisen:

|  |  |
| --- | --- |
| **Extra eis:** | **Voldaan?** |
| Complexe scene met meer objecten | Mijn game bevat muren, vallende objcten waardoor het een spel wordt. |
| Een samenhangende scene | Mijn eindopdracht is een spel, waarbij veel een functie heeft. |